Learning Complex Neural Network Policies with Trajectory Optimization

Sergey Levine - Stanford University

Vladlen Koltun - Adobe Research

1. Introduction

Goal: Learn **complex** control strategies for high-dimensional continuous systems (e.g. robots) using **expressive** nonlinear function approximators, avoiding task-specific feature engineering.

Challenge: Policy search methods often require carefully designed, low-dimensional policy classes to avoid disastrous local optima and discover successful task executions. It is very difficult to simultaneously solve complex continuous control problems and optimize a high-dimensional, nonlinear function approximator.

Supervised learning is relatively easy, even for a complex, nonlinear function approximator, but we need a good **training set**.

Trajectory optimization is an easier way to solve a control problem from one initial state, but it doesn't produce a **policy**.

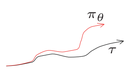
Guided policy Search uses trajectory optimization to *guide* the policy search. Trajectory optimization handles the temporal aspect of the task and constructs a training set that allows the complex, nonlinear policy to be trained with supervised learning.

2. Constrained Guided Policy Search

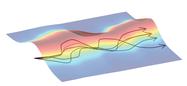
$\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$ – policy θ – policy parameters $\ell(\mathbf{x}_t, \mathbf{u}_t)$ – cost

$q(\tau) = q(\mathbf{x}_1)\prod_t q(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)q(\mathbf{u}_t|\mathbf{x}_t)$ – trajectory distribution

Supervised learning of $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$ with individual trajectories τ fails, since a small error at each time step can compound and place the policy in costly parts of the space.



Trajectory distributions provide many samples in a local neighborhood that will correct small mistakes made by $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$, allowing it to *stabilize* around the mean trajectory.

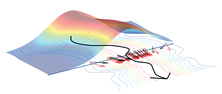


Adapting the distribution to match the policy while still achieving low cost will ensure that the policy can learn it accurately.

Can do all of this by approximately solving a constrained problem:

$$\min_{\theta, q(\tau)} \; \left.\begin{array}{l} E_{q(\tau)}[\ell(\tau)] - \mathcal{H}(q(\tau)) \end{array}\right\} \text{makes distribution broad}$$

$$\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)\|q(\mathbf{x}_t, \mathbf{u}_t))}_{\text{makes trajectory match the policy}} = 0 \;\forall t$$



3. Solving the Constrained Problem

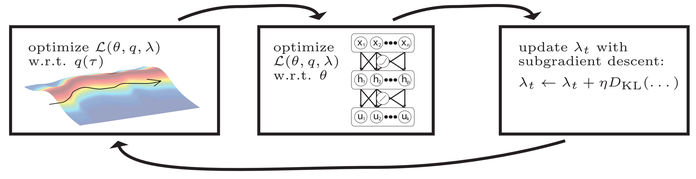
Consider the Lagrangian of the constrained problem in Box 2:

$$\mathcal{L}(\theta, q, \lambda) = E_q[\ell(\tau)] - \mathcal{H}(q) + \sum_{t=1}^{T} \lambda_t D_{\mathrm{KL}}(q(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)\|q(\mathbf{x}_t, \mathbf{u}_t))$$

We optimize the constrained problem with dual gradient descent, which consists of minimizing $\mathcal{L}(\theta, q, \lambda)$ w.r.t θ and q, and then performing a subgradient update on the dual variables:

$$\lambda_t \leftarrow \lambda_t + \eta D_{\mathrm{KL}}(q(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)\|q(\mathbf{x}_t, \mathbf{u}_t))$$

Optimizing $\mathcal{L}(\theta, q, \lambda)$ w.r.t. q corresponds to trajectory optimization, while optimizing it w.r.t. θ is supervised learning. By alternating between these two steps and incrementing λ_t, the policy and trajectory are gradually brought into agreement.



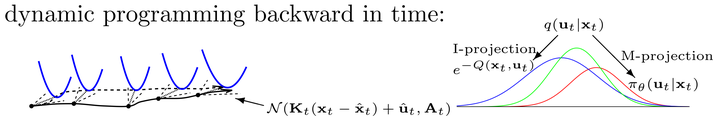
4. Trajectory Optimization

Can efficiently optimize $\mathcal{L}(q(\tau))$ by using **approximately Gaussian** trajectory distributions $q(\tau)$ with action conditional given by

$$q(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t(\mathbf{x}_t - \hat{\mathbf{x}}_t) + \hat{\mathbf{u}}_t, \mathbf{A}_t)$$

Optimization is similar to the Laplace approximation: we make linear Gaussian approximations to the dynamics and to $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$ by linearizing around the mean trajectory, and use a quadratic approximation of $\ell(\mathbf{x}_t, \mathbf{u}_t)$. This is also similar to LQR.

Under linear dynamics and quadratic cost, can show that the **value function** is quadratic, so $\mathbf{K}_t$, $\hat{\mathbf{x}}_t$, $\hat{\mathbf{u}}_t$, and $\mathbf{A}_t$ can be computed with dynamic programming backward in time:



After each dynamic programming pass, new linearizations are computed around the new mean trajectory, and the process repeats.

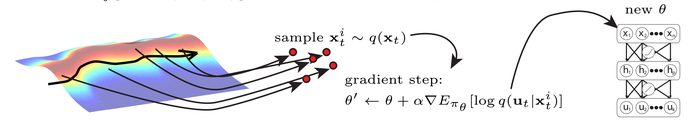
Cost terms $\ell(\mathbf{x}_t, \mathbf{u}_t)$ encourage *narrow* covariances $\mathbf{A}_t$, but policy KL-divergence constraints encourage *wider* $\mathbf{A}_t$ if policy and trajectory disagree, since $q(\tau)$ must be wider to correct all of the "mistakes" of the policy. As policy and trajectory come into agreement, this covariance becomes narrower.

5. Policy Optimization

The terms in $\mathcal{L}(\theta, q(\tau))$ that only depend on θ can be rewritten as

$$\mathcal{L}(\theta) = \sum_{t=1}^{T} \lambda_t \int q(\mathbf{x}_t) E_{\pi_\theta}[\log q(\mathbf{u}_t|\mathbf{x}_t)] d\mathbf{x}_t$$

We can therefore sample a "training points" from $q(\mathbf{x}_t)$ and minimize $\lambda_t E_{\pi_\theta}[\log q(\mathbf{u}_t|\mathbf{x}_t)]$ at each training point:



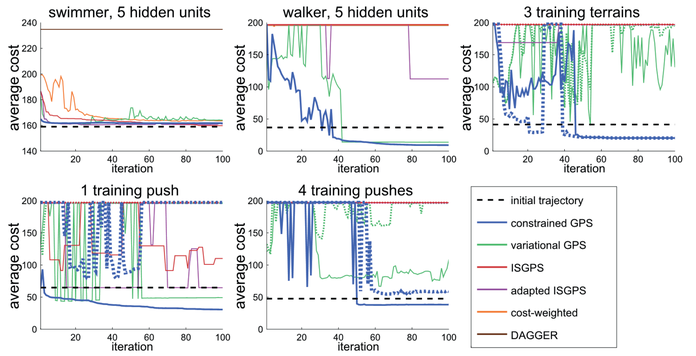
If the $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$ is a conditional (nonlinear) Gaussian, with mean $\mu^\pi(\mathbf{x}_t)$ and covariance $\Sigma^\pi(\mathbf{x}_t)$, $E_{\pi_\theta}[\log q(\mathbf{u}_t|\mathbf{x}_t)]$ is

$$\frac{1}{2}\{(\mathbf{K}_t\mathbf{x}_t + \hat{\mathbf{u}}_t - \mu^\pi)^{\mathrm{T}}\mathbf{A}_t^{-1}(\mathbf{K}_t\mathbf{x}_t + \hat{\mathbf{u}}_t - \mu^\pi) + \mathrm{tr}(\Sigma^\pi\mathbf{A}_t^{-1}) - \log|\Sigma^\pi|\}$$

Note that this is a weighted least-squares objective on $\mu^\pi(\mathbf{x}_t)$.

6. Experimental Evaluation

We used constrained guided policy search (constrained GPS) to learn controllers for a range of challenging locomotion tasks and compared to a variety of prior methods. Experiments included simulated swimming, walking on flat ground, walking on uneven terrain, and walking while recovering from very strong lateral pushes:



On the push recovery task, our method learned a highly generalizable push response strategy that could recover from a wide range of pushes, including test pushes that were not observed during training. Some of these test push responses are shown below, along with uneven terrain traversal, flat ground walking, and swimming: