Neural-iLQR: A Learning-Aided Shooting Method for Trajectory Optimization

Zilong Cheng, Yulin Li, Kai Chen, Jianghua Duan, Jun Ma, and Tong Heng Lee

Abstract—Iterative linear quadratic regulator (iLQR) has gained wide popularity in addressing trajectory optimization problems with nonlinear system models. However, as a model-based shooting method, it relies heavily on an accurate system model to update the optimal control actions and the trajectory determined with forward integration, thus becoming vulnerable to inevitable model inaccuracies. Recently, substantial research efforts in learning-based methods for optimal control problems have been progressing significantly in addressing unknown system models, particularly when the system has complex interactions with the environment. Yet an incremental machine learning method or a deep neural network is normally required to fit a substantial scale of sampling motion data. Existing works mainly focus on capturing the whole picture of the motion pattern from trial data (which is essentially noisy and hard to identify with any simple predefined structures) for complex robot systems. Instead, we present Neural-iLQR, a learning-aided shooting method over the unconstrained control space, in which a neural network with a simple structure is used to drive the optimization process towards optimal direction. Through comprehensive evaluations on two illustrative control tasks using three different kinds of lightweight network structures, the proposed method is shown to outperform the conventional iLQR significantly with fast and continuous cost convergence in the presence of inaccuracies in system models, which demonstrates the generalizability and robustness of the proposed learning-aided method.

I. INTRODUCTION

The last decade has witnessed substantial achievements in the context of trajectory optimization, pervading different application domains including unmanned aerial vehicles (UAVs) [1], autonomous driving [2], quadrupeds [3], mobile manipulators [4], etc. However, it is still an open and challenging problem on the generation of a satisfying trajectory in complex scenarios, especially when the state/control space is inherently high-dimensional, the system model is nonlinear, and the non-smooth contact and constraints are introduced in most of the robot systems [4], [5]. For such problems, the model-based method and the learning-based method are extensively investigated.

In terms of the model-based method in trajectory optimization, the most widely used Model Predictive Control (MPC) has demonstrated compelling results in controlling complex robot systems [6] [7]. Fundamentally, MPC is a powerful optimal control architecture with the ability to encode complex high-level tasks into cost functions and deal with nonlinear system models while respecting various constraints. It solves for the optimal control and state trajectory repeatedly in a receding-horizon fashion. As an indirect second-order shooting method to solve the MPC problem [8], differential dynamic programming (DDP) has gained popularity in the robotics community due to its high efficiency in dealing with nonlinear dynamic systems. In each stage, with the quadratic approximation of the system dynamics around the nominal state-input trajectory, it drives the
current input trajectory towards the optimal direction [9]. To further reduce the computation time, iterative linear quadratic regulator (iLQR) was proposed as a variant of DDP following a similar structure but using first-order approximation of the system dynamics instead. Remarkably, iLQR has been successfully applied to robot systems with precise models [10]–[12]. However, considering the complex dynamics of a robot system, the inevitable existence of model inaccuracy could lead to the deviation of obtained solution from the optimal control policy. Fig. 1 shows a typical failure case in model-based iLQR when modeling inaccuracy is introduced.

On the other hand, model-free methods have recently shown promising results in learning the system model or the optimal policy. In [13], the control problem is modeled as a Markov Decision Process (MDP) [14] and the optimal policy is then learned using Reinforcement Learning (RL) [15]. In order to learn long-term transition dynamics instead of step transitions, latent variables recurrent network is utilized to improve the performance in long prediction horizon [16]. Although learning-based methods outperform model-based methods in their versatility when dealing with complex control problems, deep neural networks and a substantial scale of exploration samples are normally required to reach a satisfying solution.

To deal with the aforementioned drawbacks, a learning-aided shooting method named Neural-iLQR is presented for trajectory optimization over the unconstrained control space, which avoids the requirement of any prior knowledge of the system model in trajectory optimization tasks. In this approach, random trials are performed such that a dataset is collected, and then a filtered neural network is used to fit the measurement data locally in the current iteration. Subsequently, the well-trained neural network is implemented in the execution of the backward pass of iLQR method such that the trajectory optimization problem is solved iteratively. Moreover, we evaluate several critical factors pertinent to the optimization results by comparing the performance of different network structures and other related parameters. Finally, we demonstrate that the online-retraining mechanism could prevent the optimization process from trapping into the local minimum, whereby the optimality of the obtained trajectory can be further improved compared to the conventional iLQR method.

The main contributions of our work are as follows:

- A learning-aided trajectory optimization method over the unconstrained control space is proposed, which generates satisfying optimal control policy for nonlinear robot systems without relying on any prior knowledge of the system dynamics.
- Instead of modeling the system dynamics accurately with incremental machine learning methods or deep neural networks, a lightweight neural network with polished gradient information is used in the proposed learning-aided framework, leading to a sample-efficient pipeline in simultaneous network training and trajectory update with fast and continuous convergence towards optimal direction.
- The universality and generalizability of our proposed framework are demonstrated by comparing three different kinds of neural networks representing different systems, and it is shown that our framework could be appropriately generalized to different trajectory optimization applications with only a slight structure change of the lightweight neural network.
- As demonstrated from the experiments, the proposed learning-aided framework mitigates the effects of model inaccuracies and exhibits improved robustness over the conventional iLQR approach.

II. RELATED WORK

A similar framework utilizing learned system dynamics in generating the optimal policy has been widely explored in the literature on model-based reinforcement learning. Simultaneously training of the optimal policy network and the transition dynamics model is proposed in [17], in which the iteratively fitted linear models are used to guide the policy search. Another developing branch in this research field which we focus on in this paper starts from [18], where the authors combine the incrementally learned system dynamics with conventional DDP-based feedback trajectory optimization method to generate the optimal control sequence. In [18], the iterative linear quadratic Gaussian (iLQG) formulation is derived efficiently using the incremental learned model identified by a locally weighted projection regression (LWR) method [19]. Moreover, in [20], a similar framework is scaled to apply to dexterous manipulation with 100-dimensional state space, where the trajectory optimization method is combined with an iteratively learned time-varying linear-Gaussian dynamics.

Although the aforementioned works provide an effective framework in combining the advantages of both the conventional iLQR and learning dynamic systems from time series, they lack comparison with the conventional optimal control method with a precise system model to show the optimality and robustness of the generated trajectory with their learned model. Besides, they either use a local linear model to fit the system dynamics which may fail when dealing with complex robot systems with high nonlinearity, or they choose a predefined model structure and intend to cover the whole motion patterns with the incrementally collected trials data (whose performance and versatility would be limited by the chosen structure, predefined model parameters, and accumulated estimation/prediction errors) [21].

To overcome the limits with traditional model fitting techniques and improve the robustness toward system modeling inaccuracy, deep neural networks have shown their powerful capability in learning system dynamics [21] [22]. In [22], the powerful deep learning technique is shown to be suitable for combining the DDP-based trajectory optimization method for its prominent advantages in fitting complex nonlinear system dynamics and automatic feature selection ability. Several closely related works are developed very recently. In [23], an iLQR framework called the curious iLQR is proposed, in which the system dynamics is reflected by the Bayesian
modeling. In [24], a multi-layer neural network is used to model the dynamics of off-road and on-road vehicles which is then used in iLQR. These methods are not sample efficient due to the large size of the built neural network. A recent work combines iLQR and RL in an augmenting way [25], which uses RL-based terminal cost and shortens the iLQR horizon adaptively in each stage to relax the requirement of an accurate system model. However, it still requires a nominal system model to drive the optimization process.

III. NEURAL-iLQR

A. Problem Definition

Generally, the trajectory planning problem can be expressed as a nonlinear optimization problem given in (1), where \(x(\tau) \in \mathbb{R}^n\) and \(u(\tau) \in \mathbb{R}^m\) denote the state and action at the time stamp \(\tau\), respectively; \(J_r(x(\tau), u(\tau))\) denotes the cost at each time stamp \(\tau\) with respective to the pair of state and action; \(\delta_T\) represents the terminal cost at the time stamp \(T\); \(f(x(\tau), u(\tau))\) is the dynamic function of the system which is not restricted to be linear; \(x_0\) denotes the initial state of the system.

Minimize

\[
\phi_T(x(T)) + \sum_{\tau=0}^{T-1} J_r(x(\tau), u(\tau))
\]

subject to

\[
x(\tau + 1) = f(x(\tau), u(\tau))
\]

\(\tau = 0, 1, \ldots, T - 1\)

\(x(0) = x_0\), \hspace{1cm} (1)

B. Overview of Neural-iLQR

With the development of the conventional model-based iLQR, Neural-iLQR is proposed in this section with a detailed analysis to solve the optimization problem (1). Essentially, Neural-iLQR utilizes a simple neural network to capture the gradient information of the dynamic function locally from trail data, and procedures of iLQR can be performed entirely based on the measurement data with the incorporation of the neural network.

As shown in Fig. 2, the implementation of Neural-iLQR is summarized. In the beginning, an empty dataset is required to be initialized with \(p\) random trials of prediction horizon \(T\) performed in the dynamic system (i.e., the system runs arbitrarily with a sequence of control actions chosen randomly). Upon completion of the dataset initialization, a lightweight neural network structure can be chosen to fit the system dynamic function. After the setup, we conduct the backward pass and forward pass iteratively following the conventional DDP/iLQR method except that the gradient and Hessian of the dynamic function are replaced by the polished analytical derivatives of the neural network. Moreover, online data collection and neural network retraining mechanisms are incorporated, ensuring continuous convergence of the trajectory cost.

Remark 1: Essentially different from [19]–[22], a lightweight neural network is systematically incorporated into the proposed learning-aided framework, which can be retrained with motion data around the current trajectory iteratively in a sample-efficient way. In this sense, the proposed framework can be applied to different systems with only a slight structure change of the neural network.

C. Backward Pass of Neural-iLQR

The primary idea behind the backward pass of DDP/iLQR [27] can be straightforwardly represented by the Bellman equation given by

\[
V_\tau(x(\tau)) = \min_{u(\tau)} \left\{ J_r(x(\tau), u(\tau)) + V_{\tau+1}(f(x(\tau), u(\tau))) \right\},
\]

where \(V_\tau(x(\tau))\) and \(V_{\tau+1}(f(x(\tau), u(\tau)))\) denote the value functions with respect to the current state \(x(\tau)\) and the state at the next time stamp \(x(\tau + 1)\), respectively.

The perturbed Q-function is then introduced and approximated by the second-order Taylor expansion as in (3), where \(\delta x(\tau)\) and \(\delta u(\tau)\) represent the amount of change with respect to the nominal state and action at the time \(\tau\).

\[
Q_\tau(\delta x(\tau), \delta u(\tau)) \approx \frac{1}{2} \begin{bmatrix} 1 & \delta x(\tau) \\ \delta x(\tau)^T & (Q_\tau)_x & (Q_\tau)_u \\ (Q_\tau)_x^T & (Q_\tau)_{ux} & (Q_\tau)_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x(\tau) \\ \delta u(\tau) \end{bmatrix},
\]

where

\[
(Q_\tau)_x = (J_\tau)_x + N_x^T (V_{\tau+1})_x,
(Q_\tau)_u = (J_\tau)_u + N_u^T (V_{\tau+1})_u,
(Q_\tau)_{xx} = (J_\tau)_{xx} + N_x^T (V_{\tau+1})_{xx} N_x + (V_{\tau+1})_x \cdot N_{xx},
(Q_\tau)_{ux} = (J_\tau)_{ux} + N_u^T (V_{\tau+1})_{ux} N_x + (V_{\tau+1})_x \cdot N_{ux},
(Q_\tau)_{uu} = (J_\tau)_{uu} + N_u^T (V_{\tau+1})_{uu} N_u + (V_{\tau+1})_x \cdot N_{uu}.
\]

We denote the neural network that fits the system dynamic function \(f\) as \(N\). The optimal solution to the perturbed control action \(\delta u(\tau)^*\) at the time stamp \(\tau\) can then be calculated by

\[
\delta u(\tau)^* = \arg \min_{\delta u(\tau)} Q_\tau(\delta x(\tau), \delta u(\tau)),
\]

which gives

\[
\delta u(\tau)^* = k(\tau) + K(\tau) \delta x(\tau),
\]

where \(k(\tau) \in \mathbb{R}^m\) and \(K(\tau) \in \mathbb{R}^{m \times n}\) are the feedforward vector and feedback matrix for the perturbed Q-function at the time stamp \(\tau\), respectively, and they can be explicitly represented by

\[
k(\tau) = - (Q_\tau)_{uu}^{-1} (Q_\tau)_u \hspace{1cm} (7a)
\]

\[
K(\tau) = - (Q_\tau)_{uu}^{-1} (Q_\tau)_{ux}. \hspace{1cm} (7b)
\]

Then the gradient and Hessian of the value function at each time stamp can be obtained by substituting (7a) and (7b) to the Taylor expansion of perturbed Q-function, and it follows that

\[
(V_\tau)_x = (Q_\tau)_x - K(\tau)^T (Q_\tau)_{uu} k(\tau) \hspace{1cm} (8a)
\]

\[
(V_\tau)_{xx} = (Q_\tau)_{xx} - K(\tau)^T (Q_\tau)_{uu} K(\tau). \hspace{1cm} (8b)
\]
D. Forward Pass of Neural-iLQR

The last and principal step in Neural-iLQR is the forward pass. An indispensable line search strategy is performed at the beginning of the forward pass to find a trajectory with better performance based on the given feedback matrices and feedforward vectors at different time stamps. The basic idea of the line search [9] can be realized by

\[ \delta u(\tau) = \alpha k(\tau) + K(\tau) \delta x(\tau), \]

where \( \alpha \) is the step size parameter to be determined in the line search iteration. Then the forward roll-out is conducted by feeding the perturbed action sequence to the system dynamic model \( f \):

\[ u(\tau) = \hat{u}(\tau) + \delta u(\tau) \]
\[ x(\tau + 1) = f(x(\tau), u(\tau)). \]

The nominal trajectory \((\hat{x}, \hat{u})\) can be updated to a new feasible trajectory obtained in the forward pass, and the updated feasible trajectory can be used to initiate the next Neural-iLQR iteration and retrain the neural network.

The stopping criterion for one iteration can be chosen as the difference of the objective function value between the current and the latest nominal trajectory, once the criterion is satisfied, we will retrain the neural network on the updated dataset. Noting that the data collecting and network retraining process need to be carefully designed to guarantee that the performance of the trajectory improves with iterations.

For the scenario when it is still not possible to improve the dynamic system performance as the maximum number of line search iterations is reached. The solving process is considered as trapping into local minimum in this situation.

Remark 2: Perturbations are added to the sequence of actions when generating the current trajectory, and the neural network is then retrained to escape from the local minimum, which facilitates the continuous convergence of the trajectory towards optimality.

IV. Experiments

In this section, we comprehensively evaluate Neural-iLQR using numerical simulation and MuJoCo [26] physics engine with two illustrative examples: a vehicle tracking problem and a cartpole control problem. To demonstrate the practicality and effectiveness of Neural-iLQR, we use the conventional iLQR method as the benchmark to show that the proposed model-free method is comparable to or even better than the model-based counterpart. Particularly, the robustness of our method is testified when model inaccuracy is introduced. Furthermore, the influence of several critical factors in Neural-iLQR is discussed thoroughly.

A. System Setups

1) Vehicle Tracking Problem Formulation: The state vector of the vehicle is defined as \( x = [p_x, p_y, \theta, v]^T \), where \( p_x \) and \( p_y \) denote the position of the center of the rear axis in the Cartesian coordinates; \( \theta \) denotes the heading angle of the vehicle; \( v \) denotes the velocity of the vehicle. The action vector is defined as \( u = [\omega, a]^T \), where \( \omega \) denotes the steering angle and \( a \) denotes the acceleration. The control target is to drive the vehicle in a straight line at a specified speed.

2) Cartpole Control Problem Formulation: The state vector of the cartpole is defined as \( x = [\theta, \omega, p, v]^T \), where \( \theta \) and \( \omega \) denote the angle between the pole and the vertical direction and its corresponding angular velocity, respectively;
\[ p \text{ and } v \text{ denote the position and the velocity of the cart, respectively. The control action } F \text{ means the force applied on the cart horizontally. We are going to swing up the pole and keep it balanced around the upward position by applying force to the cart.} \]

Noted that dynamic functions of the systems are only required for calculation in conventional model-based iLQR, and it can be fully replaced by the simulation environment and the neural network in the implementation of Neural-iLQR.

B. Overall Performance of Neural-iLQR

Provided with an accurate dynamic model, the conventional iLQR method is proven to achieve effective optimization outcomes in such two examples indicated in Fig. 3. After several iterations, the trajectory will converge asymptotically, and the objective function is reduced to a satisfying point; thus, it is a good benchmark for comparison with our proposed model-free method. The optimal objective function values reached by the conventional model-based iLQR are 10192 and 1642 respectively for vehicle tracking and cartpole control examples.

As observed from Fig. 3, the proposed model-free Neural-iLQR method can successfully generate a reliable trajectory for the control problem, attaining objective function values at 992.9 and 1178, which shows comparable optimization performance compared to the conventional iLQR method. We apply simple neural networks to fit the local trajectory data in the current iteration and use its filtered gradient information to guide the optimization directions. When it gets stuck at some local minimum points using the current neural network, the online data collection and retraining mechanisms discussed in Section III-D empower the proposed method with the ability to avoid trapping into the local minimum. It renders the possibility for the Neural-iLQR to perform even better than the conventional model-based iLQR as shown in Fig. 3.

C. Comparison of Effects on Critical Factors

The influence of the selected critical factors in the Neural-iLQR architecture is discussed in this section, and we find that these critical factors could directly affect the optimization performance of Neural-iLQR, and thus it renders the possibility to continuously improve the results by adjusting these factors in the proposed architecture.

1) Neural Network Architecture: We propose three neural network structures and demonstrate the performance attained by each in this section. The first two are fully connected neural networks (FCNN) with two hidden layers and one output layer, structures are shown in Table I.

| Network  | First Hidden Layer | Second Hidden Layer | Output Layer |
|----------|--------------------|---------------------|--------------|
| Small FCNN | \((m + n) \times 128\) | \(128 \times 64\) | \(64 \times n\) |
| Large FCNN | \((m + n) \times 1024\) | \(1024 \times 512\) | \(512 \times n\) |

Shown in Fig. 2, the third type of the neural network is a residual neural network. It consists of four ResLinear modules and one fully connected linear output layer. Batch normalization and ReLU are required before each fully connected layer.

We define the deviation between the objective function values obtained by Neural-iLQR and the conventional iLQR method given the dynamic model during iterations as \(d\) and the iteration number when it first reaches the minimum objective function value as \(k\). Neural-iLQR is randomly performed five times with each neural network structure, and the conventional iLQR method is performed with the same parameters as used in Neural-iLQR for the fair comparison.

Fig. 4 shows the deviation of the Neural-iLQR method with respect to different neural network structures after 500 Neural-iLQR iterations, respectively. From the results in Table II, it can be seen that the proposed Neural-iLQR method with all three neural network structures can achieve satisfying performance in terms of the objective function value but lead to various performance. The large FCNN structure shows the highest performance in the vehicle tracking example, but the residual neural network structure demonstrates the best performance both in its optimality and fast convergence in the cartpole control example.

2) Gaussian Filter: Fig. 5 shows the control action sequences obtained by applying Gaussian filter with different values.
Fig. 4. Objective function value in iterations using Neural-iLQR with different neural network structure.

Fig. 5. Control action sequence obtained by Neural-iLQR with different standard deviation parameters in the Gaussian filter.

standard deviations. We can see that the smoothness of the control signals in the two illustrative examples is significantly improved with a larger standard deviation of the Gaussian filter. However, to the best knowledge of the authors, an extremely large standard deviation parameter also leads to unsatisfying optimization results due to the lack of sudden changes in control actions.

D. Robustness to Modeling Inaccuracy

In this section, we further demonstrate the robustness of the proposed method against model inaccuracy compared to the conventional iLQR method in MuJoCo. We build a cartpole model and use the simulation model to conduct forward dynamics by feeding inputs to the environment. We can obtain the trajectory and collect real-time data for training in Neural-iLQR, which is convincing and close to the scenarios in the real world. Modeling inaccuracy is inevitable in the real-world situation, in this case, we introduce the model inaccuracy to the system by adjusting the model parameter in MuJoCo.

As indicated in Fig. 6 and Table III, the conventional model-based iLQR shows its effectiveness in achieving satisfying optimization results with the accurate model. The objective function value reaches 2.334 and the mean square error (MSE) of $\theta$ for the generated trajectory is 4.750. However, its optimization performance will be significantly affected by the modeling inaccuracy and it may even fail when large inaccuracy is introduced. Meanwhile, we can

| Model Inaccuracy | Model-based iLQR | Neural-iLQR |
|------------------|------------------|-------------|
|                  | Success | $\theta_{error}$ | Obj.Val ($\times 10^{3}$) | Success | $\theta_{error}$ | Obj.Val ($\times 10^{3}$) |
| 0%               | Yes     | 4.750           | 2.334       | Yes     | 7.051           | 3.269       |
| 20%              | Yes     | 7.872           | 3.436       | Yes     | 6.965           | 2.477       |
| 40%              | No      | 9.456           | 9.833       | Yes     | 7.984           | 3.458       |
| 60%              | No      | 9.287           | 12.699      | Yes     | 7.657           | 3.038       |

Table III: Comparison of Neural-iLQR and Model-based iLQR for Cartpole Control Experiment
see from the results in Table III that the proposed Neural-iLQR method shows comparable capability in generating the optimal trajectory towards the control target without any prior knowledge of the system model, and its robustness and adaptability to inaccurate model is demonstrated.

V. CONCLUSION

This paper investigates the development of Neural-iLQR, a learning-aided shooting method for trajectory optimization over the unconstrained control space. In view of an unknown dynamic system, a neural network is utilized to fit the dynamic function in an iterative framework, which enables the use of the iLQR method in trajectory planning problems. The estimated gradient matrix of the dynamic function is derived, and the improved feedforward iteration is proposed to deal with the inaccuracy and imprecision in the optimization problem. As a result, the refined iLQR method can be applied completely without any prior information about the dynamic system. Moreover, the trajectory resulting from the Neural-iLQR method can be even better than the conventional iLQR method, as the local optimal point can be escaped with the deployment of the further exploration procedure. Finally, illustrative examples are used to validate the performance of the proposed Neural-iLQR method and detailed discussions are presented. It is worthwhile to highlight that the proposed framework demonstrates satisfying performance with all three neural network structures, and due to the effectiveness and universality, the framework could be suitably adjusted or extended to address practical issues in many real-world applications. One promising future work is to extend the proposed learning-aided framework with online learning techniques to achieve real-time learning and control.

REFERENCES

[1] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors,” in 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 2520–2525.
[2] W. Xu, Q. Wang, and J. M. Dolan, “Autonomous vehicle motion planning via recurrent spline optimization,” in 2021 IEEE International Conference on Robotics and Automation, 2021, pp. 7730–7736.
[3] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, “Gait and trajectory optimization for legged systems through phase-based end-effector parameterization,” IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 1560–1567, 2018.
[4] J. P. Sleiman, F. Farshidian, and M. Hutter, “Constraint handling in continuous-time DDP-based model predictive control,” in 2021 IEEE International Conference on Robotics and Automation, 2021, pp. 8209–8215.
[5] B. Zhou, F. Gao, J. Pan, and S. Shen, “Robust real-time UAV replanning using guided gradient-based optimization and topological paths,” in 2020 IEEE International Conference on Robotics and Automation, 2020, pp. 1208–1214.
[6] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, “A unified MPC framework for whole-body dynamic locomotion and manipulation,” IEEE Robotics and Automation Letters, vol. 6, no. 3, pp. 4688–4695, 2021.
[7] M. V. Minniti, R. Grandia, K. Fäh, F. Farshidian, and M. Hutter, “Model predictive robot-environment interaction control for mobile manipulation tasks,” in 2021 IEEE International Conference on Robotics and Automation, 2021, pp. 1651–1657.
[8] K. Stachowiecz and E. A. Theodorou, “Optimal-horizon model predictive control with differential dynamic programming,” in 2022 International Conference on Robotics and Automation (ICRA). IEEE, 2022, pp. 1440–1446.
[9] Y. Tassa, N. Mansard, and E. Todorov, “Control-limited differential dynamic programming,” in 2014 IEEE International Conference on Robotics and Automation, 2014, pp. 1168–1175.
[10] W. Li and E. Todorov, “Iterative linear quadratic regulator design for nonlinear biological movement systems,” in 2004 International Conference on Informatics in Control, Automation and Robotics. Citeseer, 2004, pp. 222–229.
[11] J. Ma, Z. Cheng, X. Zhang, M. Tomizuka, and T. H. Lee, “Alternating direction method of multipliers for constrained iterative LQR in autonomous driving,” IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 7, pp. 9240–9251, 2022.
[12] X. Zhang, Z. Cheng, J. Ma, S. Huang, F. L. Lewis, and T. H. Lee, “Semi-definite relaxation-based ADMM for cooperative planning and control of connected autonomous vehicles,” IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 7, pp. 9240–9251, 2022.
[13] W. Zhao, H. Liu, and F. L. Lewis, “Robust formation control for cooperative underactuated quadrotors via reinforcement learning,” IEEE Transactions on Neural Networks and Learning Systems, 2020.
[14] C. M. Bishop, Pattern Recognition and Machine Learning. New York, NY, USA: Springer-Verlag, 2006.
[15] D. Bertsekas, Dynamic Programming and Optimal Control: Volume I. Belmont, MA, USA: Athena Scientific, 2012.
[16] N. R. Ke, A. Singh, A. Touati, A. Goyal, Y. Bengio, D. Parikh, and D. Batra, “Learning dynamics model in reinforcement learning by incorporating the long term future,” arXiv preprint arXiv:1903.01599, 2019.
[17] S. Levine and P. Abbeel, “Learning neural network policies with guided policy search under unknown dynamics,” Advances in Neural Information Processing Systems, vol. 27, 2014.
[18] D. Mitrovisc, S. Klankel, and S. Vijayakumar, “Adaptive optimal feedback control with learned internal dynamics models,” in From Motor Learning to Interaction Learning in Robots. Berlin: Springer, 2010, vol. 264, pp. 65–84.
[19] S. Vijayakumar and S. Schaal, “Locally weighted projection regression: An o(n) algorithm for incremental real time learning in high dimensional space,” in International Conference on Machine Learning, vol. 1. Morgan Kaufmann, 2000, pp. 288–293.
[20] V. Kumar, E. Todorov, and S. Levine, “Optimal control with learned local models: Application to dexterous manipulation,” in 2016 IEEE International Conference on Robotics and Automation. IEEE, 2016, pp. 378–383.
[21] S. Mukhopadhyay and S. Banerjee, “Learning dynamical systems in noise using convolutional neural networks,” Chaos: An Interdisciplinary Journal of Nonlinear Science, vol. 30, no. 10, p. 103125, 2020.
[22] A. Yamaguchi and C. G. Atkeson, “Neural networks and differential dynamic programming for reinforcement learning problems,” in 2016 IEEE International Conference on Robotics and Automation, 2016, pp. 5434–5441.
[23] S. Bechtle, Y. Lin, A. Rai, L. Righetti, and F. Meier, “Curious iLQR: Resolving uncertainty in model-based RL,” in Conference on Robot Learning. PMLR, 2020, pp. 162–171.
[24] A. Nagariya and S. Saripalli, “An iterative LQR controller for off-road and on-road vehicles using a neural network dynamics model,” arXiv preprint arXiv:2007.11449, 2020.
[25] T. Zong, L. Sun, and Y. Liu, “Reinforced iLQR: A sample-efficient robot locomotion learning,” in 2021 IEEE International Conference on Robotics and Automation, 2021, pp. 5906–5913.
[26] E. Todorov, T. Erez, and Y. Tassa, “MuJoCo: A physics engine for model-based control,” in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2012, pp. 5026–5033.
[27] D. Mayne, “A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems,” International Journal of Control, vol. 3, no. 1, pp. 85–95, 1966.