Learn to Throw a Ball

Sven, Luis, Heiko

July 13, 2016

Features and Controller

Robot tries to hit a **target**, specified by an *Alvar-marker*.

- Cotroller:

$$\mu(W, t, s) = W\phi_t(s)$$

- Features:

$$\phi_t(s) = \left(t, \quad \sin\left(\frac{t\pi}{T}\right), \quad x_{eff}^{\mathrm{rot}}, \quad 1\right)^\top$$

- Robot keeps on sending *velocities* until **either**:
  - $T - 15$ commands have been send, **or**
  - x-rot. of the gripper, x_{eff}^{rot}, is above a threshold (0.005)
- Distance between ball and target is measured placing a second Alvar-marker where the ball landed.

Algorithm: Greedy Policy Search

Init W randomly with $w_{ij} \sim \mathcal{U}(0,1)$, $r_{max} \leftarrow -10.000$
while not converged **do**
    $i \leftarrow rand(1,2,3)$ ▷ Use pr. sweeping
    Add noise $\varepsilon \sim \mathcal{N}(0,\sigma)$ to w_i
    for $t \in T_i$ **do**
        Get velocities for each joint: $v_t \leftarrow \mu(W, \phi(s))$
        Apply current velocity vector v_t
    end for
    Get reward $r = -sq\_dist(target, \hat{x})$ ▷ $\hat{x}$: measured position
    if $r > r_{max}$ **then**
        $W \leftarrow W + \varepsilon$
        Reset σ
    else
        $\sigma \leftarrow \sigma \cdot 1.1$
    end if
end while

Offline-learning

During execution: collect *Data* $\mathcal{D} = \left\{W_{11}^i, \ldots, W_{34}^i, x_i, y_i, R_i\right\}_{i=1}^N$.

Idea 1: Extract relevant data points arround point of interest (kNN)
Use *Linear Regression* on the new data set to estimate *gradient*.

Idea 2: Use *supervised learning* to estimate *model* (on either $W \mapsto R$ or $W \mapsto (x, y)$).