Interactive Control of Diverse Complex Characters with Neural Network

Igor Mordatch, Kendall Lowrey, Galen Andrew,
Zoran Popovic, Emanuel V. Todorov

University of Washington

Paper presentation for robotics class
Jules Kozolinsky
23 January 2017

Context

Context of the article

- Title : *Interactive Control of Diverse Complex Characters with Neural Network*
- Authors : Igor Mordatch, Kendall Lowrey, Galen Andrew, Zoran Popovic, Emanuel V. Todorov

  from the Department of Computer Science, University of Washington
- published in NIPS 2015 (Neural Information Processing Systems)

Goal of the article

Interactive real-time controllers (generating complex, stable and realistic movements) have many potential applications.

State of the art of controllers designing

- time-consuming
- largely manual
- relying on motion capture datasets

Goal

Automate this process, *i.e.* find a policy

- universal methods applicable to arbitrary behaviors or body morphologies
- online changes in task objectives
- perturbations due to noise and modelling errors

Goal of the article

Interactive real-time controllers (generating complex, stable and realistic movements) have many potential applications.

State of the art of controllers designing

- time-consuming
- largely manual
- relying on motion capture datasets

Goal

Automate this process, *i.e.* find a policy

- universal methods applicable to arbitrary behaviors or body morphologies
- online changes in task objectives
- perturbations due to noise and modelling errors

How?

Trajectory optimization

- Contact-Invariant-Optimization [4]
- Offline method to find a optimal trajectory

Deep learning

- supervised learning using neural network
- normally used in speech-recognition or computer vision

Method : neural network learn from the optimizer and generate similar behaviour online

Outline

Definitions

State of character and trajectory

- State of character : $(q\, f\, r)$

  q physical pose (position, orientation, joint angles)
  f contact forces
  r recurrent memory

- Trajectory of length T : $(q^0\, f^0\, r^0\, ...\, q^T\, f^T\, r^T)$

Definitions

Neural network policy

$$\pi_\theta : s \mapsto a$$

where

- θ neural network weight
- $s^t(X) = (q^t \; r^t \; \dot{q}^{t-1} \; f^{t-1})$ sensory state a time t of trajectory X
- $a^t(X) = (\dot{q}^t \; \dot{r}^t \; f^t)$ optimal action a time t of trajectory X
  $(\dot{x}^t = x^{t+1} - x^t)$

Note : The neural network learns both optimal control and model of dynamics

Problem

Offline procedure

$$\min_{\theta, X^1, \dots, X^N} \sum_{i=1}^{N} C_i(X^i) \qquad \text{subject to} \quad \forall i, t : a^t(X^i) = \pi_\theta(s^t(X^i)) \tag{1}$$

where

- θ policy parameters ($\pi_\theta : s \mapsto a$)
- X^i trajectory of task i (different initial conditions)

Note : Then optimized policy parameter θ^* is used to execute policy in real-time

Noise into the sensory inputs

Why add noise?

- Produce more robust movement stategies [3] [5]
- Reduce overfitting [1]
- Stabilize behaviour of neural network [2]

Main reason : Learning policy does not diverge at execution time

Noise into the sensory inputs

How to add noise?

Add gaussian noise into inputs s given to the neural network :

$$\pi_\theta(s+\varepsilon) = a + a_s\varepsilon$$

where

- $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2 I) \qquad (\sigma_\varepsilon^2 \simeq 10^{-2})$
- a_s gradient in first order expansion : matrix of optimal feedback (Section 3 for analytic calculation)

Note : Policy automatically corrects small deviations from the optimal trajectory

Algorithm

Optimization under constraints

Problem (1) non-convex : we replace the hard equality constraint

$$\min_{\theta, X^1, \ldots, X^N} \sum_{i=1}^{N} C_i(X^i) + \sum_{i,t} R(s^t(X^i), a^t(X^i), \theta, \varepsilon^{i,t})$$

$$\text{with} \quad R(s, a, \theta, \epsilon) = \frac{\alpha}{2} ||(a + a_s \varepsilon) - \pi_\theta(s + \epsilon)||^2$$

where

- θ policy parameters ($\pi_\theta : s \mapsto a$)
- X^i trajectory of task i
- α weight on quadratic penalty ($\alpha \simeq 10$)

Algorithm

1. Sample noise $\varepsilon^{i,t}$
2. Optimize N trajectories :

$$\forall i,\ \bar{X}^i = \text{argmin}_X\ C_i(X) + \sum_t R(s^t(X), a^t(X), \bar{\theta}, \varepsilon^{i,t}) + \frac{\eta}{2}||X - \bar{X}^i||^2$$

3. Train neural network :

$$\bar{\theta} = \text{argmin}_\theta\ \sum_{i,t} R(s^t(\bar{X}_i), a^t(\bar{X}_i), \theta, \bar{\varepsilon}^{i,t}) + \frac{\eta}{2}||\theta - \bar{\theta}||^2$$

4. Repeat

$(\eta \simeq 10^{-2})$

Outline

Trajectory Optimization

Step 2 of the algorithm

$$\bar{X} = \text{argmin}_X \; C_i(X) + \sum_t R(s^t(X), a^t(X), \bar{\theta}, \bar{\varepsilon}^{i,t}) + \frac{\eta}{2}||X - \bar{X}||^2$$

Find trajectories that start with partiular inital conditions and execute the task, while satisfying physical realism of character's motion.

Trajectory Optimization

Physical realism

$$H(q)\ddot{q} + \hat{C}(q,\dot{q}) = \tau + J(q,\dot{q})^\top f, \qquad d(q) \geq 0, \quad d(q)^\top f = 0, \quad f \in K(q)$$

where

- f contact forces acting on all end-effectors
- τ inverse of the dynamics
- H the inertia matrix
- $\hat{C}$ the matrix of Coriolis and centrifugal terms
- J Jacobian matrix
- $d(q)$ distance of the contact to the ground
- $K(q)$ contact friction cone

These constraints and initial conditions are implemented as soft constraints and included in $C(X)$

Optimal trajectory

Total cost

$$C(X) = \sum_t c^t(\phi^t(X))$$

where

- $\phi^t(X)$ extracts features from trajectory
- $c(\phi)$ cost over these features

Optimization problem

For simplicity, objectives folded into C :

$$X^* = \text{argmin}_X C(X)$$

Newton's method

Gaussian-Newton Hessian approximation

$$\nabla_X C(X) = \sum_t c_\phi^t \phi_X^t$$

$$D_X^2 C(X) = \sum_t (\phi_X^t)^\top c_{\phi\phi}^t \phi_X^t + c_\phi^t \phi_{XX}^t \approx \sum_t (\phi_X^t)^\top c_{\phi\phi}^t \phi_X^t$$

where

- $c(\phi)$ cost functions (gradient and hessian analytically calculable)
- ϕ_X^t calculated by finite differencing

Newton's method

$$X^* = X^* - D_X^2 C(X)^{-1} \nabla_X C(X)$$

Note : Not run to convergence, only between 1 and 10 iterations

Optimal feedback gains

How to add noise? (Reminder)

Add gaussian noise into inputs s given to the neural network :

$$\pi_\theta(s+\varepsilon) = a + a_s\varepsilon$$

where

- $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2 I) \qquad (\sigma_\varepsilon^2 \simeq 10^{-2})$
- a_s gradient in first order expansion : matrix of optimal feedback

Perturbation of optimal trajectory

Let $\tilde{X}$ be the perturbation of optimal trajectory X such that $s(\tilde{X}) = \bar{s}$.
Then

$$a_{\bar{s}} = a_X \tilde{X}_{\bar{s}}$$

Optimal feedback gains $a_{\bar{s}}$

Calculating a_X

a, s and ϕ are functions that extract features overs X.
So s_X and a_X are subsets of ϕ_X.

Calculating $\tilde{X}_{\bar{s}}$

We have (with $\lambda \simeq 10^2$),

$$\tilde{X}(\bar{s}) = \text{argmin}_{X^*}\left(C(X^*) + \frac{\lambda}{2}||s(X^*) - \bar{s}||^2\right)$$

Then,

$$\tilde{X}(\bar{s}) = X - (C_{XX} + \lambda s_X^\top s_X)^{-1}(C_X + \lambda s_X^\top (s(X) - \bar{s})$$

And,

$$\tilde{X}_{\bar{s}} = \lambda(C_{XX} + \lambda s_X^\top s_X)^{-1} s_X^\top$$

Outline

Neural network training

Step 3 of the algorithm

$$\bar{\theta} = \text{argmin}_\theta \sum_{i,t} R(s^t(\bar{X}_i), a^t(\bar{X}_i), \theta, \bar{\varepsilon}^{i,t}) + \frac{\eta}{2}||\theta - \bar{\theta}||^2$$

with

$$R(s, a, \theta, \epsilon) = \frac{\alpha}{2}||(a + a_s\varepsilon) - \pi_\theta(s + \epsilon)||^2$$

Neural network policy regression

Training data : $(s + \varepsilon, a + a_s\varepsilon)^{i,t}$

Output function : θ^* weight of neural network used in policy $\pi_{\theta^*} : s \mapsto a$

Neural network training

A difficult problem

- data set harder to obtain (// computer vision)
- regression (real valued outputs) $\neq$ classification (categorical outputs)
- no i.i.d. values

Neural network training

Parameters of the neural network

- hidden layer activation function $\sigma = tanh$
- 3 hidden layers with 250 hidden units in each layer
- $\gamma \sim \mathcal{N}(0, \sigma_\gamma^2 I)$ noise at each layer during training ($\sigma_\gamma \simeq 10^{-2}$)

10 neurons	0.337 ± 0.06
25 neurons	0.309 ± 0.06
100 neurons	0.186 ± 0.02
250 neurons	0.153 ± 0.02
500 neurons	0.148 ± 0.02

(a) Increasing Neurons per layer with 4 layers

1 layer	0.307 ± 0.06
2 layers	0.253 ± 0.06
3 layers	0.153 ± 0.02
4 layers	0.158 ± 0.02

(b) Increasing Layers with 250 neurons per layer

Table 1: Mean and variance of joint position error on test rollouts with our method after training with different neural network configurations.

Note : Not run to convergence

Outline

Policy execution

Real-time execution

We found policy parameters θ^* offline.
Let x^0 be the initial state. Then desired action : $a^{des} = \pi_{\theta*}(s(x^0))$. So,

$$x^1 = \text{argmin}_x ||\dot{q} - \dot{q}^{des}||^2 + ||\dot{r} - \dot{r}^{des}||^2 + ||f - f^{des}||^2 \text{ subject to dynamics}$$

(same trajectory optimization problem with horizon $T = 1$)

Outline

Results

Set-up

- $\Delta t = 50$ ms
- takes about 2.5 hours
- MuJoCo physics simulator

Experiments

- swimming
- flying
- biped walk
- quadruped walk

Video : `https://www.youtube.com/watch?v=IxrnT0JOs4o`

Conclusion and future work

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov.
Improving neural networks by preventing co-adaptation of feature detectors.
arXiv preprint arXiv:1207.0580, 2012.

Gregor M Hoerzer, Robert Legenstein, and Wolfgang Maass.
Emergence of complex computational structures from chaotic neural networks through reward-modulated hebbian learning.
Cerebral cortex, 24(3):677–690, 2014.

Dongsung Huh and Emanuel Todorov.
Real-time motor control using recurrent neural networks.
In *Adaptive Dynamic Programming and Reinforcement Learning, 2009. ADPRL'09. IEEE Symposium on*, pages 42–49. IEEE, 2009.

Igor Mordatch, Emanuel Todorov, and Zoran Popović.
Discovery of complex behaviors through contact-invariant optimization.
ACM Transactions on Graphics (TOG), 31(4):43, 2012.

Jack M Wang, David J Fleet, and Aaron Hertzmann.
Optimizing walking controllers for uncertain inputs and environments.
In *ACM Transactions on Graphics (TOG)*, volume 29, page 73. ACM, 2010.