Scalable Reinforcement Learning via Trajectory Optimization and Approximate Gaussian Process Regression

Yunpeng Pan, Xinyan Yan, Evangelos A. Theodorou and Byron Boots

Introduction

We present a model-based reinforcement learning framework based on differential dynamic programming (DDP) and sparse spectrum Gaussian process regression (SSGPR) [4].

Key features of our method

- ✓ In contrast to recent GP-based reinforcement learning approaches [1-3], our method is able to scale high-dimensional dynamical system and large dataset.
- ✓ Our method performs on-line optimization during interactions with the physical systems, differs from most related approaches.
- ✓ Local trajectory optimization is computationally efficient.
- ✓ Incremental update of learned models given new samples.

Problem Formulation

Consider a general unknown dynamical system

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x},\mathbf{u})\mathrm{d}t + \mathrm{d}\xi, \qquad \mathbf{x}(t_0) = \mathbf{x}_0, \qquad \mathrm{d}\xi \sim \mathcal{N}(0,\sigma_\xi)$$

The goal is to find controls that minimize the expected cost

$$J^\pi(\mathbf{x}(t_0)) = \mathbb{E}_\mathbf{x}\Big[\underbrace{q(\mathbf{x}(T))}_{\text{Terminal cost}} + \int_{t_0}^{T} \underbrace{\mathcal{L}(\mathbf{x}(t),\pi(\mathbf{x}(t)),t)\,\mathrm{d}t}_{\text{Running cost}}\Big]$$

Our analysis is based on discrete-time representations, e.g., $\mathbf{x}_k = \mathbf{x}(t_k)$

Model Learning via Incremental Sparse Spectrum Gaussian Process Regression

Probability distribution over transition dynamics — Gaussian processes

$$f(\tilde{\mathbf{x}}) \sim \mathcal{GP}(0, k(\tilde{\mathbf{x}}_i,\tilde{\mathbf{x}}_j)) \qquad \tilde{\mathbf{x}} = (\mathbf{x},\mathbf{u})$$

SE kernel $k(\tilde{\mathbf{x}}_i,\tilde{\mathbf{x}}_j) = \sigma_f^2 \exp(-\frac{1}{2}(\tilde{\mathbf{x}}_i-\tilde{\mathbf{x}}_j)^\mathsf{T}\mathbf{P}^{-1}(\tilde{\mathbf{x}}_i-\tilde{\mathbf{x}}_j))$

GP regression $\mathrm{d}\mathbf{x}^*|\tilde{\mathbf{X}},\mathrm{d}\mathbf{X},\tilde{\mathbf{x}}^* \sim \mathcal{N}(\mathbf{k}^{*\mathsf{T}}\mathbf{G}^{-1}\mathrm{d}\tilde{\mathbf{X}}, \quad k^* - \mathbf{k}^{*\mathsf{T}}\mathbf{G}^{-1}\mathbf{k}^*)$

Fourier feature approximation of shift-invariant kernel functions

$$k(\tilde{\mathbf{x}}_i-\tilde{\mathbf{x}}_j) = \int_{\mathbb{R}^n} e^{i\boldsymbol{\omega}^\mathsf{T}(\mathbf{x}_i-\mathbf{x}_j)}p(\boldsymbol{\omega})d\boldsymbol{\omega} = \mathbb{E}_{\boldsymbol{\omega}}[\phi_{\boldsymbol{\omega}}(\tilde{\mathbf{x}}_i)\phi_{\boldsymbol{\omega}}(\tilde{\mathbf{x}}_j)],\ \phi_{\boldsymbol{\omega}}(\tilde{\mathbf{x}}) = [\cos(\boldsymbol{\omega}^\mathsf{T}\tilde{\mathbf{x}})\ \ \sin(\boldsymbol{\omega}^\mathsf{T}\tilde{\mathbf{x}})]$$

Draw r random samples from the distribution $\mathrm{p}(\boldsymbol{\omega})$

Unbiased approx. $k(\tilde{\mathbf{x}}_i,\tilde{\mathbf{x}}_j) \approx \sum_{i=1}^{r}\phi_{\boldsymbol{\omega}_i}(\tilde{\mathbf{x}}_i)^\mathsf{T}\phi_{\boldsymbol{\omega}_i}(\tilde{\mathbf{x}}_j) = \boldsymbol{\phi}(\tilde{\mathbf{x}}_i)^\mathsf{T}\boldsymbol{\phi}(\tilde{\mathbf{x}}_j)$

Feature mapping $\boldsymbol{\phi}(\tilde{\mathbf{x}}) = \frac{\sigma_f}{\sqrt{r}}[\ \cos(\boldsymbol{\Omega}^\mathsf{T}\tilde{\mathbf{x}}) \quad \sin(\boldsymbol{\Omega}^\mathsf{T}\tilde{\mathbf{x}})\ ]^\mathsf{T},\ \boldsymbol{\Omega} \sim \mathcal{N}(\mathbf{0},\mathbf{P}^{-1})$

Posterior $\mathrm{d}\mathbf{x}^*|\tilde{\mathbf{X}},\mathrm{d}\mathbf{X},\tilde{\mathbf{x}}^* \sim \mathcal{N}(\mathbf{w}^\mathsf{T}\boldsymbol{\phi}^*, \sigma_n^2(1+\boldsymbol{\phi}^{*\mathsf{T}}\mathbf{A}^{-1}\boldsymbol{\phi}^*))$

$$\boldsymbol{\phi}(\tilde{\mathbf{x}}^*) = \boldsymbol{\phi}^*, \mathbf{w} = \mathbf{A}^{-1}\boldsymbol{\Phi}\mathrm{d}\mathbf{X}, \mathbf{A} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\mathsf{T} + \sigma_n^2\boldsymbol{\Sigma}_p^{-1}, \boldsymbol{\Phi} = [\ \boldsymbol{\phi}(\tilde{\mathbf{x}}_1) \quad \dots \quad \boldsymbol{\phi}(\tilde{\mathbf{x}}_N)\ ].$$

Incremental update given a new sample

Keep track of Cholesky factor $\mathbf{A} = \mathbf{R}^\mathsf{T}\mathbf{R}$ **Rank-1update** $\mathbf{R}$

Complexity O(r^2), where r is the number of random feature

Controller Learning via Differential Dynamic Programming

Linear approximation of the SSGP dynamics model around a trajectory

$$\mathbf{x}_{k+1} = \mathcal{F}(\mathbf{x}_k,\mathbf{u}_k) \longrightarrow \delta\mathbf{x}_{k+1} = \mathcal{F}_k^x\delta\mathbf{x}_k + \mathcal{F}_k^u\delta\mathbf{u}_k$$

Bellman equation for value function

$$V(\mathbf{x}_k,k) = \min_{\mathbf{u}_k}\mathbb{E}\Big[\underbrace{\mathcal{L}(\mathbf{x}_k,\mathbf{u}_k) + V\big(\mathcal{F}(\mathbf{x}_k,\mathbf{u}_k),k+1\big)}_{Q(\mathbf{x}_k,\mathbf{u}_k)}|\mathbf{x}_k\Big]$$

Quadratic approximation of the value function

$$Q_k(\bar{\mathbf{x}}_k+\delta\mathbf{x}_k,\bar{\mathbf{u}}_k+\delta\mathbf{u}_k) \approx Q_k^0 + Q_k^x\delta\mathbf{x}_k + Q_k^u\delta\mathbf{u}_k + \frac{1}{2}\begin{bmatrix}\delta\mathbf{x}_k\\ \delta\mathbf{u}_k\end{bmatrix}^\mathrm{T}\begin{bmatrix}Q_k^{xx} & Q_k^{xu}\\ Q_k^{ux} & Q_k^{uu}\end{bmatrix}\begin{bmatrix}\delta\mathbf{x}_k\\ \delta\mathbf{u}_k\end{bmatrix}$$

Optimal control policy $\hat{\mathbf{u}}_k = \bar{\mathbf{u}}_k + \delta\hat{\mathbf{u}}_k$

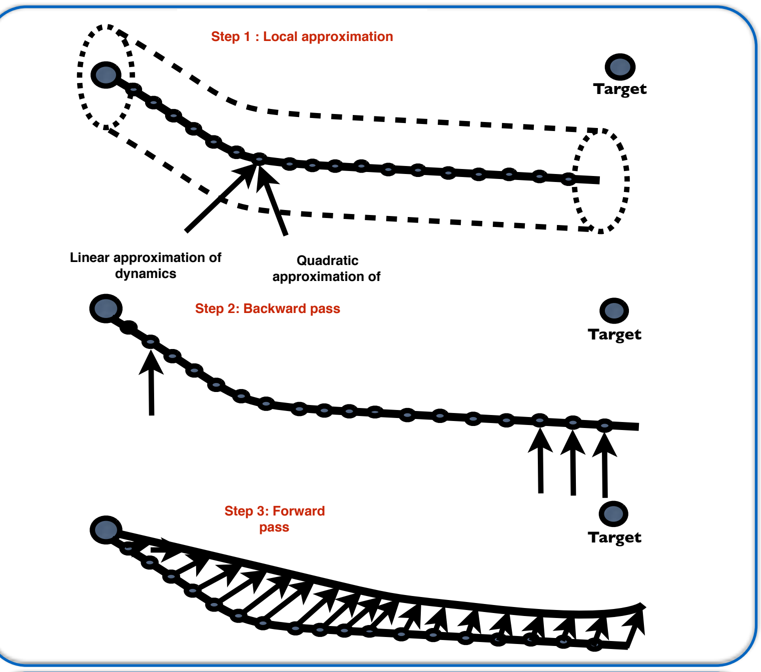
$$\delta\hat{\mathbf{u}}_k = \arg\min_{\delta\mathbf{u}_k}\Big[Q_k(\bar{\mathbf{x}}_k+\delta\mathbf{x}_k,\bar{\mathbf{u}}_k+\delta\mathbf{u}_k)\Big] = \underbrace{-(Q_k^{uu})^{-1}Q_k^u}_{\mathbf{I}_k}\underbrace{-(Q_k^{uu})^{-1}Q_k^{ux}}_{\mathbf{L}_k}\delta\mathbf{x}_k = \mathbf{I}_k + \mathbf{L}_k\delta\mathbf{x}_k$$

Backward propagation

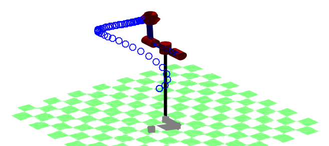
$$Q_k^x = \mathcal{L}_k^x + V_k^x\mathcal{F}_k^x, Q_k^u = \mathcal{L}_k^u + V_k^x\mathcal{F}_k^u, Q_k^{xx} = \mathcal{L}_k^{xx} + (\mathcal{F}_k^x)^\mathrm{T}V_k^{xx}\mathcal{F}_k^x,$$

$$Q_k^{ux} = \mathcal{L}_k^{ux} + (\mathcal{F}_k^u)^\mathrm{T}V_k^{xx}, \mathcal{F}_k^x, Q_k^{uu} = \mathcal{L}_k^{uu} + (\mathcal{F}_k^u)^\mathrm{T}V_k^{xx}\mathcal{F}_k^u,$$

$$V_{k-1} = V_k + Q_k^u\mathbf{I}_k, V_{k-1}^x = Q_k^x + Q_k^u\mathbf{L}_k, V_{k-1}^{xx} = Q_k^{xx} + Q_k^{xu}\mathbf{L}_k.$$

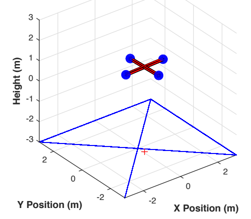

Step 1 : Local approximation
Target
Linear approximation of dynamics
Quadratic approximation of
Step 2: Backward pass
Target
Step 3: Forward pass
Target

Numerical Results



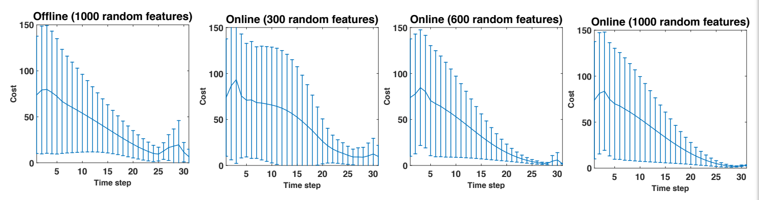
PUMA-560

Goal: steer the end-effector to the desired position and orientation in 50 time steps.

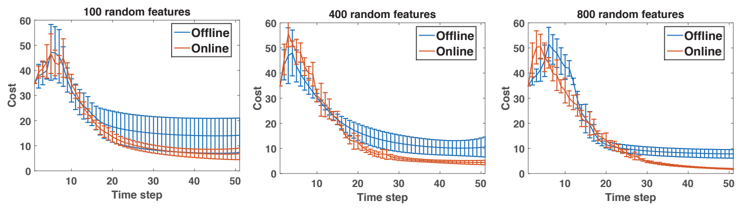


Quadrotor

Goal: fly to 10 different targets from the same initial position in 30 time steps.

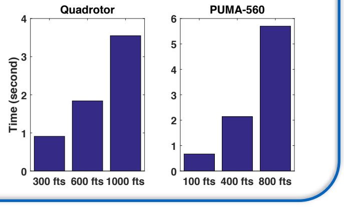


Quad rotor example. Each figure show the mean and standard deviation of final trajectory costs over 10 independent trials using offline and online learning. We obtained training data by performing 150 rollouts (4500 data points) around a pre-specified trajectory



PUMA-560 example. Each figure show the mean and standard deviation of final trajectory costs over 10 independent trials using offline and online learning. We collected 5000 data points offline from random explorations

Average online computation time per time step



References

[1] M.Deisenroth, D.Fox, and C.Rasmussen. Gaussian processes for data-efficient learning in robotics and control. PAMI 2015.
[2] Y.Pan and E.Theodorou. Probabilistic Differential Dynamic Programming. NIPS 2014.
[3] A.Kupcsik, M.Deisenroth,J.Peters,A.Loh,P.Vadakkepat and G.Neumann. Model-based contextual policy search for data-efficient generalization of robot skills. Artificial Intelligence 2015.
[4] M.La´zaro-Gredilla,J.Quin˜onero-Candela,C.E.Rasmussen,andA.R.Figueiras-Vidal.Sparsespectrum gaussian process regression. JMLR 2010.
[5] A. Rahimi and B. Recht. Random features for large-scale kernel machines. NIPS 2007.

D.Guggenheim School of Aerospace Engineering
Institute for Robotics and Intelligent Machines
Georgia Institute of Technology