"MODULARITY, POLYRHYTHMS, AND WHAT ROBOTICS AND CONTROL MAY YET LEARN FROM THE BRAIN"

Jean-Jacques Slotine, Nonlinear Systems Laboratory, MIT

Thursday, Nov 5th, 4:00 p.m., 3110 Etcheverry Hall

ABSTRACT

Although neurons as computational elements are 7 orders of magnitude slower than their artificial counterparts, the primate brain grossly outperforms robotic algorithms in all but the most structured tasks. Parallelism alone is a poor explanation, and much recent functional modelling of the central nervous system focuses on its modular, heavily feedback-based computational architecture, the result of accumulation of subsystems throughout evolution. We discuss this architecture from a global functionality point of view, and show why evolution is likely to favor certain types of aggregate stability. We then study synchronization as a model of computations at different scales in the brain, such as pattern matching, restoration, priming, temporal binding of sensory data, and mirror neuron response. We derive a simple condition for a general dynamical system to globally converge to a regime where diverse groups of fully synchronized elements coexist, and show accordingly how patterns can be transiently selected and controlled by a very small number of inputs or connections. We also quantify how synchronization mechanisms can protect general nonlinear systems from noise. Applications to some classical questions in robotics, control, and systems neuroscience are discussed.

The development makes extensive use of nonlinear contraction theory, a comparatively recent analysis tool whose main features will be briefly reviewed.

CS 287: Advanced Robotics
Fall 2009

Lecture 19:
Actor-Critic/Policy gradient for learning to walk in 20 minutes
Natural gradient

Pieter Abbeel
UC Berkeley EECS

Case study: learning bipedal walking

- **Dynamic gait:**
  - A bipedal walking gait is considered dynamic if the ground projection of the center of mass leaves the convex hull of the ground contact points during some portion of the walking cycle.
- **Why hard?**
  - Achieving stable dynamic walking on a bipedal robot is a difficult control problem because bipeds can only control the trajectory of their center of mass through the unilateral, intermittent, uncertain force contacts with the ground.
- ←→ "fully actuated walking"

Passive dynamic walkers

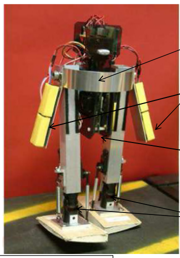
Passive dynamic walkers

- The energy lost due to friction and collisions when the swing leg returns to the ground are balanced by the gradual conversion of potential energy into kinetic energy as the walker moves down the slope.
- Can we actuate them to have them walk on flat terrains?

- John E. Wilson. Walking toy. Technical report, United States Patent Office, October 15 1936.
- Tad McGeer. Passive dynamic walking. *International Journal of Robotics Research, 9(2):62.82, April 1990.*

Learning to walk in 20 minutes --- Tedrake, Zhang, Seung 2005

Learning to walk in 20 minutes --- Tedrake, Zhang, Seung 2005



44 cm

passive hip joint [1DOF]

Arms: coupled to the opposite leg to reduce yaw moment

freely swinging load [1DOF]

2 x 2 (roll, pitch) position controlled servo motors [4 DOF]

9DOFs:
* 6 internal DOFs
* 3 DOFs for the robot's orientation (always assumed in contact with ground at a single point, absolute (x,y) ignored)

Natural gait down 0.03 radians ramp: 0.8Hz, 6.5cm steps

Dynamics

$$\ddot{q} = f(q, \dot{q}, u, d(t))$$

- q: vector of joint angles
- u: control vector (4D)
- d(t): time-varying vector of random disturbances

- Discrete footstep-to-footstep dynamics: consider state at touchdown of robot's left leg

$$F_\pi(x', x) = P(\hat{x}_{n+1} = x' | \hat{x}_n = x; \pi)$$

- Stochasticity due to
  - Sensor noise
  - Disturbances d(t)

Reinforcement learning formulation

- **Goal:** stabilize the limit cycle trajectory that the passive robot follows when walking down the ramp, making it invariant to slope.
- **Reward function:**

$$R(x(n)) = -\frac{1}{2}\|x(n) - x^*\|_2^2$$

  - x* is taken from the gait of the walker down a slope of 0.03 radians
- **Action space:**
  - At the beginning of each step cycle (=when a foot touches down) we choose an action in the discrete time RL formulation
  - Our action choice is a feedback control policy to be deployed during the step, in this particular example it is a column vector w
  - Choosing this action means that throughout the following step cycle, the following continous-time feedback controls will be exerted:

$$u(t) = \sum_i w_i \phi_i(\hat{x}(t)) = w^\top \phi(\hat{x}(t))$$

→ Goal: find the (constant) action choice w which maximizes expected sum of rewards

Policy class

- To apply the likelihood gradient ratio method, we need to define a stochastic policy class. A natural choice is to choose our action vector w to be sampled from a Gaussian:

$$w \sim \mathcal{N}(\theta, \sigma^2 I)$$

Which gives us:

$$\pi_\theta(w|x) = \frac{1}{(2\pi)^d \sigma^d} \exp\left(\frac{-1}{2\sigma^2}(w-\theta)^\top (w-\theta)\right)$$

[Note: it does not depend on x, this is the case b/c the actions we consider are feedback policies themselves!]

- The policy optimization becomes optimizing the mean of this Gaussian. [In other papers people have also included the optimization of the variance parameter.]

Policy update

Likelihood ratio based gradient estimate from a single trace of H footsteps:

$$\hat{g} = \sum_{n=0}^{H-1} \nabla_\theta \log \pi_\theta(w(n)|\hat{x}(n)) \left(\sum_{k=n}^{H-1} R(\hat{x}(k)) - b\right)$$

We have:

$$\nabla_\theta \log \pi_\theta(w|\hat{x}) = \frac{1}{2\sigma^2}(w - \theta)$$

Rather than waiting till horizon H is reached, we can perform the updates online as follows: (here η_θ is a step-size parameter, $b(n)$ is the amount of baseline we allocate to time n—see next slide)

$$\begin{aligned} e(n) &= e(n-1) + \frac{1}{2\sigma^2}(w(n) - \theta(n)) \\ \theta(n+1) &= \theta(n) + \eta_\theta e(n)(R(\hat{x}(n)) - b(n)) \end{aligned}$$

To reduce variance, can discount the eligibilities:

$$e(n) = \gamma e(n-1) + \frac{1}{2\sigma^2}(w(n) - \theta(n))$$

Choosing the baseline b(n)

A good choice for the baseline is such that it corresponds to an estimate of the expected reward we should have obtained under the current policy.

Assuming we have estimates of the value function $\hat{V}$ under the current policy, we can estimate such a baseline as follows:

$$b(n) = \hat{V}(\hat{x}(n)) - \gamma \hat{V}(\hat{x}(n+1))$$

To estimate $\hat{V}$ we can use TD(0) with function approximation. Using linear value function approximation, we have:

$$\hat{V}(\hat{x}) = \sum_i v_i \psi_i(\hat{x}).$$

This gives us the following update equations to learn $\hat{V}$ with TD(0):

$$\begin{aligned} \delta(n) &= R(\hat{x}(n)) + \gamma \hat{V}(\hat{x}(n+1)) - \hat{V}(\hat{x}(n)) \\ v(n+1) &= v(n) + \eta_v \delta(n) \psi(\hat{x}(n)) \end{aligned}$$

The complete actor critic learning algorithm

Before each foot step, sample the feedback control policy parameters $w(n)$ from $\mathcal{N}(\theta(n), \sigma^2 I)$.

During the foot step, execute the following controls in continuous time: $u(t) = w(n)^\top \phi(\hat{x}(t))$.

After the foot step is completed, compute the reward function $R(\hat{x}(n))$ and perform the following updates:

Policy updates:

$$
\begin{aligned}
e(n) &= \gamma e(n-1) + \frac{1}{2\sigma^2}(w - \theta(n)) \\
\theta(n+1) &= \theta(n) + \eta_\theta e(n)(R(\hat{x}(n)) - b(n)) \\
b(n) &= \hat{V}(\hat{x}(n)) - \gamma \hat{V}(\hat{x}(n+1))
\end{aligned}
$$

TD(0) updates:

$$
\begin{aligned}
\delta(n) &= R(\hat{x}(n)) + \gamma \hat{V}(\hat{x}(n+1)) - \hat{V}(\hat{x}(n)) \\
v(n+1) &= v(n) + \eta_v \delta(n) \psi(\hat{x}(n))
\end{aligned}
$$

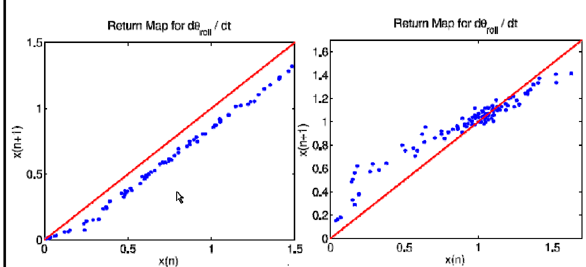
Manual dimensionality reduction

- Decompose the control problem in the frontal and sagittal planes
- Due to simplicity of sagittal plane control---hand set.
- Left with control of the ankle roll actuators to control in the frontal plane
  - Let roll control input only depend on θ_{roll} and $d\theta_{roll}/dt$
  - Basis functions: non-overlapping tile encoding
    - Policy: 35 tiles (5 in θ_{roll} x 7 in $d\theta_{roll}/dt$)
    - Value: 11 tiles (a function in $d\theta_{roll}/dt$ only because the value is evaluated at the discrete time when θ_{roll} hits a particular value)

Experimental setup and results

- When the learning begins, the policy parameters, w, are set to 0 and the baseline parameters, v, are initialized so that \hat{V}(x) ≈ R(x) / (1-γ)
- Train the robot on flat terrain.
- Reset with simple hand-designed controller that gets it into a random initial state every 10s.
- Results:
  - After 1 minute: foot clearance on every step
  - After 20 minutes: converged to a robust gait (=960 steps at 0.8Hz)

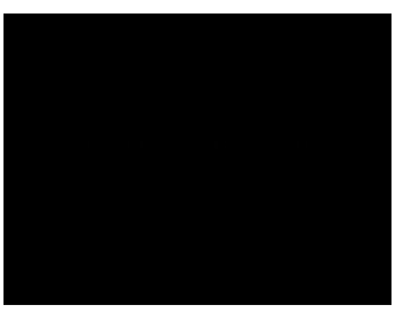
Return maps



before learning after learning

[Note: this is a projection from 2x9-1 dim to 1dim]

Toddler movie



- On tread-mill: passive walking. On ground: learning.