CS599 Reinforcement Learning and Learning Control: Probabilistic Optimization Methods

Stefan Schaal
Computer Science, Neuroscience, & Biomedical Engineering
University of Southern California, Los Angeles

&
ATR Computational Neuroscience Laboratory
Kyoto, Japan

sschaal@usc.edu
http://www-clmc.usc.edu
Reinforcement Learning
Based on Path Integrals

• Pre-requisites

System Dynamics:
\[ \dot{x} = f(x, t) + G(x)(u(t) + \varepsilon(t)) = F(x, u, t) \]

Cost Function:
\[ r_t = q(x) + \frac{1}{2} u^T R u \]

\[ J_{x_t} = E_{x_t} \left\{ q_T + \int_{t'}^{t} r_{t'} dt' \right\} \]

→ Goal: find commands \( u \) that minimizes this cost
Reinforcement Learning
Based on Path Integrals

• Sketch of the Path-Integral Derivation

System Dynamics: \( \dot{x} = f(x,t) + G(x)(u(t) + \epsilon(t)) = F(x,u,t) \)

Cost Function: \( J_x = E_x \left\{ q_T + \int_{t'=t}^T r_{t'} dt' \right\} \rightarrow \) Goal: find commands that minimizes this cost

Stochastic HJB Equations: \( -\partial_t V(x,t) = \min_{u_{x,m}} \left[ r_t + \partial_x V(x,t)^T F(x,u,t) + \frac{1}{2} Tr \left\{ \Omega(x,u,t) \partial_x^2 V(x,t) \right\} \right] \)

Log-Transformation Trick: \( \partial_t \psi(x,t) = \frac{1}{\lambda} \psi(x,t) q_t - \partial_x \psi(x,t)^T f(x,t) - \frac{1}{2} Tr \left\{ G(x) \Sigma G(x)^T \partial_x^2 \psi(x,t) \right\} \)

Application of Feynman-Kac Theorem: \( \psi(x,t) = E_{\tau} \left\{ \psi(x_T,T) \exp \left\{ -\int_{t'=t}^T \frac{1}{\lambda} q_{t'} dt' \right\} \right\} \)

Optimal Control Law: \( u_t = E_{\tau} \left\{ w_T \left[ \begin{array}{c} \epsilon_t - \lambda R^{-1} G(x)^T \end{array} \right] \right\} \)

Note: this derivation is very general for a broad class of control systems.

Kappen et al., Theodorou et al.
Reinforcement Learning Based on Path Integrals

• Important General Result

\[
\begin{align*}
\mathbf{u}_t &= \lambda \mathbf{R}^{-1} \mathbf{G}(x_t)^T \frac{\partial_x \psi(x_t, t)}{\psi(x_t, t)} \\
&= -\int_T^T \frac{\exp\left(-\frac{1}{\lambda} Z(\tau_i)\right)}{\int_T^T \exp\left(-\frac{1}{\lambda} Z(\tau_i)\right) d\tau_i} \mathbf{R}^{-1} \mathbf{G}(x_t)^T \partial_x Z(\tau) d\tau_t = -\int_T^T \mathbf{w}_t \mathbf{R}^{-1} \mathbf{G}(x_t)^T \partial_x Z(\tau) d\tau_t \\
&\approx \sum_T^T \mathbf{w}_t \mathbf{R}^{-1} \mathbf{G}(x_t)^T \partial_x Z(\tau)
\end{align*}
\]

\[
\begin{align*}
Z(\tau) &= q_T + \sum_{m=t}^{T/dt} \mathbf{q}_m dt + \frac{\lambda}{2} \sum_{m=t}^{T/dt} \left[ \mathbf{e}_m^T \mathbf{G}(x_m)^T \left( \mathbf{G}(x_m) \Sigma \mathbf{G}(x_m)^T \right)^{-1} \mathbf{G}(x_m) \mathbf{e}_m dt \right] \\
&\quad + \frac{\lambda}{2} \sum_{t=m}^{T/dt} \left[ Tr \left\{ \log (2\pi dt) + \log \left( \mathbf{G}(x_m) \Sigma \mathbf{G}(x_m)^T \right) \right\} \right] \\
\frac{\partial_x Z(\tau)}{\partial_x} &= \lambda Tr \left\{ \left( \mathbf{G}(x_t) \Sigma \mathbf{G}(x_t)^T \right)^{-1} \mathbf{G}(x_t) \Sigma \left( \mathbf{G}'(x_t)^T - \Sigma^{-1} \mathbf{e}_s s^T \right) \right\}
\end{align*}
\]

\[\lambda \mathbf{R}^{-1} = \Sigma\]
Reinforcement Learning Based on Path Integrals

For parameterized policies like dynamic motor primitives, a beautifully simple algorithm results:

1) Create K trajectories of the motor primitive for a given task with noise.
2) We can write the cost to go from every time step t of the trajectory as:

\[ R_t = q_T + \sum_{i=t}^{T} r_i \]

3) The probability of a trajectory becomes

\[ P(\xi_t^k) = \frac{\exp\left(-\frac{1}{\lambda} R_t^k \right)}{\sum_{j=1}^{K} \exp\left(-\frac{1}{\lambda} R_t^j \right)} \]

4) Update the parameter \( \theta \) of the motor primitive as

\[ \Delta \theta_t = \sum_{k=1}^{K} P(\xi_t^k) R_t^{-1} g^k(x_t) g^k(x_t)^T \mathbf{c}_t^k \]

5) Final parameter update

\[ \theta_{\text{new}} = \theta_{\text{old}} + \Delta \theta_t \]

Note that there are NO open tuning parameters except for the exploration noise.
Compare to PoWER Algorithm

- **Update Equations of PoWER**

\[
\ln J_{\theta'} = \ln \int_{T} p_{\theta'}(\tau) R(\tau) d\tau = \ln \int_{T} \frac{p_{\theta}(\tau)}{p_{\theta}(\tau)} p_{\theta'}(\tau) R(\tau) d\tau = \ln \int_{T} p_{\theta}(\tau) R(\tau) \frac{p_{\theta'}(\tau)}{p_{\theta}(\tau)} d\tau \\
\geq \int_{T} p_{\theta}(\tau) R(\tau) \ln p_{\theta'}(\tau) d\tau - \int_{T} p_{\theta}(\tau) R(\tau) \ln p_{\theta}(\tau) d\tau
\]

\[
\theta' = \theta + \left\{ E_t \left[ \sum_{t=0}^{T-1} \left( \hat{Q}(s_t, a_t) \frac{g_t g^T_t}{g_t g^T_t} \right) \right] \right\}^{-1} E_t \left[ \sum_{t=0}^{T-1} \left( \hat{Q}(s_t, a_t) \frac{g_t g^T_t}{g_t g^T_t} e_t \right) \right]
\]

\[
\hat{Q}(s_t, a_t) = \frac{1}{T} \sum_{m=t}^{T-1} r(s_m, a_m, s_{m+1}, m)
\]
Example: Results on 2D Reaching Through a Via Point
Example: Results on 20D Reaching Through a Via Point
Example: Results on 50D Reaching Through a Via Point

[Graph showing the relationship between cost and number of roll-outs, with the x-axis representing the number of roll-outs and the y-axis representing cost.]
This is a 12 DOF motor system, using 50 basis functions per primitive. Learning converges after about 20-30 trials! Performance improved by 15cm (0.5 body lengths).