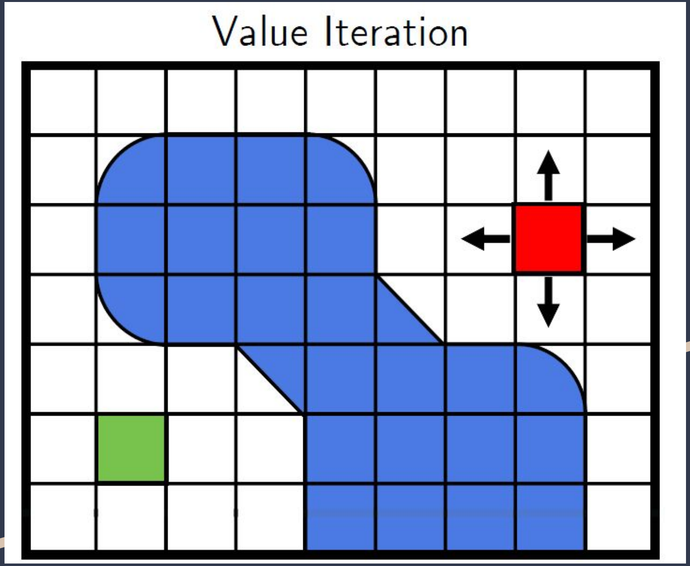
Value Iteration in Continuous Actions, States and Time

By Mayuri Praveen Shimpi and Jyotika Hariom Patil

Value Iteration (VI) discrete state, value



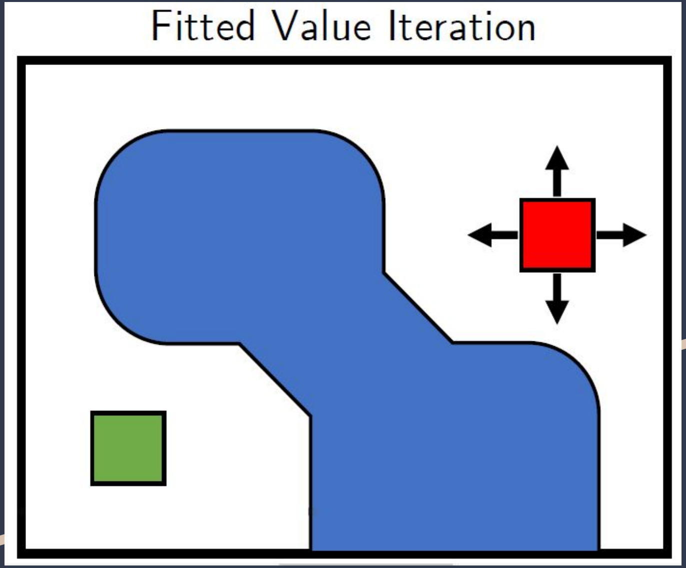
- Value iteration (VI) computes optimal value function V^* and policy π^* for discrete time, state and action environments with known rewards.

$$V^{k+1}(\boldsymbol{x}_t) = \max_{\boldsymbol{u}_{0\ldots\ell}} \sum_{i=0}^{\ell-1} \gamma^i r(\boldsymbol{x}_{t+i}, \boldsymbol{u}_i) + \gamma^\ell V^k(\boldsymbol{x}_{t+\ell}),$$

- Iteratively updates Value Function for each state using Bellman optimality equation.

- Computationally heavy for larger MDPs.

VI continuous state, discrete action



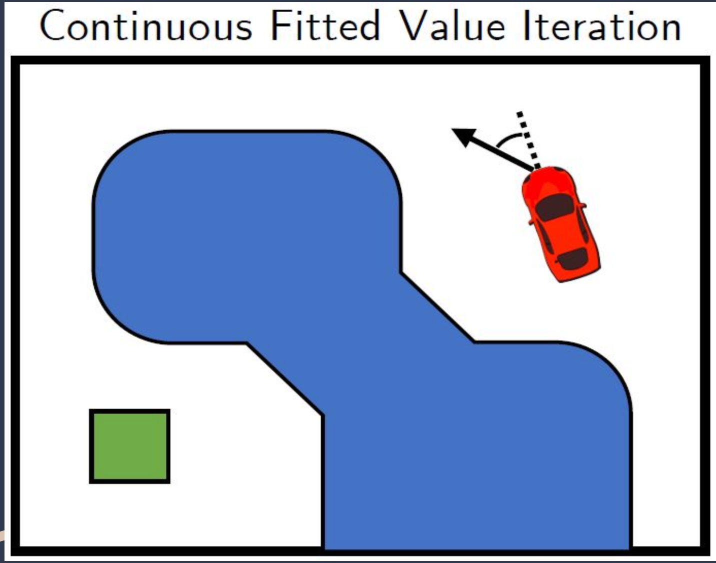
- Fitted Value Iteration (FVI) computes value function target for continuous state spaces and discrete actions by using a function approximator instead of tabular value function.

$$V_{\text{tar}}(\boldsymbol{x}_t) = \max_{\boldsymbol{u}} \; r(\boldsymbol{x}_t, \boldsymbol{u}) + \gamma V^k(f(\boldsymbol{x}_t, \boldsymbol{u});\, \psi_k)$$

$$\psi_{k+1} = \arg\min_{\psi} \sum_{\boldsymbol{x} \in \mathcal{D}} \|V_{\text{tar}}(\boldsymbol{x}) - V^k(\boldsymbol{x};\, \psi)\|_p^p$$

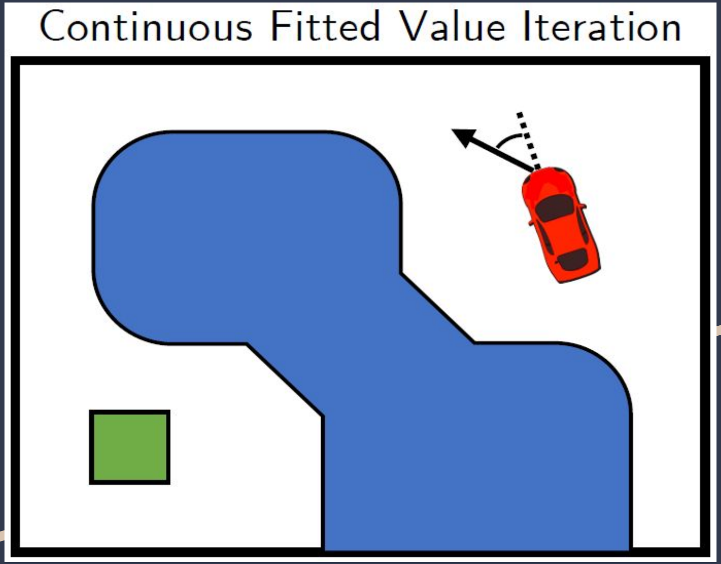
- FVI cannot be directly applied to continuous actions because of the maximization in the equation above.

VI continuous state, action



- For continuous state space and actions, popular RL method is Policy Iteration (PI).
- Maximization not required for PI, however additional optimization is required.
- In VI for continuous actions, maximization is solved analytically for continuous time.
- This solution computes efficiently.

VI continuous state, action



- For continuous state space and actions, popular RL method is Policy Iteration (PI).
- Maximization not required for PI, however additional optimization is required.
- In VI for continuous actions, maximization is solved analytically for continuous time.
- This solution computes efficiently.

  How?

Related Work

Continuous Reinforcement Learning : Continuous state space, actions and time in dynamic systems.
Pioneering work was done by Doya et. al. (2000).
The researchers used Hamilton-Jacobi-Bellman (HJB) equation for infinite horizon with discounted reward problems to derive algorithms for estimating value functions obtaining optimal policy.
Trajectory based HJB : obtain optimal trajectory
State-space based HJB: obtain optimal nonlinear controller

Classical HJB	Dynamic Programming HJB
Discretize the continuous spaces into grid with regression(using PDE solver)	Apply Bellman optimality principle to solve HJB iteratively

Dynamics

Obtain optimal policy for non-linear control affine systems

The dynamics model f_c is assumed to be nonlinear w.r.t system state, x and affine w.r.t action, u

- Nonlinear drift, a
- Nonlinear control matrix, B
- System parameters, θ

$$\dot{x} = a(x;\theta) + B(x;\theta)u,$$

The reward is separable into a non-linear state reward q_c and the action cost g_c described by

$$r_c(x,u) = q_c(x) - g_c(u).$$

- Action penalty g_c is nonlinear, positive definite and strictly convex (to have an unique optimal action).
- Action cost penalizes non-zero actions to avoid emergence of bang-bang policies (on–off).

Problem Statement

Consider Hamilton-Jacobi-equation (continuous time version of Bellman Optimality Equation)
the dynamics f_c.

- Reward : r (x, u) = $\Delta t\ r_c$(x, u)
- Discount factor for the continuous time: $\gamma = \exp(-\varrho \Delta t)$ with the sampling interval Δt
- The equations of policy and value function have integral over time.

$$\pi^*(x_0) = \arg\max_{\pi} \int_0^{\infty} \exp(-\rho t)\ r_c(x_t, u_t)\, dt$$

$$V^*(x_0) = \max_{u} \int_0^{\infty} \exp(-\rho t)\, r_c(x_t, u_t)\, dt$$

$$\text{with} \quad x(t) = x_0 + \int_0^t f_c(x_\tau, u_\tau)\, d\tau$$

Roadmap

- Classical value iteration can be extended to environments with continuous actions and states.
- Learning the value function V and deducting policy from V in the continuous-time limit. Thus, no need of the policy optimization used by the predominant actor-critic approaches or the discretization required by the classical algorithms.
- An in-depth quantitative and qualitative evaluation with comparisons to actor-critic algorithms.

Solution

Fitted value iteration (FVI) computes the value function target using the VI update and minimizes the ℓp-norm between the target and the approximation Vk

$$V_{\text{tar}}(\boldsymbol{x}_t) = \max_{\boldsymbol{u}} \; r(\boldsymbol{x}_t, \boldsymbol{u}) + \gamma V^k(f(\boldsymbol{x}_t, \boldsymbol{u}); \; \psi_k)$$

$$\psi_{k+1} = \arg\min_{\psi} \sum_{\boldsymbol{x} \in \mathcal{D}} \|V_{\text{tar}}(\boldsymbol{x}) - V^k(\boldsymbol{x}; \; \psi)\|_p^p$$

However, we cannot directly apply FVI to continuous actions due to the maximization in above equation.

To extend value iteration to continuous actions, we solve this maximization analytically for the considered continuous-time problem to get a closed form solution

Optimal Action and Value function

For the dynamics discussed, optimal policy and actions can be described as

$$\pi^k(\boldsymbol{x}) = \nabla \tilde{g}_c\left(\boldsymbol{B}(\boldsymbol{x})^T \nabla_x V^k\right)$$

$$\boldsymbol{u}^* = \underbrace{\nabla \tilde{g}_c}_{\text{Rescales the direction}}\underbrace{\left(\boldsymbol{B}^T \nabla_x V^k\right)}_{\text{Direction of steepest ascent}}$$

- g bar is the convex conjugate of g
- delta Vk is value function gradient

Substituting u* in the Vtar for FVI, we obtain

$$V_{\text{tar}}(\boldsymbol{x}_t) = r\left(\boldsymbol{x}_t, \nabla \tilde{g}\left(\boldsymbol{B}(\boldsymbol{x}_t)^T \nabla_x V^k\right)\right) + \gamma V^k(\boldsymbol{x}_{t+1};\, \psi_k)$$

$$\text{with} \quad \boldsymbol{x}_{t+1} = f\left(\boldsymbol{x}_t, \nabla \tilde{g}(\boldsymbol{B}(\boldsymbol{x}_t)^T \nabla_x V^k)\right).$$

cFVI Algorithm

Input: Dynamics Model $f_c(x, u)$ & Dataset $\mathcal{D}$
Result: Value Function $V^*(x;\, \psi^*)$

for k = 0 . . . N **do**
  // Compute Value Target for $x \in \mathcal{D}$:
  $V_{\text{tar}}(x_i) = \int_0^T \beta \, \exp(-\beta t) \, R_t \, dt + \exp(-\beta T) R_T$
  $R_t = \int_0^t \exp(-\rho\tau) \, r_c(x_\tau, u_\tau) d\tau + \exp(-\rho t) V^k(x_t)$
  $x_\tau = x_i + \int_0^\tau f_c(x_t, u_t) dt$
  $u_\tau = \nabla \tilde{g} \left(B(x_\tau) \nabla_x V^k(x_\tau) \right)$

  // Fit Value Function:
  $\psi_{k+1} = \arg\min_\psi \sum_{x \in \mathcal{D}} \| V_{\text{tar}}(x) - V(x; \psi) \|^p$

  if RTDP cFVI **then**
    // Add samples from π^{k+1} to FIFO buffer $\mathcal{D}$
    $\mathcal{D}^{k+1} = h(\mathcal{D}^k, \{x_0^{k+1} \, \dots \, x_N^{k+1}\})$
  end if
end for

Continuous time formulation of the discounted n-step value target similar to the forward eligibility trace

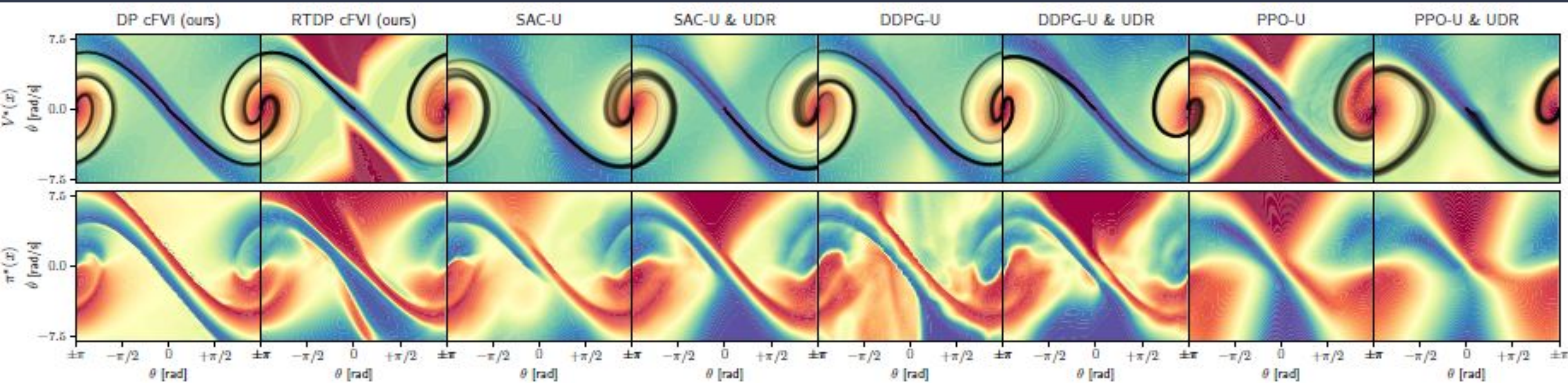
Fit value function network to the target value function

In case of Real time Dynamic Programming, add data of the current policy to the reply memory

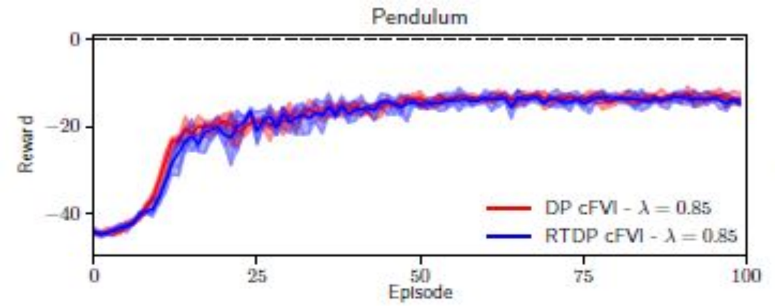
Simulation results

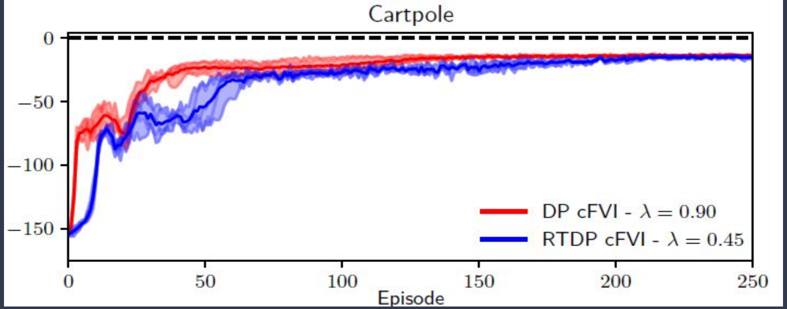
Baseline : Performance is compared to three actor-critic deep RL methods: DDPG, SAC and Proximal Policy Optimization (PPO).
Systems: Pendulum, Cartpole and Furuta Pendulum
Types: Simulated and Physical

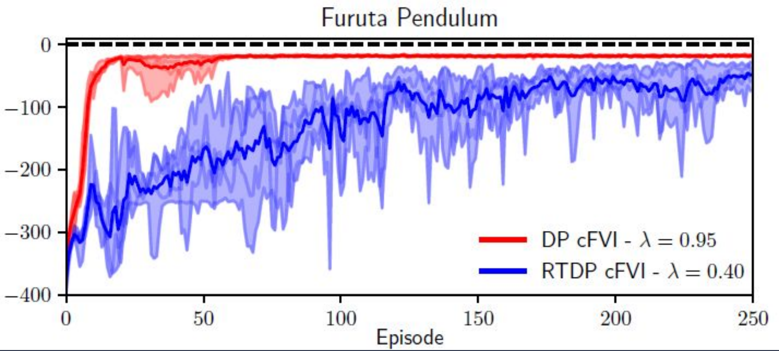
Simulation Results
Dynamic Programming cFVI (our model), with offline cFVI and a fixed dataset, achieved best rewards for all systems.
RTDP cFVI, uses the state distribution of the current policy (rather than a fixed dataset) Solves the tasks for all three systems.



- Torque Limited pendulum: Symmetric swing-up
- cFVI learns symmetric and smooth optimal policy (swing-up from both sides)
- DP cFVI has sharp ridge to the upward pointing pendulum.
- Other baselines do not achieve symmetric swing-up.
- Result : Achieves higher performance compared to most baselines.





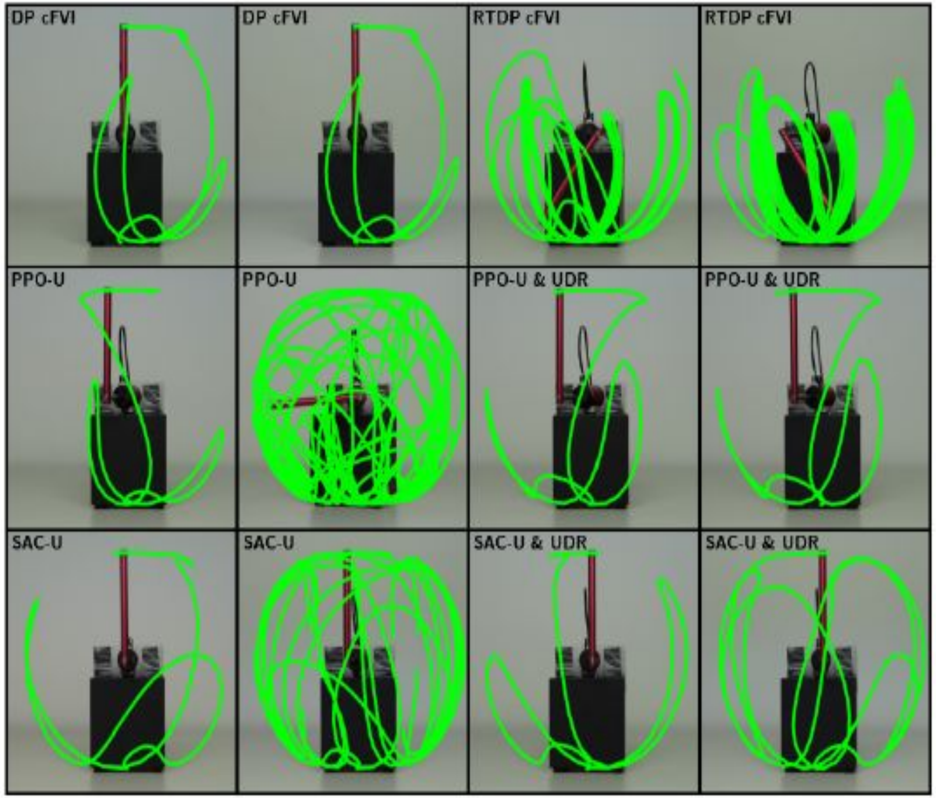


DP cFVI vs **RTDP cFVI**

- **DP cFVI** averaged over 5 seeds.
- Consistent learning of the optimal policy
- Low variance between seeds

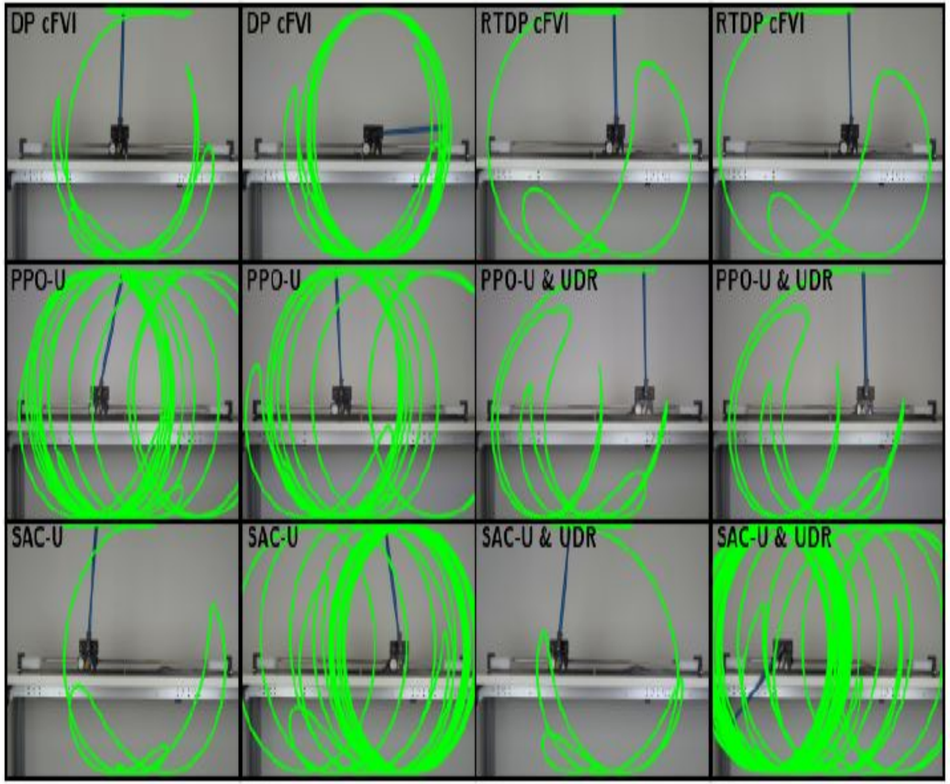
- **RTDP cFVI** averaged over 5 seeds
- Slow learning of the optimal policy
- High variance between seeds

Pendulum Experiment on Physical System



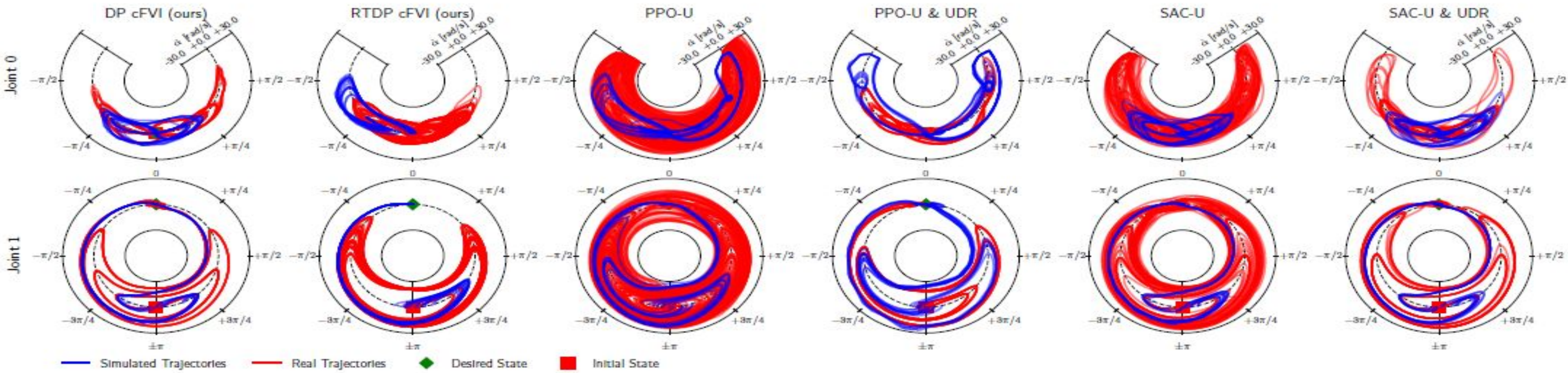
- Pendulum: Swing up for best and worst roll out.
- **DP cFVI** can consistently swing-up the pendulum.
- Variance is minimal between roll outs.
- Result : Achieves higher performance compared to most baselines.

Cartpole Experiment on Physical System



- Cartpole: Swing up for best and worst roll out.
- **DP cFVI** and **RTDP cFVI** can consistently swing-up the pendulum.
- Other deep RL baseline move cart between joints limits in case of failures.
- Result : cFVI Achieves higher performance compared to most baselines.

Furuta Pendulum Experiment on Physical System



- The simulated (blue) and real world (red) roll outs for the Furuta pendulum.
- **cFVI** achieves the best qualitative performance (swing-up the pendulum from both directions and achieves nearly identical roll-outs).
- Deep RL baselines have higher distribution shift due to the dynamics mismatch.
- cFVI gives policy robust to the changes in the dynamics.

Implementation Challenge

- State transformations should be explicitly incorporated in the value function and should not be contained in the environment, as for example in the openAI Gym. If the transformation is implicit, $\nabla_x V$ might not be sensible.

- If the state transform is explicitly incorporated in the value function, this problem does not occur. The transformation can be included explicitly by

$$V(x;\, \psi) = f(h(\boldsymbol{x});\, \psi) \qquad \nabla_x V(x; \psi) = \partial f(h(\boldsymbol{x}); \psi)/\partial h \, \partial h(\boldsymbol{x})/\partial \boldsymbol{x}.$$

- In this case, the gradient of the transformed state is projected to the tangent space of the feature transform. Therefore, the value function gradient points in a plausible direction.

Conclusion

- Extended standard value iteration to continuous states, actions and time
- Showed that optimal actions can be computed in closed form for control affine system dynamics
- Applied the resulting algorithms to continuous control problems in simulation and the physical world.
- cFVI performs much better than the standard deep RL algorithms on the physical system.

Thank You

Any Questions?

Appendix

Control Affine Systems

- A nonlinear system in which the control(s) appear(s) linearly is called a control/input-affine nonlinear system (or simply control/input-affine system, where the nonlinearity with respect to the state is automatically implied).
- A system with nonlinearities both in the state and in the control(s) is called a control/input-nonaffine nonlinear system (or simply control/input-nonaffine system).

In this paper, experiments are performed on Control Affine Systems in the continuous time.

HJB Equation for Discounted Reward

According to the optimality principle, infinite horizon integral into two parts $[t; t+\Delta t]$ and $(t+\Delta t; \infty)$ and then solve a optimization problem

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

For a small Δt, the first term is approximated $r(\mathbf{x}(t), \mathbf{u}(t))\Delta t + o(\Delta t)$

and the second term is Taylor expanded

$$V^*(\mathbf{x}(t+\Delta t)) = V^*(\mathbf{x}(t)) + \frac{\partial V^*}{\partial \mathbf{x}(t)} f(\mathbf{x}(t), \mathbf{u}(t))\Delta t + o(\Delta t)$$

HJB Equation for Discounted Reward

By substituting them into 1st equation and collecting V (x(t)) on the left-hand side, we have an optimality condition for [t; t + Δt] as

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

By dividing both sides by Δt and taking Δt to zero, we have the condition for the optimal value function

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