Reinforcement Learning in Continuous Time and Space

• From K. Doya, Neural Computation 12,219-245, 2000
Continuous Time Discounted Value Function

- Continuous time dynamical system: \( \dot{x} = f(x(t), u(t)) \)
- Reward: \( r(t) = r(x(t), u(t)) \)
- Policy: \( u(t) = \mu(x(t)) \)
- The policy’s decaying (i.e. discounted) value function:
  \[
  V^\mu(x(t)) = \int_t^\infty e^{-\frac{s-t}{\tau}} r(x(s), u(s)) \, ds
  \]
- Optimal policy’s value function
  \[
  V^\ast(x(t)) = \max_{u[t, \infty)} \left[ \int_t^\infty e^{-\frac{s-t}{\tau}} r(x(s), u(s)) \, ds \right]
  \]
Continuous Time HJB for Discounted Rewards

- Separate integral into \([t, t+\Delta t]\) and \([t+\Delta t, \infty)\):

\[
V^* (x(t)) = \max_{u[t, t+\Delta t]} \left[ \int_t^{t+\Delta t} e^{-\frac{s-t}{r}} r(x(s), u(s)) ds + e^{-\frac{\Delta t}{r}} V^* (x(t+\Delta t)) \right]
\]

\[
\approx r(x(t), u(t)) \Delta t
\]

- Approximate \(V^*(x(t+\Delta t))\) by Taylor 1st degree

\[
V^* (x(t+\Delta t)) \approx V^*(x(t)) + \frac{\partial V^*}{\partial x(t)} f(x(t), u(t)) \Delta t
\]

- Plug in and rearrange a bit

\[
\left(1 - e^{-\frac{\Delta t}{r}}\right) V^*(x(t)) = \max_{u[t, t+\Delta t]} \left[ r(x(t), u(t)) \Delta t + e^{-\frac{\Delta t}{r}} \frac{\partial V^*}{\partial x(t)} f(x(t), u(t)) \Delta t \right]
\]
• Take $\Delta t \to 0$

$$\frac{1}{\tau} V^* (x(t)) = \max_{u(t)} \left[ r(x(t), u(t)) + \frac{\partial V^*}{\partial x(t)} f(x(t), u(t)) \right]$$

* compare with original HJB

$$- \frac{\partial J^0}{\partial t} = \min_{u(t)} \left\{ L(x, u(t), t) + \frac{\partial J^0}{\partial x} f(x, u(t), t) \right\}$$

• Solution approach: GPI

**Must use function approximators!**
Learning the Value Function

- Use function approximator with parameter vector $w$: $V^\mu (x(t)) \approx V(x(t); w)$
- by HJB: $\frac{1}{\tau} V^\mu (x(t)) = r(x(t), \mu(x(t))) + \frac{\partial V^\mu}{\partial x(t)} f(x(t), \mu(x(t)))$

  i.e. $\dot{V}^\mu (x(t)) = \frac{1}{\tau} V^\mu (x(t)) - r(t)$

- Define the inconsistency (TD error) as $\delta(t) \equiv r(t) - \frac{1}{\tau} V(t) + \dot{V}(t)$

- Reduce inconsistency by correcting weights:

  $\dot{w} = \eta \delta(t) \frac{\partial V(x(t), w)}{\partial w}$

  where $\eta$ is a scaling factor

- This is TD(0)
Correction decays exponentially. I.e. the desired correction due to the current discrepancy is

\[
\hat{\nabla}(t) = \begin{cases} 
\delta(t_0)e^{-\frac{t_0-t}{\tau}} & t \leq t_0, \\
0 & t > t_0,
\end{cases}
\]

The weights should therefore be updated by

\[
\dot{\mathbf{w}} = \eta\delta(t_0) \int_{-\infty}^{t_0} e^{-\frac{t_0-t}{\tau}} \frac{\partial V(\mathbf{x}(t), \mathbf{w})}{\partial \mathbf{w}} dt
\]

The eligibility can be computed as a linear (time varying) dynamical system

\[
\dot{\mathbf{w}} = \eta\delta(t)e(t)
\]

\[
\dot{e}_i(t) = -\frac{1}{\kappa}e(t) + \frac{\partial V(\mathbf{x}(t), \mathbf{w})}{\partial \mathbf{w}}
\]

where \(\kappa\) is a time decay constant.
Policy Improvement by Value Gradient

- If we know \( r(x(t), u) \) and \( f(x(i), u) \) we can select action that maximizes expected reward:

\[
    u(t) = \mu(x(t)) = \arg \max_{u \in U} \left[ r(x(t), u) + \frac{\partial V(x)}{\partial x} f(x(t), u) \right]
\]

- This is not full DP because it is only done on visited states

- Can be difficult in general. If \( r \) is convex in \( u \) and \( f \) is linear in \( u \) the solution is unique and easy to find.

- If \( u \) is bounded (say by ±1) we can either clip the result or do it more smoothly with a sigmoid, \( s(u) = \frac{2}{\pi} \arctan(cu) \) (where \( c \) determines sensitivity).

- \( f \) and \( r \) can be learned on line (with function approximators) and used here instead of the “true” pair.
Pendulum Swing-Up, Limited Torque

- State = \([\theta, \omega = \frac{d\theta}{dt}]\)
- Control \(u=\) torque = \(\frac{d\omega}{dt}\)
- Model is known: \(\dot{\theta} = \omega \) and \(ml^2 \ddot{\omega} = -\mu \omega + mgl \sin \theta + u\)
- Value function approximated by a normalized Gaussian network

\[
V(x, w) = \sum_{k=1}^{K} w_k e^{-\|x-c_k\|^2/\sigma_k^2} - \sum_{k=1}^{K} e^{-\|x-c_k\|^2/\sigma_k^2}
\]

- Reward = \(\cos(\theta) - 0.1u - 0.1|\omega|\)
- Used eligibility trace (time constant \(\kappa = 0.7\))
- Model simulated by Runge Kutta 4 with \(dt = 0.07\). Learning dynamics (eligibility trace) simulated by Euler method
Figure 4: Comparison of the time course of learning with different control schemes: (A) discrete actor-critic, (B) continuous actor-critic, (C) value-gradient-based policy with an exact model, (D) value-gradient policy with a learned model (note the different scales). \( t_{up} \): time in which the pendulum stayed up. In