

Value iteration with deep neural networks for optimal control of input-affine nonlinear systems

Hirofumi Beppu, Ichiro Maruta & Kenji Fujimoto

To cite this article: Hirofumi Beppu, Ichiro Maruta & Kenji Fujimoto (2021) Value iteration with deep neural networks for optimal control of input-affine nonlinear systems, SICE Journal of Control, Measurement, and System Integration, 14:1, 140-149, DOI: [10.1080/18824889.2021.1936817](https://doi.org/10.1080/18824889.2021.1936817)

To link to this article: <https://doi.org/10.1080/18824889.2021.1936817>



© 2021 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group



Published online: 14 Jul 2021.



Submit your article to this journal [↗](#)



Article views: 675



View related articles [↗](#)



View Crossmark data [↗](#)

Value iteration with deep neural networks for optimal control of input-affine nonlinear systems

Hirofumi Beppu ^{a,b}, Ichiro Maruta ^a and Kenji Fujimoto ^a

^aDepartment of Aeronautics and Astronautics, Graduate School of Engineering, Kyoto University, Kyoto Japan; ^bJapan Society for the Promotion of Science, Tokyo, Japan

ABSTRACT

This paper proposes a new algorithm with deep neural networks to solve optimal control problems for continuous-time input nonlinear systems based on a value iteration algorithm. The proposed algorithm applies the networks to approximating the value functions and control inputs in the iterations. Consequently, the partial differential equations of the original algorithm reduce to the optimization problems for the parameters of the networks. Although the conventional algorithm can obtain the optimal control with iterative computations, each of the computations needs to be completed precisely, and it is hard to achieve sufficient precision in practice. Instead, the proposed method provides a practical method using deep neural networks and overcomes the difficulty based on a property of the networks, under which our convergence analysis shows that the proposed algorithm can achieve the minimum of the value function and the corresponding optimal controller. The effectiveness of the proposed method even with reasonable computational resources is demonstrated in two numerical simulations.

ARTICLE HISTORY

Received 30 October 2020
Accepted 12 March 2021

KEYWORDS



Value iteration; optimal control; deep neural networks; input-affine nonlinear systems; convergence analysis

1. Introduction

Optimal control problems [1–3] for nonlinear systems are generally challenging since it requires to solve nonlinear partial differential equations called Hamilton–Jacobi–Bellman (HJB) equations. While those still remain as open problems, approximate dynamic programming (ADP) [4,5] has been one of the successful methods to solve such problems in many areas, e.g. process control [6] and vehicle routing problems [7]. In this method, instead of calculating all the necessary values in time and state spaces which causes a serious computational complexity known as the *curse of dimensionality* [5], appropriate approximations are searched for the value functions and controllers. Value iteration (VI) is one of the ADP-based algorithms, which achieves optimal control by iterative computing from a simple initial value function. The main interest of VI algorithms has lain in discrete-time (DT) systems [8,9] though a number of physical systems desired to be controlled are continuous-time (CT) systems. In this case, we cannot utilize the results of DT systems directly for CT systems since the relationships between them which are especially involved in the important properties of control are not trivial. For avoiding the non-trivial problems and extending the scope of application of the VI algorithm, several works have made efforts to develop new VI-based algorithms for CT systems recently [10,11].

Even though the effectiveness of the VI algorithm has been demonstrated in a lot of situations, the practical forms of the algorithm with theoretical guarantees have not been established yet. This is due to the regulation where each step of the algorithm is proceeded precisely. This requirement is not easy to meet because the preciseness depends on the approximations or the classes of functions for each of the step which cannot be known in advance inherently. In [8], the approximation of value functions is proposed based on linear basis expansion, where the designer has to prepare suitable handmade basis functions while meeting some other requirements of regulations. To broaden the class of functions for approximations, the value functions are approximated with kernel functions in [11], where the convergence of the proposed algorithm is guaranteed successfully. The kernel functions have also been studied for data-driven optimal control problems [12] recently. However, the computational cost with kernel functions increases cubically depending on the number of data [13] for representing value function, so that it is desired to use some techniques [14,15] together to decrease the computational burden.

Deep neural networks (DNNs) [16] are expected to solve both the above problems of approximations and computational costs, which have attracted attention in many fields including reinforcement learning [17] in recent years because of the higher level of the generality

CONTACT Hirofumi Beppu  beppu.hirofumi.36a@st.kyoto-u.ac.jp; hirofumi.beppy@gmail.com  Department of Aeronautics and Astronautics, Graduate School of Engineering, Kyoto University, Kyoto 615-8540, Japan

and fitting ability. In neural networks (NNs), the performances of activation functions have been researched a lot as well [18–20]. According to the results, designers can pick up activation functions from commonly used ones depending on the requirements, so that they do not have to make any basis functions satisfying constraints carefully. This benefit relieves us from burdensome tasks which directly connect to the quality of results. In addition, the update of the parameters for the networks demands the computational cost whose size is at most around the number of data. The recent research also verifies that under a rational assumption standard stochastic gradient descent (SGD) methods can find the global minima of loss functions for DNNs [21]. This result is fundamental to guarantee the convergence of our algorithm based on DNNs.

In this paper, we propose Deep Value Iteration (DVI) algorithm for CT input-affine nonlinear systems for solving the optimal control problems. The proposed method applies DNNs to modelling the value functions and control inputs of each iteration. By doing so, the procedures of the original VI algorithm reduce to the computations of the parameter optimization to minimize a loss function which are set to be the square errors between the values of data points and of the outputs of DNNs. Its convergence to the optimal cost function and optimal control input is proved under the assumption to gain the global minima of loss functions with the universal approximation theorem [22].

This paper is organized as follows. In Section 2, the optimal control problem for nonlinear systems is formulated and the relevant results are shown. The VI algorithm is also introduced. In Section 3, we provide the proposed algorithm followed by its convergence analysis. In Section 4, the effectiveness of the DVI is demonstrated practically using a nonlinear scalar system and an inverted pendulum. In Section 5, we conclude this paper.

The notations used in this paper are defined as follows.

- \mathbb{S}_{++}^n : the set of $n \times n$ positive-definite matrices.
- $\text{diag}(a)$: the $n \times n$ diagonal matrix whose diagonal components are the components of $a \in \mathbb{R}^n$.

2. Preliminaries

2.1. Optimal control problem for CT input-affine nonlinear systems

Consider a CT nonlinear input-affine system as follows.

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t), \quad (1)$$

where $x(t) \in \mathbb{R}^n$ is a state, $u(t) \in \mathbb{R}^m$ is a control input, and $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ and $g: \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$ are functions which characterize the dynamics. We assume that $x = 0$ is an equilibrium which satisfies $f(x) = 0$. It is

also supposed that the system (1) is stabilizable on a compact set $\mathcal{X} \subset \mathbb{R}^n$. For $t \geq 0$, we define the solution to the equation (1) with an initial state $x_0 := x(0)$ and an input u as

$$\varphi(t; x_0, u) := x_0 + \int_0^t f(x(\tau)) + g(x(\tau))u(\tau) d\tau. \quad (2)$$

If the argument is obvious, the notation in (2) is abbreviated as $\varphi(t)$ occasionally. Then suppose that there exist control inputs u such that for all $x_0 \in \mathcal{X}$,

$$\lim_{t \rightarrow \infty} \varphi(t; x_0, u) = 0$$

holds. Our goal is to acquire a controller that minimizes the following cost function.

$$\begin{aligned} J(x_0, u) &= \int_0^\infty q(\varphi(t; x_0, u)) + u(t)^T R u(t) dt \\ &=: \int_0^\infty L(\varphi(t), u(t)) dt, \quad \forall x_0 \in \mathcal{X} \end{aligned} \quad (3)$$

where $q: \mathbb{R}^n \rightarrow \mathbb{R}$ is a positive-definite function and $R \in \mathbb{S}_{++}^m$. The controller u is supposed to stabilize the system (1), to be continuous with $u = 0$ at $x = 0$, and to guarantee the boundedness of value function in (3). We define this class of controllers as an admissible controller. The value function in (3) is assumed to be continuously differentiable. According to dynamic programming [3], the minimum of (3) with respect to u satisfies

$$\begin{aligned} V^*(x_0) &= \min_u \left(\int_0^{\Delta t} L(\varphi(t), u(t)) dt \right. \\ &\quad \left. + V^*(\varphi(\Delta t; x_0, u)) \right), \end{aligned} \quad (4)$$

which is called the HJB equation. Here V^* is time-invariant and $\Delta t > 0$ is an arbitrary constant. When $\Delta t \rightarrow 0$, the optimal feedback controller

$$u^*(x) = -\frac{1}{2} R^{-1} g(x)^T \frac{\partial V^*(x)}{\partial x} \quad (5)$$

is obtained. Substituting (5) into (4) as $\Delta t \rightarrow 0$, the HJB equation (4) is rewritten as a nonlinear partial differential equation expressed as

$$\frac{\partial V^*}{\partial x} f - \frac{1}{4} \frac{\partial V^*}{\partial x} g R^{-1} g^T \frac{\partial V^*}{\partial x} + q(x) = 0. \quad (6)$$

For linear systems, the equation in (6) can be solved efficiently with well-established algorithms. In general cases of nonlinear systems, however, it is difficult to solve (6) analytically.

2.2. Value iteration algorithm

Before we give a new algorithm, the Value iteration (VI) algorithm for CT nonlinear systems proposed in [11]

Algorithm 1 Value Iteration

-
- 1: Initialize with $V^0(x) := 0, \forall x \in \mathcal{X}$
 - 2: Given an arbitrary Δt and set $i = 0$
 - 3: **repeat**
 - 4: $u^i \leftarrow \arg \min_u \left(\int_0^{\Delta t} L(\varphi(t), u(t)) dt \right.$
 - + $V^i(\varphi(\Delta t; x_0, u))$
 - 5: $V^{i+1} : x_0 \mapsto \int_0^{\Delta t} L(\varphi(t), u^i(t)) dt +$
 - $V^i(\varphi(\Delta t; x_0, u^i))$
 - 6: $i \leftarrow i + 1$
 - 7: **until** convergence
-

to solve (6) is described in this section. The recursive computation is based on (4), where finding the best controller for the value function and updating the current value function based on the controller are repeated. This algorithm is shown in Algorithm 1. Although this algorithm assumes that the value function can be expressed analytically, it is not realistic to expect that every V^{i+1} in Algorithm 1 can be obtained perfectly since solving nonlinear partial equations is required to get the value function. It is noted that the VI algorithm is a conceptual algorithm based on which [11] also proposed the practical algorithm using kernel functions whose computational cost increases cubically depending on the number of data points due to the calculations of inverse matrices. This motivates us to develop the improved method to approximate the function V^{i+1} in Section 3.

Note that the superscript of control inputs or value functions denotes the iteration number. The size of Δt can be chosen from $0 < \Delta t < \infty$ arbitrarily, but as Δt is approaching to infinity, the task to find the argument of minimum for u^i is getting as difficult as that to find the solution of (4) directly, so Δt should be smaller preferably. In particular, since in the limit as Δt goes to 0,

$$\begin{aligned} & \arg \min_u \left(\int_0^{\Delta t} L(\varphi(t), u(t)) dt + V^i(\varphi(\Delta t; x_0, u)) \right) \\ &= -\frac{1}{2} R^{-1} g(x)^T \frac{\partial V^i}{\partial x}^T, \end{aligned} \quad (7)$$

it is practical to use (7) approximately by setting Δt as small as possible. Under the assumption that Δt can be small arbitrarily, the convergence of Algorithm 1 is proved in [11]. In the following discussion, Δt is small enough to satisfy (7).

3. Main results

3.1. Deep value iteration algorithm

In this section, we will show our proposed algorithm. The function V^i is modelled as fully connected DNNs,

Algorithm 2 Deep Value Iteration

-
- 1: Initialize with $w^0 := 0$
 - 2: Given an arbitrary Δt and set $i = 0$
 - 3: **repeat**
 - 4: $u_{w^i}(x) \leftarrow -\frac{1}{2} R^{-1} g(x)^T \frac{\partial V_{w^i}(x)}{\partial x}^T$
 - 5: $w^{i+1} \leftarrow w^i$
 - 6: **repeat**
 - 7: Update w^{i+1} by descending the stochastic gradient of $\mathcal{L}(w^i, w^{i+1})$ in (8).
 - 8: **until** convergence
 - 9: $i \leftarrow i + 1$
 - 10: **until** convergence
-

which is expressed as $V_{w^i}(x) := V(x; w^i)$ in the following. Here w^i is a parameter of the function V_{w^i} for each iteration and V_{w^i} is assumed to be over-parametrized, which means that the number of hidden nodes can be sufficiently large compared with the input dimension or the number of training data [23]. The loss function is defined as follows.

$$\mathcal{L}(w^i, w^{i+1}) := l(\epsilon(x_0, w^i, w^{i+1})), \quad (8)$$

where $l: \mathbb{R} \rightarrow \mathbb{R}$ is a positive-definite function and $\epsilon(x_0, w^i, w^{i+1})$ is defined as

$$\begin{aligned} \epsilon(x_0, w^i, w^{i+1}) &:= V_{w^{i+1}}(x_0) - \int_0^{\Delta t} L(\varphi(t), u_{w^i}(t)) dt \\ &\quad - V_{w^i}(\varphi(\Delta t; x_0, u_{w^i})), \end{aligned} \quad (9)$$

and $u_{w^i}(x) := u(x; w^i)$ is defined as

$$u_{w^i}(x) = -\frac{1}{2} R^{-1} g(x)^T \frac{\partial V_{w^i}(x)}{\partial x}^T. \quad (10)$$

The proposed algorithm, Deep Value Iteration (DVI), is described in Algorithm 2. In the updating steps 6–8, which learn the parameters w^{i+1} of DNNs, training data are collected by picking states from \mathcal{X} and computing the values $\epsilon(x_0, w^i, w^{i+1})$ on each of the states to calculate the values of (8). Note that the initial setting of parameter is $w^0 = 0$, so that $V_{w^0} \equiv 0$. It is also noted that the computational cost of the DVI algorithm mainly depends on the SGD¹. Although the cost also depends on which variants of the SGD one uses, the general cost for one iteration of optimization is $O(1)$, i.e. constant. This means that the cost does not depend on the size of training data sets. This is very effective especially when big sizes of data sets are required. For this reason, the proposed algorithm is expected to achieve the lower computational cost compared with the kernel function-based algorithm of [11].

3.2. Convergence analysis for DVI algorithm

The recent study showed that the standard stochastic optimization methods such as SGD can find the global

minima of the loss function with polynomial time [21]. Based on this fact, we give the following assumption to Algorithm 2.

Assumption 3.1: For all $i \geq 0$, let V_{w^i} be over-parametrized by DNNs. Then V_{w^i} and $V_{w^{i+1}}$ can achieve zero for the loss function $\mathcal{L}(w^i, w^{i+1})$ in (8) with certain parameters. Furthermore, it is possible to obtain such parameters for the global minima of the loss function, i.e. zero with standard stochastic gradient methods, e.g. SGD.

Note that in terms of the application, this assumption does not completely hold since the numerical error is unavoidable in general situations. Still, if a network with a large enough structure is prepared, the errors will be negligible. In such situations, the validity of the assumption will be ensured approximately.

The next lemma, called the universal approximation theorem, guarantees that NNs can approximate any continuous functions with arbitrary accuracy.

Lemma 3.2 (Universal approximation theorem [22]): For any continuous functions $C(x)$ defined on a compact set \mathcal{X} , there exists a single hidden layer feed-forward NN, that approximates $C(x)$ and its gradient with an arbitrarily small error.

In the following, we derive a theorem which guarantees the convergence of Algorithm 1 by using Assumption 3.1 and Lemma 3.2 while showing some lemmas necessary to prove the theorem.

Lemma 3.3: Let Assumption 3.1 hold. Suppose that for all $i \geq 0$, V_{w^i} is over-parametrized with DNNs and is updated by Algorithm 2, where u_{w^i} is defined in (10). Then,

$$V_{w^{i+1}}(x_0) = \min_u \left(\int_0^{\Delta t} L(\varphi(t), u(t)) dt + V_{w^i}(\varphi(\Delta t; x_0, u)) \right), \quad \forall x_0 \in \mathcal{X} \quad (11)$$

holds for $i = 0, 1, \dots$

Proof: Since Δt satisfies (7), for any $i \geq 0$,

$$\begin{aligned} u_{w^i}(x) &= -\frac{1}{2}R^{-1}g(x)^T \frac{\partial V_{w^i}(x)}{\partial x} \\ &= \arg \min_u \left(\int_0^{\Delta t} L(\varphi(t), u(t)) dt + V_{w^i}(\varphi(\Delta t; x_0, u)) \right). \end{aligned} \quad (12)$$

It also follows from Assumption 3.1 and Lemma 3.2 that $\mathcal{L}(w^i, w^{i+1}) = 0$, that is, by the steps 6–8 of Algorithm 2

we can find w^{i+1} satisfying

$$\begin{aligned} V_{w^{i+1}}(x_0) &= \int_0^{\Delta t} L(\varphi(t), u_{w^i}(t)) dt \\ &\quad + V_{w^i}(\varphi(\Delta t; x_0, u_{w^i})), \quad \forall x_0 \in \mathcal{X}. \end{aligned} \quad (13)$$

Thus, it is shown that (11) holds by (12) and (13). ■

Lemma 3.4: Suppose that for all $i \geq 0$, V_{w^i} is over-parametrized with DNNs and is updated by Algorithm 2, where u_{w^i} is defined in (10). Let Assumption 3.1 hold and arbitrary controllers be a_0, a_1, \dots . Define $V_{w_a^i}$ as the same DNNs as V_{w^i} . Then, according to Assumption 3.1 and Lemma 3.3, there exists a parameter w_a^{i+1} which defines

$$\begin{aligned} V_{w_a^{i+1}}(x_0) &:= \int_0^{\Delta t} L(\varphi(t), a^i(t)) dt \\ &\quad + V_{w_a^i}(\varphi(\Delta t; x_0, a^i)), \quad \forall x_0 \in \mathcal{X}. \end{aligned} \quad (14)$$

If $w^0 = w_a^0 = 0$, then $V_{w^i}(x_0) \leq V_{w_a^i}(x_0)$ holds for all $i \geq 0$.

Proof: Because $w^0 = w_a^0 = 0$, it follows that $V_{w^0}(x_0) = V_{w_a^0}(x_0) = 0$.

Next, let us deal with the cases of $i \geq 1$. When $i = 1$, it follows from (13) that

$$\begin{aligned} V_{w^1}(x_0) &= \int_0^{\Delta t} L(\varphi(t), u_{w^0}(t)) dt \\ &\quad + V_{w^0}(\varphi(\Delta t; x_0, u_{w^0})) \\ &= \int_0^{\Delta t} L(\varphi(t), u_{w^0}(t)) dt. \end{aligned} \quad (15)$$

Since by Lemma 3.3, the right-hand side of (15) is minimized with respect to the control input, it follows that

$$V_{w^1}(x_0) \leq \int_0^{\Delta t} L(\varphi(t), u_{w_a^0}(t)) dt = V_{w_a^1}(x_0).$$

Now we assume that $V_{w^i}(x_0) \leq V_{w_a^i}(x_0), \forall x_0 \in \mathcal{X}$ holds for $i \geq 1$. Then it follows from this assumption and Lemma 3.3 that

$$\begin{aligned} V_{w^{i+1}}(x_0) &= \int_0^{\Delta t} L(\varphi(t), u_{w^i}(t)) dt \\ &\quad + V_{w^i}(\varphi(\Delta t; x_0, u_{w^i})) \\ &\leq \int_0^{\Delta t} L(\varphi(t), a^i(t)) dt \\ &\quad + V_{w^i}(\varphi(\Delta t; x_0, a^i)) \\ &\leq \int_0^{\Delta t} L(\varphi(t), a^i(t)) dt \\ &\quad + V_{w_a^i}(\varphi(\Delta t; x_0, a^i)) \\ &= V_{w_a^{i+1}}(x_0), \end{aligned}$$

where the second and third inequalities follow from Lemma 3.3 and the assumption $V_{w^i}(x_0) \leq V_{w_a^i}(x_0), \forall x_0$

$\in \mathcal{X}$, respectively. Therefore by mathematical induction, it is proved that $V_{w^i}(x_0) \leq V_{w_a^i}(x_0), \forall x_0 \in \mathcal{X}$ holds for all $i \geq 0$. ■

Lemma 3.5: *Let Assumption 3.1 hold. Suppose that for all $i \geq 0$, V_{w^i} is over-parametrized with DNNs and is updated by Algorithm 2, where u_{w^i} is defined in (10). If $w^0 = 0$ and the system in (1) is controllable, then there exists an upper bound U such that for all $i \geq 0$ and $x_0 \in \mathcal{X}$, $V_{w^i}(x_0) \leq U(x_0)$ holds. In addition, if the HJB equation in (6) has a unique solution V^* , it is the lower bound of U , i.e., $V^*(x_0) \leq U(x_0)$ for all $x_0 \in \mathcal{X}$ and it also satisfies $V_{w^i}(x_0) \leq V^*(x_0) \leq U(x_0)$ for all $i \geq 0$.*

Proof: Let s be any admissible controller and suppose that $V_{w_s^i}$ is parametrized by DNNs in the same way as V_{w^i} . Then because of Assumption 3.1 and Lemma 3.3, there exists a parameter w_s^{i+1} which defines

$$V_{w_s^{i+1}}(x_0) := \int_0^{\Delta t} L(\varphi(t), s(t)) dt + V_{w_s^i}(\varphi(\Delta t; x_0, s)), \quad \forall x_0 \in \mathcal{X}, \quad (16)$$

where we suppose that $w_s^0 = 0$ and $V_{w_s^0} = 0$. Subtracting $V_{w_s^i}$ from $V_{w_s^{i+1}}$ for $i \geq 0$, it follows from (16) that

$$\begin{aligned} & V_{w_s^{i+1}}(x_0) - V_{w_s^i}(x_0) \\ &= V_{w_s^i}(\varphi(\Delta t; x_0, s)) - V_{w_s^{i-1}}(\varphi(\Delta t; x_0, s)) \\ &= V_{w_s^{i-1}}(\varphi(2\Delta t; x_0, s)) - V_{w_s^{i-2}}(\varphi(2\Delta t; x_0, s)) \\ &\quad \vdots \\ &= V_{w_s^1}(\varphi(i\Delta t; x_0, s)) - V_{w_s^0}(\varphi((i-1)\Delta t; x_0, s)) \\ &= V_{w_s^1}(\varphi(i\Delta t; x_0, s)), \end{aligned}$$

which is a recurrence relation. From this relation, it follows that

$$\begin{aligned} V_{w_s^{i+1}}(x_0) - V_{w_s^i}(x_0) &= V_{w_s^1}(\varphi(i\Delta t; x_0, s)) \\ V_{w_s^i}(x_0) - V_{w_s^{i-1}}(x_0) &= V_{w_s^1}(\varphi((i-1)\Delta t; x_0, s)) \\ &\quad \vdots \\ V_{w_s^1}(x_0) - V_{w_s^0}(x_0) &= V_{w_s^1}(x_0). \end{aligned} \quad (17)$$

Using the fact that $V_{w_s^0}(x_0) = 0, \forall x_0 \in \mathcal{X}$ and adding all the both sides of (17), it follows that

$$\begin{aligned} V_{w_s^{i+1}}(x_0) &= \sum_{n^i=1}^i V_{w_s^1}(\varphi(n^i \Delta t; x_0, s)) \\ &\leq \sum_{n^i=0}^{\infty} V_{w_s^1}(\varphi(n^i \Delta t; x_0, s)) \\ &= \int_0^{\infty} L(\varphi(t), s(t)) dt. \end{aligned} \quad (18)$$

Here s is an admissible controller, so that the right-hand side of (18) is bounded. Defining this upper bound as

$U(x_0)$,

$$V_{w_s^{i+1}}(x_0) \leq \int_0^{\infty} L(\varphi(t), s(t)) dt =: U(x_0) \quad (19)$$

holds. Since $V_{w_s^0}(x_0) = 0, V_{w_s^i}(x_0) \leq U(x_0), \forall x_0 \in \mathcal{X}$ holds for all $i \geq 0$. Setting $a_i = s, \forall i \geq 0$ leads to that $V_{w^i}(x_0) \leq V_{w_a^i} = V_{w_s^i} \leq U(x_0), \forall x_0 \in \mathcal{X}$ for all $i \geq 0$.

Now u^* is a stabilizing and admissible controller which minimizes (3), from which it follows that

$$\begin{aligned} V^*(x_0) &= \int_0^{\infty} L(\varphi(t), u^*(t)) dt \\ &\leq \int_0^{\infty} L(\varphi(t), s(t)) dt = U(x_0). \end{aligned} \quad (20)$$

From (20), $V^*(x_0)$ is clearly a lower bound of $U(x_0)$. Setting $s = u^*$, it follows that

$$V_{w^i}(x_0) \leq V_{w_s^i}(x_0) = V^*(x_0) \leq U(x_0).$$

Thus it is shown that for all $i \geq 0, V_{w^i}(x_0) \leq V^*(x_0) \leq U(x_0), \forall x_0 \in \mathcal{X}$ holds. ■

Now we can show the convergence of Algorithm 2 by all the lemmas proven above.

Theorem 3.6: *Let Assumption 3.1 hold. Suppose that for all $i \geq 0, V_{w^i}$ is over-parametrized with DNNs and is updated by Algorithm 2, where u_{w^i} is defined in (10). If $w^0 = 0, V_{w^i}(x_0) \leq V_{w^{i+1}}(x_0)$ holds for all $i \geq 0$. Moreover, $\lim_{i \rightarrow \infty} V_{w^i} = V^*$ and $\lim_{i \rightarrow \infty} u_{w^i} = u^*$ hold, where V^* is the solution to (6) and u^* is the optimal controller.*

Proof: Let a_i and $V_{w_a^{i+1}}$ be defined in the same way as in Lemma 3.4. If $w^0 = w_a^0 = 0$, it follows from Lemma 3.4 that $V_{w^i}(x_0) \leq V_{w_a^i}(x_0), \forall x_0 \in \mathcal{X}$ for all $i \geq 0$. Then choosing the arbitrary controller as $a_i = u_{w^{i+1}}$, there exists a parameter w_a^{i+1} such that satisfies

$$\begin{aligned} V_{w_a^{i+1}}(x_0) &= \int_0^{\Delta t} L(\varphi(t), u_{w^{i+1}}(t)) dt \\ &\quad + V_{w_a^i}(\varphi(\Delta t; x_0, u_{w^{i+1}})), \quad \forall x_0 \in \mathcal{X}. \end{aligned}$$

Since $V_{w^0}(x_0) = V_{w_a^0}(x_0) = 0$, subtracting $V_{w_a^0}(x_0)$ from $V_{w^1}(x_0)$,

$$V_{w^1}(x_0) - V_{w_a^0}(x_0) = \int_0^{\Delta t} L(\varphi(t), u_{w^0}(t)) dt \geq 0$$

holds. Now we assume that $V_{w_a^{i-1}}(x_0) \leq V_{w^i}(x_0), \forall x_0 \in \mathcal{X}$ holds for all $i \geq 1$. Then,

$$\begin{aligned} V_{w^{i+1}} - V_{w_a^i} &= V_{w^i}(\varphi(\Delta t; x_0, u_{w^i})) \\ &\quad - V_{w_a^{i-1}}(\varphi(\Delta t; x_0, u_{w^i})) \geq 0, \end{aligned}$$

so that $V_{w_a^i}(x_0) \leq V_{w^{i+1}}(x_0)$ holds as well. Therefore, by induction, it is shown that $V_{w_a^i}(x_0) \leq V_{w^{i+1}}(x_0), \forall x_0 \in$

\mathcal{X} holds for all $i \geq 0$. Recalling the fact that $V_{w^i}(x_0) \leq V_{w^i}(x_0), \forall x_0 \in \mathcal{X}$, it is proved that $V_{w^i}(x_0) \leq V_{w^{i+1}}(x_0), \forall x_0 \in \mathcal{X}$.

Next, we investigate the property of V_{w^∞} . In $i \rightarrow \infty$, it follows from (13) that

$$\begin{aligned} & V_{w^\infty}(\varphi(\Delta t; x_0, u_{w^\infty})) - V_{w^\infty}(x_0) \\ &= - \int_0^{\Delta t} L(\varphi(t), u_{w^\infty}(t)) dt \leq 0. \end{aligned}$$

Using Lemma 3.5, we can see that $V_{w^\infty}(x_0) \leq V^*(x_0) \leq U(x_0), \forall x_0 \in \mathcal{X}$ and $V_{w^\infty}(x_0)$ is bounded for all $x_0 \in \mathcal{X}$. Thus, it is verified that u_{w^∞} is a stabilizing admissible controller and V_{w^∞} is a candidate of a Lyapunov function and of a solution to the HJB equation (6). Furthermore, based on the fact that $V_{w^i}(x_0) \leq V_{w^\infty}(x_0), \forall x_0 \in \mathcal{X}$ for all $i \geq 0$, V_{w^∞} is the upper bound in Lemma 3.5, that is, $V^*(x_0) \leq U(x_0) = V_{w^\infty}(x_0), \forall x_0 \in \mathcal{X}$. From these reasons, it follows that $V^*(x_0) \leq V_{w^\infty}(x_0) \leq V^*(x_0), \forall x_0 \in \mathcal{X}$. Finally, it is proven that $\lim_{i \rightarrow \infty} V_{w^i} = V^*$ and $\lim_{i \rightarrow \infty} u_{w^i} = u^*$. ■

4. Numerical example

In this section, the proposed method is applied to two optimal control problems of a scalar nonlinear polynomial system and an inverted pendulum system by using practical computational resources. For the optimization tool, PyTorch [24] is used in the examples. In the following examples, the loss function, called the Huber loss [25],

$$\begin{aligned} & l(\epsilon(x_0, w^i, w^{i+1})) \\ &:= \begin{cases} \mathbb{E}_{x_0 \in \mathcal{X}} \left[\frac{1}{2} \epsilon(x_0, w^i, w^{i+1})^2 \right], & \text{for } |\epsilon| \leq 1, \\ \mathbb{E}_{x_0 \in \mathcal{X}} \left[|\epsilon(x_0, w^i, w^{i+1})| - \frac{1}{2} \right], & \text{otherwise,} \end{cases} \end{aligned}$$

is used since the error values in our problem settings can be wide and the Huber loss can deal with such situations by using the square errors for smaller errors while utilizing the absolute errors for bigger errors.

4.1. Scalar nonlinear system

Consider the nonlinear polynomial system as follows [26].

$$\dot{x} = 0.01x^2 + u.$$

We set

$$\begin{aligned} J(x_0, u) &= \int_0^\infty (0.01\varphi(t; x_0, u)^4 \\ &\quad + 0.01\varphi(t; x_0, u)^2 + u(t)^2) dt \end{aligned}$$

as a cost function. It is known that the exact optimal cost of this problem setting is given as $V^*(x) = (x^3/150) + ((101x^2 + 100)^{3/2}/15150) - 20/303$ [26].

For the setting of the DVI algorithm, $\Delta t = 0.5$ (s). In order to carry out the steps 6–8 of Algorithm 2, we get the training data set from 201 equally spaced points on a region $[-2.5, 2.5]$ by computing the values $\epsilon(x_0, w^i, w^{i+1})$ in (9) on each of the points. For updating each w^i , Adam algorithm [27], one of SGD algorithms, is employed. Each iteration of the update is set as 1001 times which are empirically expected to let the gradient descent converge in this case. The DNNs are equipped with an input layer, an output layer, and three hidden layers, each of which has 50 nodes. The number of the weight parameters between the input layer and the first hidden layer is same as the number of the edges between them, so the number is $2 \times 50 = 100$. Considering the structure of the DNNs, the total number of the weight parameters is 5150. On the other hand, the number of bias parameters between the input layer and the first hidden layer is 50 since each of the bias parameters is added to each of the nodes of the first hidden layer. This means that the total number of bias parameters of the DNNs is 151. Thus, the total number of parameters, i.e. the dimension of each w^i is 5301. Note that there is no general design policy for the DNNs currently, so the relatively bigger size of the network which may not be optimal is adopted here for higher abilities to express the value function in iterations. For the activation function, we choose a hyperbolic tangent function, which can satisfy the requirements of the differentiability of value function.

The DVI algorithm converged after 29 iterations. The comparison between the value function $V_{w^{29}}$ obtained in DVI algorithm and the true optimal cost function is shown in Figure 1, where the value $V_{w^{29}}(0)$ is subtracted to offset the numerical error at the origin. It is obvious that the proposed method achieves the significantly close solution to the true one.

4.2. Inverted pendulum

Consider the CT pendulum system illustrated in Figure 2 with $x = [\theta, \dot{\theta}]^T$ as

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ &= \begin{pmatrix} \dot{\theta} \\ \frac{mgl}{m^2} \sin \theta \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{ml^2} \end{pmatrix} u, \end{aligned} \quad (21)$$

where θ (rad) and $\dot{\theta}$ (rad/s) are angle and angular velocity of the pendulum, respectively, and $m = 1$ (kg) is the mass of pendulum, $g_r = 9.8$ (m/s²) is the gravitational acceleration, and $l = 1$ (m) is the length of pendulum. The goal is to swing up the pendulum and balance it in the inverted position at $[\theta, \dot{\theta}]^T = 0$ with an arbitrary initial state. We choose

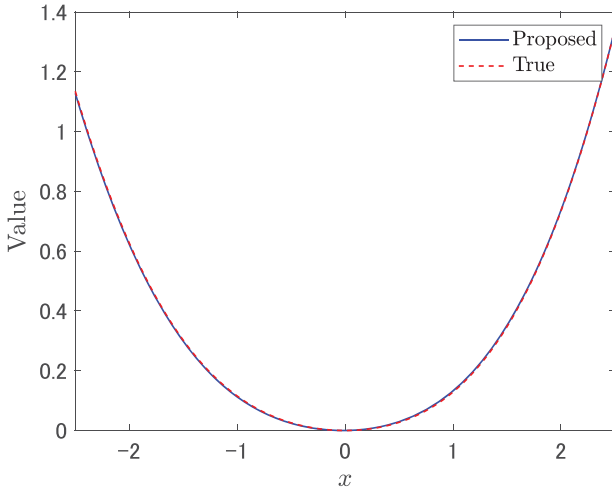


Figure 1. Comparison of the value functions.

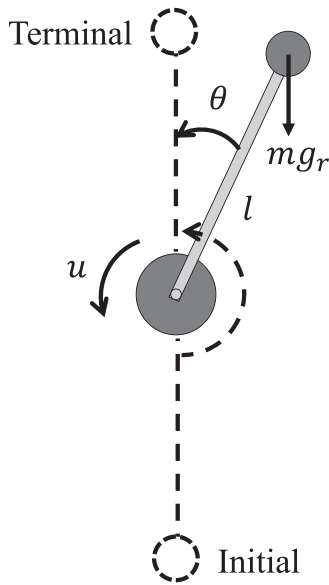


Figure 2. Pendulum.

$$J(x_0, u) = \int_0^{\infty} \varphi(t; x_0, u)^T Q \varphi(t; x_0, u) + u(t)^T R u(t) dt,$$

as a cost function, where $Q = \text{diag}([1, 0.01]^T)$ and $R = 1$.

We set Δt as 0.1 (s) for the DVI algorithm. The data set is picked from 201×201 uniform grids on a region of $[-2\pi, 2\pi] \times [-10, 10]$, on each of which the values $\epsilon(x_0, w^i, w^{i+1})$ in (9) are computed for the steps 6–8 of Algorithm 2. For each w^i , the update by the same algorithm as in the first example is repeated 1001 times as well based on the same reason of the first case. The DNNs are equipped with the same number of layers, where each of hidden layers has 100 nodes. The dimension of each w^i is then 20601. For the activation function, we choose a hyperbolic tangent function here too.

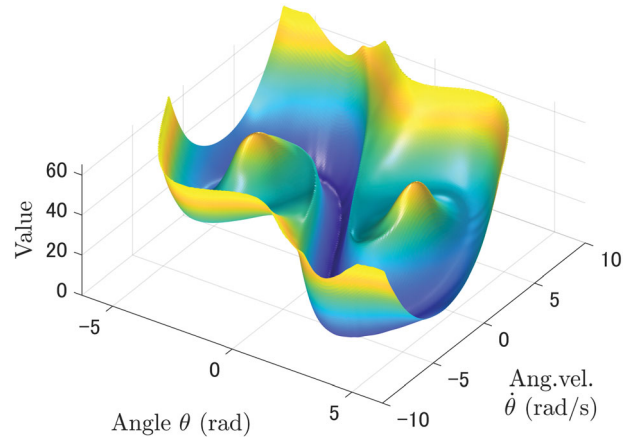


Figure 3. Value function $V_{w^{60}}$ by the DVI algorithm.

Table 1. Ratio of the cost of DVI to that of LQR in (22).

	Average	Minimum	Maximum
Ratio	0.302	0.031	0.994

The iteration is stopped after 60 iterations, and the value function $V_{w^{60}}$ obtained by the proposed algorithm is shown in Figure 3. We give the results of numerical simulation with ten initial states and a terminal state shown as the white squares and circle, respectively, in Figure 5, where the solid lines are the trajectories of the system (21) and the dashed lines are the contours of the value function. For the purpose of comparison, We also show the results of a linear quadratic regulator, LQR, which obtains suboptimal control policies by linearizing the system (21), with the same initial states and terminal state. In Figure 5, we compare the input histories of the proposed method and LQR for each of initial states. Both of the methods succeed to stabilize the pendulum at the position $[0, 0]^T$, but it is seen that the inputs obtained by the DVI algorithm are suppressed overall, whereas the inputs of LQR from some initial states are much bigger. In order to investigate the efficiencies of the proposed method in terms of the cost, we set another cost for 6 (s)

$$J(x_0, u) = \int_0^6 \varphi(t; x_0, u)^T Q \varphi(t; x_0, u) + u(t)^T R u(t) dt, \quad (22)$$

and we compute the ratios of the costs in (22) for the DVI algorithm to those for the LQR in Table 1. This table shows the DVI algorithm could cut the cost by about 70% on average. The minimum and maximum of the ratios are of the initial states $[-6, 5]^T$ and $[-2, 8]^T$, respectively. From this result and Figure 4, we can see that the DVI algorithm is relatively more effective around the areas where the trajectories of the proposed method far from those of the LQR. This also implies that the proposed method could enlarge the cost-wise efficient region compared with the linearization strategy.

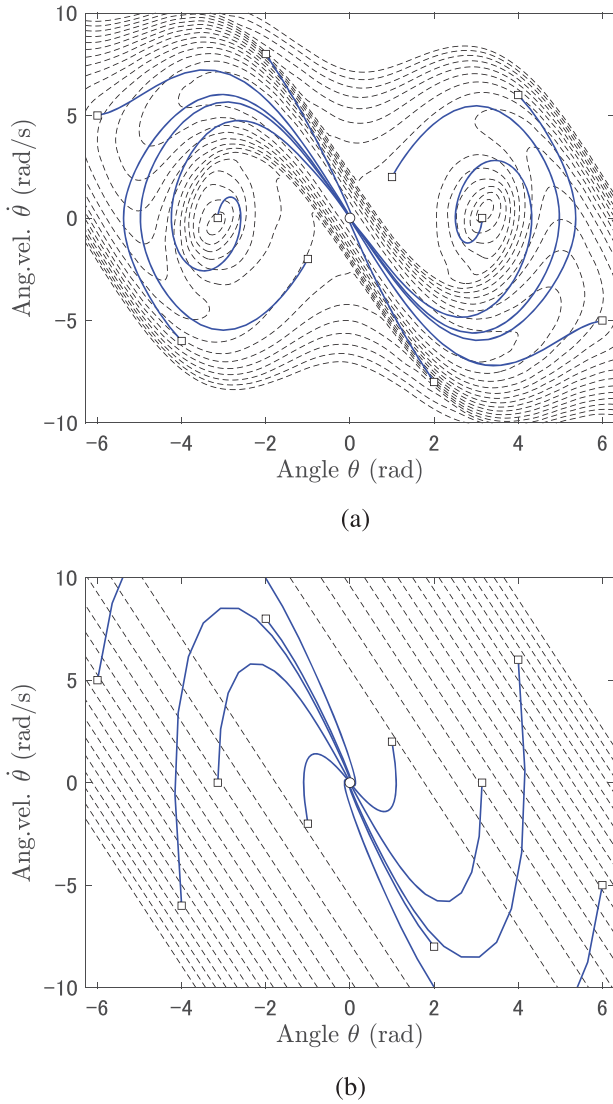


Figure 4. Trajectories of the system (21) from ten different initial states on the contour maps of value functions, (a) DVI algorithm (proposed), (b) LQR.

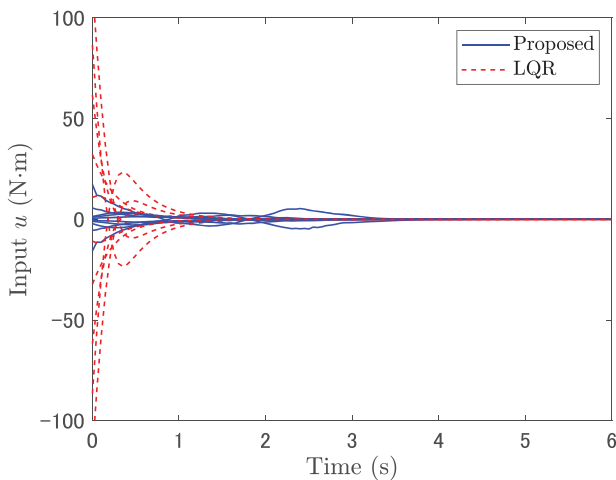


Figure 5. Comparison of the control inputs.

The results are also compared with the kernel function-based algorithm of [11] in terms of the performance and computation time for the same

example. It is reported that the DVI algorithm achieved 55.2 for the cost in (22) with the initial state $[\pi, 0]^T$, which is about 94.6% of the cost of the algorithm of [11]. The DVI algorithm took 6.97 (s) for one iteration of the algorithm, which is measured by taking the average of ten iterations and is about 19.3% of the computation time of the algorithm of [11]. The total number of iterations in the DVI algorithm is also 60% of the algorithm of [11]. Note that the types of parameters of [11] are different from those of the proposed method, so that it is difficult to compare the algorithms fairly. For example, the number of basis functions in [11] is automatically determined by the number of training data, whereas the number of activation functions of the proposed method does not necessarily match with the number of training data. Moreover, their accuracy also depends on what kind of activation functions or kernel functions we choose. Even considering the unfairness, however, the proposed method could achieve better performance with a shorter computation time in this example.

5. Conclusion

In this paper, we have proposed a new value iteration algorithm to obtain optimal control for CT nonlinear input-affine systems with DNNs. Instead of solving the partial differential equations, the DVI algorithm provides us optimization problems for parameters of DNNs. The convergence analysis shows that the proposed algorithm can achieve the optimality using the property of DNNs. We have verified the effectiveness of the algorithm by two numerical simulations of the scalar nonlinear system and the inverted pendulum system under the situations where only reasonable computational resources are provided. It is also confirmed that the DVI algorithm has the potential to achieve the equivalent or a better level of performance with smaller computational costs compared with the kernel function-based algorithm of [11] in the example of the inverted pendulum.

Note

1. For the stochastic gradient, the estimate gradient G is used practically, where it satisfies $\mathbb{E}[G] = (\partial \mathcal{L} / \partial w^{i+1})(w^i, w^{i+1})$. Then, the update rule is $w^{i+1} := w^{i+1} - \alpha G$, where α is some constant.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by JSPS KAKENHI [grant number JP19J23306].

Notes on contributors



Hirofumi Beppu received his B.S. and M.S. degrees in Engineering from Kyoto University, Japan, in 2017 and 2019, respectively. He is currently a Ph.D. student of the Graduate School of Engineering, Kyoto University, Japan. Since 2019, he has been a fellow of the Research Fellowship for Young Scientists (DC1) of the Japan Society for the Promotion of Science.



Ichiro Maruta received the Bachelor of Engineering, Master of Informatics, and Doctor of Informatics degrees from Kyoto University, Kyoto, Japan, in 2006, 2008, and 2011, respectively. He was a research fellow of the Japan Society for the Promotion of Science from 2008 to 2011. From 2012 to 2017, he was an Assistant Professor at the Graduate School of Informatics, Kyoto University. In 2017, he joined the Graduate School of Engineering, Kyoto University, as a Lecturer of the Department of Aeronautics and Astronautics, and since 2019, he has been an Associate Professor.



Kenji Fujimoto received his B.Sc. and M.Sc. degrees in Engineering and Ph.D. degree in Informatics from Kyoto University, Japan, in 1994, 1996, and 2001, respectively. He is currently a professor of Graduate School of Engineering, Kyoto University, Japan. From 1997 to 2004, he was a research associate of Graduate School of Engineering and Graduate School of Informatics, Kyoto University, Japan. From 2004 to 2012, he was an associate professor of Graduate School of Engineering, Nagoya University, Japan. From 1999 to 2000, he was a re-search fellow of Department of Electrical Engineering, Delft University of Technology, The Netherlands. He has held visiting research positions at the Australian National University, Australia and Delft University of Technology, The Netherlands in 1999 and 2002, respectively. His research interests include nonlinear control and stochastic systems theory.

ORCID

Hirofumi Beppu  <http://orcid.org/0000-0002-4368-3863>

Ichiro Maruta  <http://orcid.org/0000-0002-2246-3570>

Kenji Fujimoto  <http://orcid.org/0000-0001-6345-4884>

References

- [1] Bellman R. On the theory of dynamic programming. *Proc Natl Acad Sci USA*. 1952;38(8):716.
- [2] Bellman R. The theory of dynamic programming. *Bull Amer Math Soc*. 1954;60(6):503–515.
- [3] Bellman R. Dynamic programming. *Science*. 1966;153(3731):34–37.
- [4] Sutton RS, Barto AG. Reinforcement learning: an introduction. Cambridge, MA: MIT Press; 1998.
- [5] Powell WB. Approximate dynamic programming: solving the curses of dimensionality. Vol. 703, Hoboken: John Wiley & Sons; 2007.
- [6] Lewis FL, Liu D. Reinforcement learning and approximate dynamic programming for feedback control. Vol. 17. Hoboken: John Wiley & Sons; 2013.
- [7] Novoa C, Storer R. An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *Eur J Oper Res*. 2009;196(2): 509–515.
- [8] Al-Tamimi A, Lewis FL, Abu-Khalaf M. Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof. *IEEE Trans Syst Man Cyber Part B (Cyber)*. 2008;38(4):943–949.
- [9] Wei Q, Liu D, Lin H. Value iteration adaptive dynamic programming for optimal control of discrete-time nonlinear systems. *IEEE Trans Cybern*. 2016;46(3):840–853.
- [10] Wu HN, Luo B. Heuristic dynamic programming algorithm for optimal control design of linear continuous-time hyperbolic pde systems. *Ind Eng Chem Res*. 2012;51(27):9310–9319.
- [11] Beppu H, Maruta I, Fujimoto K. Approximate dynamic programming with gaussian processes for optimal control of continuous-time nonlinear systems. *IFAC-PapersOnLine*, 2020. 21st IFAC World Congress, to appear.
- [12] Ito Y, Fujimoto K, Tadokoro Y. Kernel-based Hamilton–Jacobi equations for data-driven optimal and h-infinity control. *IEEE Access*. 2020;8:131047–131062.
- [13] Rasmussen CE, Williams CKI. Gaussian processes for machine learning (adaptive computation and machine learning). Cambridge, MA: The MIT Press; 2005.
- [14] Snelson E, Ghahramani Z. Sparse Gaussian processes using pseudo-inputs. In: *Advances in neural information processing systems*. Cambridge, MA: MIT Press; 2006; p. 1257–1264.
- [15] Seeger M, Williams C, Lawrence N. Fast forward selection to speed up sparse Gaussian process regression; 2003.
- [16] Goodfellow I, Bengio Y, Courville A. Deep learning. Cambridge, MA: MIT press; 2016.
- [17] Mnih V, Kavukcuoglu K, Silver D, et al., Human-level control through deep reinforcement learning. *Nature*. 2015;518(7540):529–533.
- [18] Hornik K, Stinchcombe M, White H. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks/DIFdel; Neural Netw*. 1990;3(5):551–560.
- [19] Leshno M, Lin VY, Pinkus A, et al., Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Netw*. 1993;6(6):861–867.
- [20] Ramachandran P, Zoph B, Le Q. Searching for activation functions; 2018.
- [21] Allen-Zhu Z, Li Y, Song Z. A convergence theory for deep learning via over-parameterization. In: *Proceedings of the 36th International Conference on Machine Learning*, Vol. 97. *Proceedings of Machine Learning Research*; 2019; p. 242–252.
- [22] Hornik K, Stinchcombe M, White H. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Netw*. 1990;3(5):551–560.
- [23] Du S, Lee J. On the power of over-parametrization in neural networks with quadratic activation. In: *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80. *Proceedings of Machine Learning Research*; 2018; p. 1329–1338.
- [24] Paszke A, Gross S, Massa F, et al., Pytorch: An imperative style, high-performance deep learning library. In: *Advances in neural information processing systems*. Vol. 32. Red Hook: Curran Associates, Inc; 2019; p. 8024–8035.

- [25] Huber PJ. Robust estimation of a location parameter. *Ann Math Statist.* 1964 03;35(1):73–101.
- [26] Jiang Y, Jiang Z. Global adaptive dynamic programming for continuous-time nonlinear systems. *IEEE Trans Automat Contr.* 2015;60(11):2917–2929.
- [27] Kingma DP, Ba J. A method for stochastic optimization. preprint [arXiv:14126980](https://arxiv.org/abs/1412.6980); 2014.