Reinforcement Learning for Autonomous Quadrotor Helicopter Control

Michael Koval, Christopher Mansley, and Michael Littman

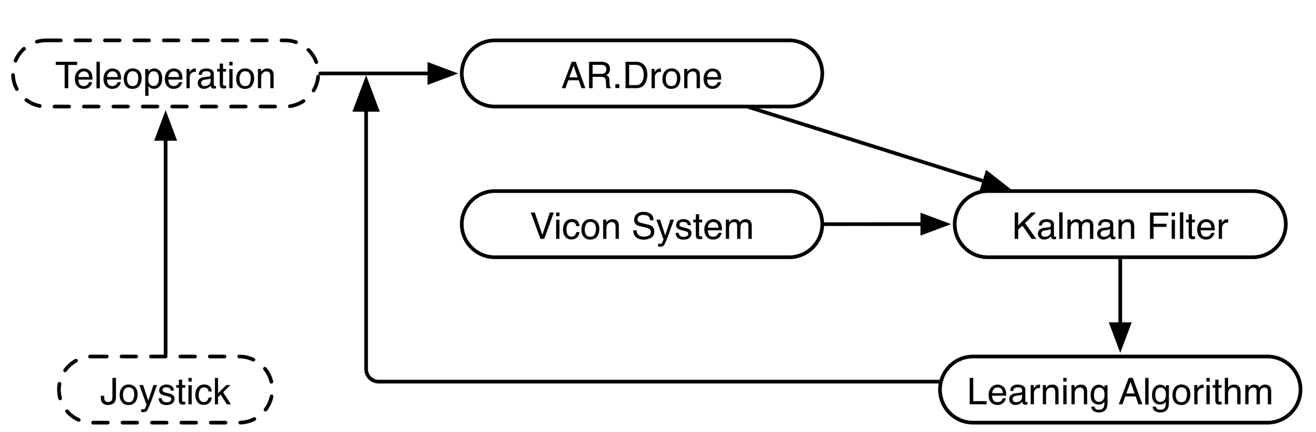
Rutgers, the State University of New Jersey

Abstract

- Quadrotor helicopters are rapidly finding use for a number of applications
- Programming a new behavior means implementing a control algorithm
- Tuning a new algorithms is costly and requires expertise in control theory
- Supervised learning is not a viable alternative: training data is hard to get
- Reinforcement learning can learn directly from the quadrotor's raw sensors
- **Goal:** Use reinforcement learning to teach a quadrotor helicopter how to perform complex tasks with minimal human interaction

Experimental Setup

1. **Parrot AR.Drone** [4]
   - Inexpensive quadrotor helicopter with impressive technical specifications
   - Outfitted with two cameras, an altitude sensor, and an IMU
   - On-board computer running a variant of embedded Linux
   - Communicates with a controlling computer over a dedicated wifi network
2. **Vicon Motion Tracking System**
   - Quadrotor is tagged with an asymmetric pattern of reflective spheres
   - Configure the Vicon system to track the pattern as a rigid body
   - Track the quadrotor's flight using the Vicon system's infrared cameras
3. **Control Software**
   - Exerts direct, low-level control over the AR.Drone's flight
   - Closes the loop between the motion tracking system and the AR.Drone
   - Uses reinforcement learning to find an optimal control algorithm
4. **Robot Operating System (ROS)**
   - Inter-process communication framework developed by Willow Garage
   - Allows the system to be easily extended for more complex experiments



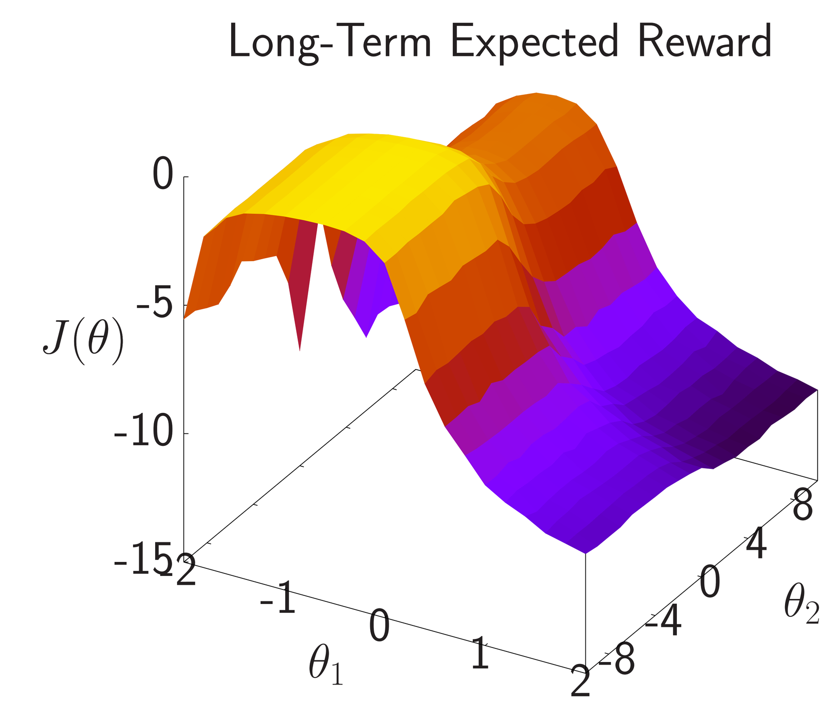
Reinforcement Learning

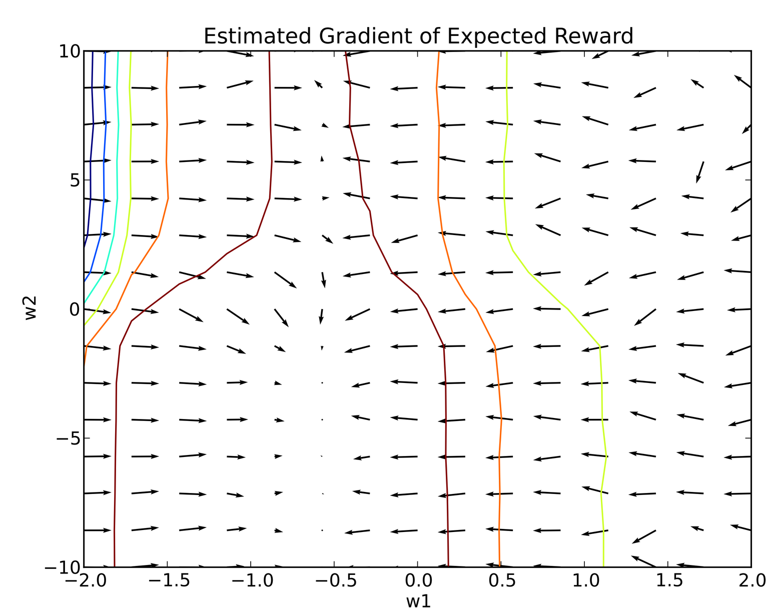
- Agent learns from rewards earned while interacting with a stochastic environment
- Does not require annotated examples of the agent's correct actions
- Ideal for situations rewards can automatically be assigned
- Environment is modeled as a Markov decision process where, at time t:
  1. the agent is in state $s_t \in S$,
  2. chooses action $a_t \sim \pi(s_t, a)$ using policy π, and
  3. receives reward r_t
- Goal is to find the policy, π^*, that maximizes the agent's reward:

$$J|_\pi = E\left\{\sum_{t=0}^{\infty} \gamma^t r_t\right\}$$

- Popular algorithms include: Q Learning, TD Learning, and Actor-Critic
- Policy gradient and natural actor critic are most popular in robotics

Simulation: Linear Quadratic Regulator



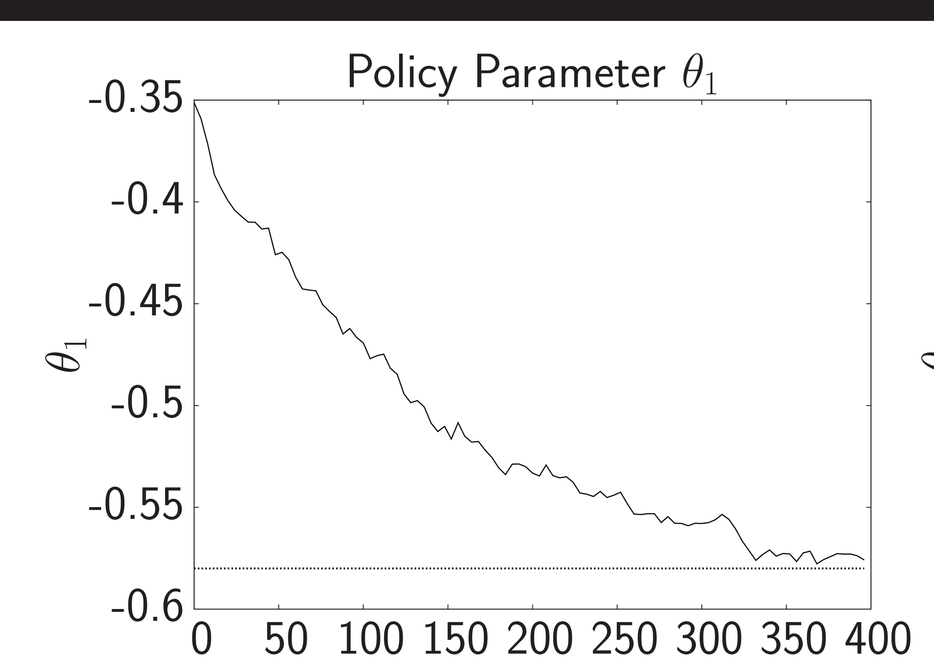


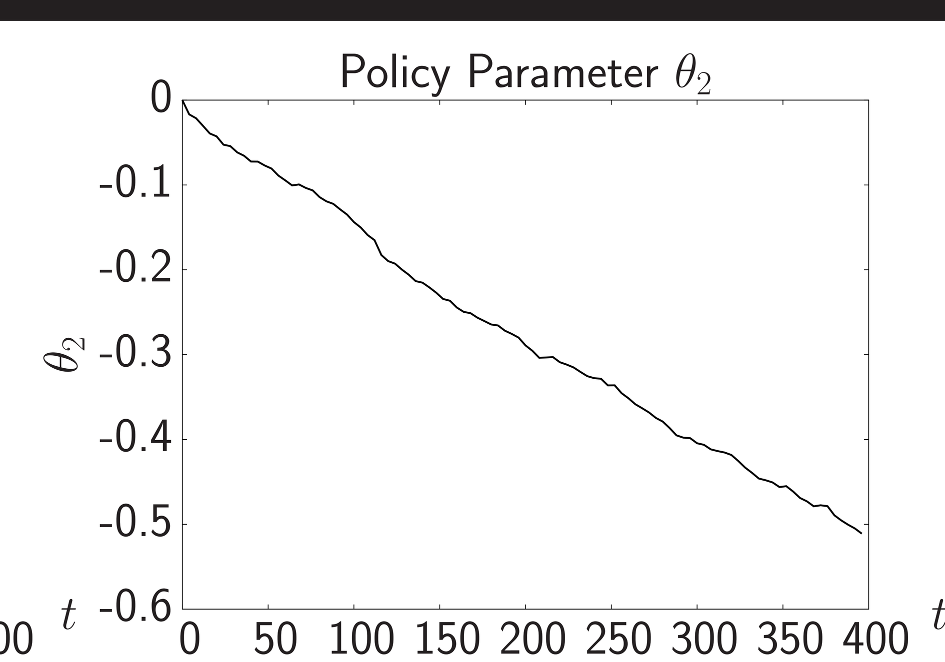
- Simple one-dimensional control theory problem with $s \in \mathbb{R}$, $a \in \mathbb{R}$, and the environment: [2]

$$s_{t+1} = s_t + a_t + \text{noise}$$
$$R_t = -s_t^2 - a_t^2$$

- Stochastic controller draws actions from $a_t \sim N(\mu_\theta, \sigma_\theta)$ where $\mu_\theta = \theta_1 s_t$ and $\sigma_\theta = \left[1 + e^{-\theta_2}\right]^{-1}$ [2]
- Vanilla policy gradient was simulated using central-difference estimator to compute gradients [3]

Results: Linear Quadratic Regulator Simulation





- Control theory proves that the theoretical optimum is at $\theta_1^* = -0.58$ and $\theta_2^* = -\infty$ [2]
- Vanilla gradient ascent quickly converges to $\theta_1 = \theta_1^*$ and continues moving towards $\theta_2 = \theta_2^*$

Policy Gradient

- Assumes the policy $\pi(s, a \mid \theta)$ is uniquely parameterized by θ
- Learning the optimal policy π^* is now simplified to learning the optimal θ^*
- For any given θ, $\nabla J|_\theta$ points in the direction of maximum increase
- Step the parameters by a small amount in the direction of $\nabla J|_{\theta_i}$:

$$\theta_{i+1} = \theta_i + \alpha \nabla J|_{\theta_i}$$

- Gradually decrease α until the algorithm converges to a local maximum
- **Problem:** This requires an empirical estimate of the gradient $\nabla J|_\theta$

Finite Difference Gradient Estimate

- Simple form of gradient approximation that requires no knowledge of π
- Slightly perturb θ with several small changes (i.e. $\Delta\theta_1$, $\Delta\theta_2$, ...)
- Approximate $J(\theta + \Delta\theta_i)$ for each $\Delta\theta_i$ by averaging multiple roll-outs [3]
- Construct $\Delta\Theta$ and ΔJ such that $\Delta\Theta_i = \theta_i$, $\Delta J_i = J(\theta + \Delta\theta_i)$, and

$$\nabla J|_\theta = \left(\Delta\Theta\Delta\Theta^T\right)^{-1} \Delta\Theta^T \Delta J$$

Likelihood Ratio Gradient Estimate

- Requires specific knowledge of the policy to estimate $\nabla\pi(s, a|\theta)$ [2]
- Estimates $\nabla J|_\theta$ without requiring the policy to be perturbed:

$$\nabla J|_\theta = E\left\{\sum_{t=0}^{\infty} (r_t - b) \nabla \log \pi(s_t, a_t|\theta)\right\}$$

- Faster convergence than finite difference methods in noisy environments

Natural Policy Gradient

- Gradient is defined in terms of Euclidean distance:

$$\lim_{\Delta\theta \to 0} \left(\frac{||J(\theta + \Delta\theta) - J(\theta) + \nabla J(\theta) \cdot \Delta\theta||_2}{||\Delta\theta||_2}\right) = 0$$

- Parameters have arbitrary units, making the space highly non-Euclidean
- Vanilla gradient no longer points in the direction of maximum increase [3]
- Natural gradient is rotated to face the direction of maximum increase: [3]

$$\tilde{\nabla} J|_\theta = G_\theta^{-1} \nabla J|_\theta$$

- Dramatically better convergence rates than vanilla policy gradient [1]

References

S.I. Amari.
Natural gradient works efficiently in learning.
Neural computation, 10(2):251–276, 1998.

H. Kimura and S. Kobayashi.
Reinforcement learning for continuous action using stochastic gradient ascent.
Intelligent Autonomous Systems (IAS-5), pages 288–295, 1998.

J. Peters and S. Schaal.
Policy gradient methods for robotics.
In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2219–2225. IEEE, 2006.

Stephane Piskorski.
AR.Drone Developers Guide SDK 1.5.
Parrot, 2010.

mkoval@eden.rutgers.edu, cmansley@cs.rutgers.edu, mlittman@cs.rutgers.edu