ECE-GY 9223 Reinforcement Learning Spring 2019

Lecture 4 — Feb 19, 2019

Prof. Quanyan Zhu *Scribe: Tao Li*

1 Overview

In the last lecture we discuss about value iteration in computations for MDPs and in this lecture we continue our discussion by introducing **policy iteration**. Besides, a function approximation technique: **neural networks** is also included and we demonstrate an **optimal-control-based approach** for dealing with multilayer neural networks, where **Pontryagin's Maximum Principle** is applied to solve the optimal control problem derived from neural networks.

2 Policy Iteration

Generally, in value iteration, an infinite number of iterations are required to reach the optimal cost-to-go function, whereas policy iteration, an alternative to value iteration, always terminates finitely. In policy iteration, we start with an initial proper policy μ_0 and generate a sequence of policies $\mu_1, \mu_2, \cdots$ by the following scheme:

initialization k=0, pick an arbitrary policy μ_0,

policy evaluation at time k, compute J_{μ_k} by solving the linear system: $J_{\mu_k} = T_{\mu_k}(J_{\mu_k})$,

policy improvement find a better policy μ_{k+1} computed as

$$\mu_{k+1} \in \arg\min_{u \in U(i)} \sum_{j=0}^{n} p_{ij}(u)\left(g(i,u,j) + J_{\mu_k}(i)\right),$$

termination let $k \leftarrow k+1$ and repeat the policy evaluation and policy improvement step until $J_{\mu_{k+1}} = J_{\mu_k}$.

The following remarks establish the validity of policy iteration.

Remark 1. *Policy iteration always terminates in a finite number of steps, since the number of proper policies is finite (the state and action space are finite).*

Remark 2. *The cost improves at each step, i.e.* $J_{\mu_{k+1}} \leq J_{\mu_k}$.

Proof. We first prove that $T_{\mu_{k+1}}(J_{\mu_k}) \leq J_{\mu_k}$. By definition,

$$T_{\mu_{k+1}}(J_{\mu_k})(i) = \sum_{j=0}^{n} p_{ij}(\mu_{k+1}(i))\left(g(i,\mu_{k+1},j) + J_{\mu_k}(j)\right).$$

Since μ_{k+1} is the solution to the minimization problem: $\min_{\mu \in U(i)} \sum_{j=0}^{n} p_{ij}(u)\left(g(i,u,j) + J_{\mu_k}(j)\right)$,

$$\begin{aligned} T_{\mu_{k+1}}(J_{\mu_k})(i) &\leq \sum_{j=0}^{n} p_{ij}(\mu_k)\left(g(i,\mu_k(i),j) + J_{\mu_k}(j)\right) \\ &= T_{\mu_k}(J_{\mu_k})(i) = J_{\mu_k}(i). \end{aligned}$$

Hence, $T_{\mu_{k+1}}(J_{\mu_k}) \leq J_{\mu_k}$. Then, by monotonicity of T_{μ_k},

$$T^n_{\mu_{k+1}}(J_{\mu_k}) \leq J_{\mu_k},$$

which gives $J_{\mu k+1} \leq J_{\mu_k}$ by letting $k \to +\infty$. □

Remark 3. *$J_{\mu_{k+1}} = J_{\mu_k}$ implies that μ_{k+1} is the optimal policy, i.e. policy evaluation terminates at the optimal one.*

Proof. From policy improvement step,

$$T_{\mu_{k+1}} J_{\mu_k} = T J_{\mu_k}.$$

With $J_{\mu_{k+1}} = J_{\mu_k}$ and $J_{\mu_{k+1}} = T_{\mu_{k+1}} J_{\mu_{k+1}}$, we have

$$J_{\mu_{k+1}} = T_{\mu_{k+1}} J_{\mu_{k+1}} = T J_{\mu_k} = T J_{\mu_{k+1}},$$

which implies that $J_{\mu_{k+1}} = J^*$. □

Remark 4. *J^* is the "largest" J that satisfies the constraint $J \leq TJ$: for $J \leq TJ$, with the monotonicity of T, we have*

$$J \leq T^n J,$$

and let $n \to +\infty$, we obtain $J \leq J^*$. More explicitly, $J^*(i)$ is the solution to the following linear programming problem:

$$\begin{aligned} &\textit{maximize } \sum_{i=1}^{n} J(i) \\ &\textit{subject to } J(i) \leq \sum_{j=0}^{n} p_{ij}(u)\left(g(i,u,j) + \alpha J(j)\right), \quad i = 1,\ldots,n, u \in U(i) \end{aligned}$$

Remark 5. *An interesting interpretation of policy iteration is that it behaves like an Actor-Critic System[1](see Fig 1). In this interpretation, the policy evaluation is viewed as the work of critic, evaluating the performance of the current policy, i.e. computing J_{μ_k}, while the policy improvement acts as an actor, choosing an optimal control μ_{k+1} based on the latest evaluation J_{μ_k}.*

3 Neural Networks

As discussed in previous lectures, our goal is to find the optimal cost-to-go function $J(i)^*$ in a MDP problem. However, solving the Bellman that $J(i)^*$ satisfies is usually challenging, since it suffers from the curse of dimensionality. One possible approach to tackle it is to construct an approximate representation $J(i,r) \approx J^*(i)$, where only a few parameters r are needed[1]. Among various approximation methods, neural networks, adopting a compositional approach, has some really encouraging success in practice. In this section, we shall briefly introduce the architecture

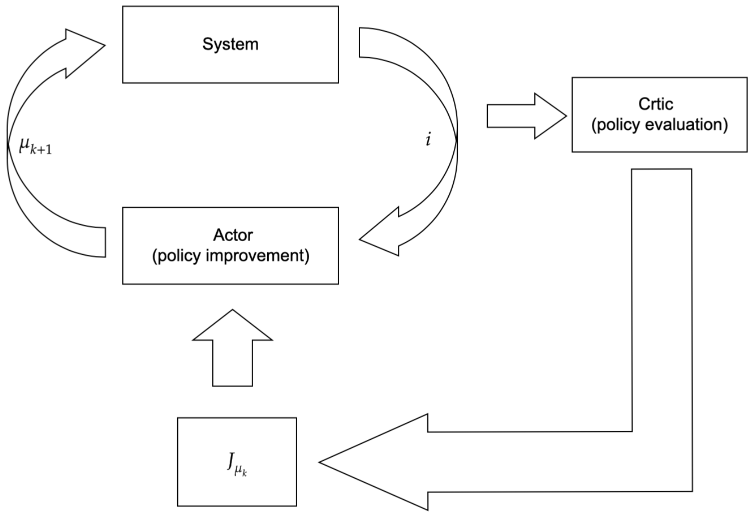


Figure 1: actor-critic system

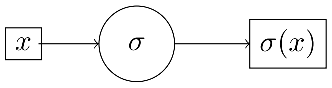


Figure 2: neuron

of neural networks from a viewpoint of optimal control. In addition, Maximum principle is also included for solving the optimal control problem drived from the training process of neural networks.

We start from a basic unit in neural networks: neuron. A neuron is a nonlinear and scalar-valued function $\sigma : \mathbb{R} \to \mathbb{R}$ and the following examples of neurons have been widely used in practice.

Example 1. *sign function*

$$\operatorname{sgn}(x) := \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$$

Example 2. *sigmoid function*

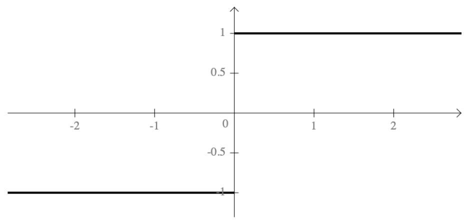
$$f(x) = \frac{1}{1+e^{-x}}$$

Example 3. *Rectifier linear units(ReLu):*

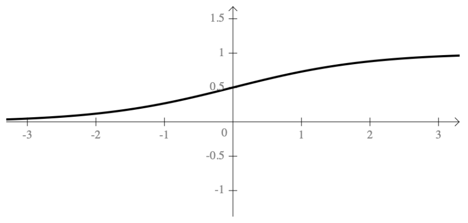
$$R(x) = \max(0, x)$$

Example 4. *Leaky ReLu:*

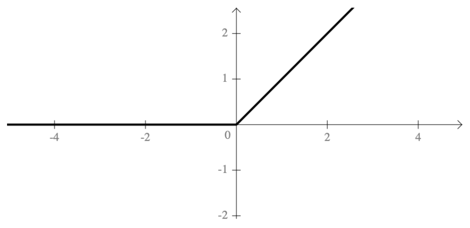
$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$



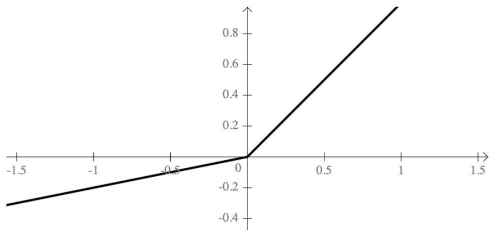
(a) example 1: sign function



(b) example 2: sigmoid function



(c) example 3: ReLu function



(d) example 4: Leaky ReLu function

Figure 3: Four different neurons

3.1 Multilayer Neural Network

The reason why neural networks is, to some extent, superior to traditional approximation techniques, like wavelets and framelets, is that neural networks use compositions of simple functions to approximate complicated ones, i.e., the neural network approach is compositional, whereas classical approximation theory is usually additive[2]. Thus, it is necessary to investigate a multilayer neural network and its mathematical foundation and we shall begin with its building block.

In general case, a neuron can also act on a vector input in a componentwise way, i.e.

$$\sigma\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}\right) = \begin{bmatrix} \sigma(x_1) \\ \sigma(x_2) \\ \vdots \\ \sigma(x_n) \end{bmatrix}$$

A simplified block diagram, the building block of neural networks, is illustrated in Fig 4 and the multilayer neural networks is shown in Fig 5

Remark 6. *The number of neurons at each layer can be different*

Remark 7. *The goal is to choose weights W and b so that $\tilde{f} \approx f$, where f is the given function to be approximated by neural networks and $\tilde{f}$ is the composition of each layer of neurons.*

Remark 8 (viewpoint of dynamical systems)**.** *At layer k, we have*

$$y_k = \sigma(W_k y_{k-1} + b_k), \quad k = 1, \cdots L,$$

where y_0 is the input x and the dimension of W_k, b_k are determined by the number of neurons at

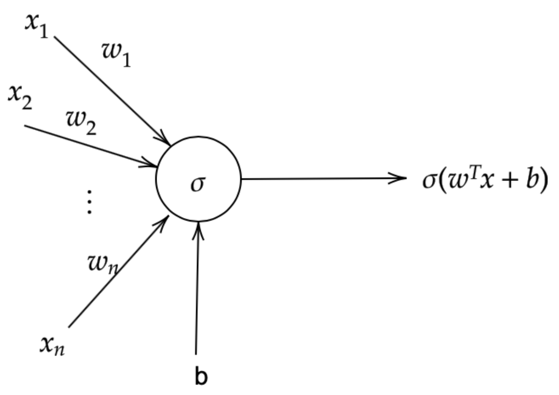


Figure 4: A simplified block diagram

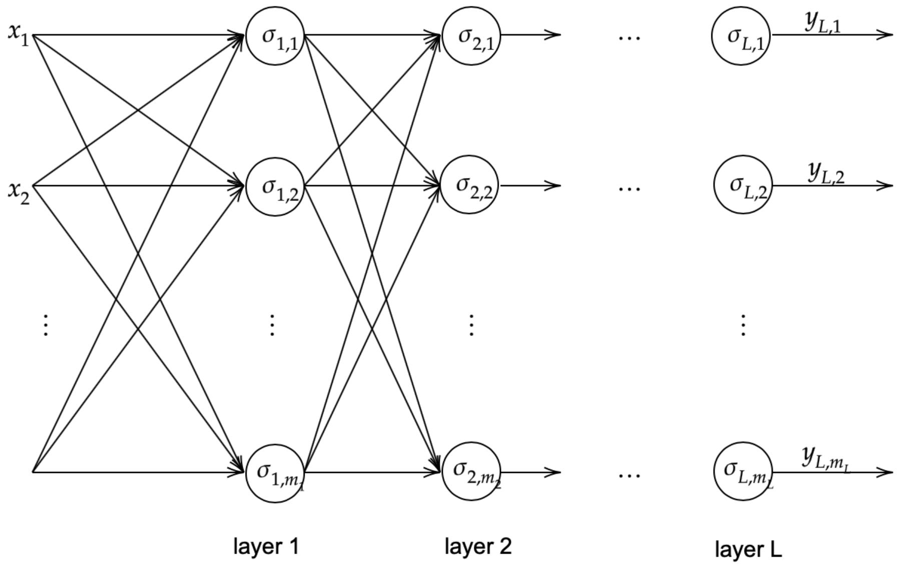


Figure 5: multilayer neural networks

the corresponding layer. If m_k denote the number of neurons of layer k, then we have

$$\begin{bmatrix} y_k \end{bmatrix}_{m_k} = \begin{bmatrix} \quad W_k \quad \end{bmatrix}_{m_k \times m_{k-1}} \begin{bmatrix} y_{k-1} \end{bmatrix}_{m_{k-1}} + \begin{bmatrix} b_k \end{bmatrix}_{m_k}.$$

Therefore, the following dynamical system is introduced(see Fig 6):

$$\begin{aligned} z_k &= W_k y_{k-1} + b_k, \quad k = 1, \cdots L \\ y_k &= \sigma(z_k) \qquad\qquad\qquad\qquad\qquad y_0 = x. \end{aligned}$$

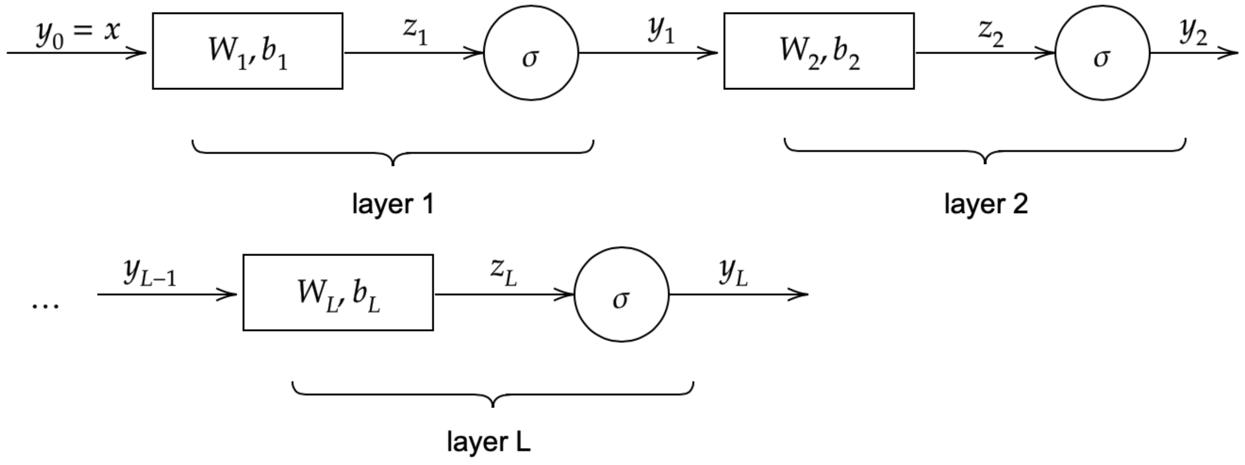


Figure 6: dynamical viewpoint for neural networks

A more formal formulation is included in the following subsection.

3.2 Reformulation: Optimal Control

The very essential task for deep learning is to find a parametric function approximator for some given function $f : X \to Y$, which maps inputs in X to labels Y, based on a given data set of pairs $\{x^i \in X, y^i = f(x^i) \in Y\}_{i=1}^N$. The process for finding the function approximator or equivalently the process for finding the best parameters, since the approximator is parametric, is referred as training. As mentioned in the last remark, for each pair of (x^i, y^i), there is a corresponding dynamical system, which is given by, in a more abstract way, the following

$$\begin{aligned} &y_k^i = F(k, y_{k-1}^i, \theta_k), \quad \theta_k = (W_k, b_k) \\ &y_0^i = x^i. \end{aligned}$$

If we define the loss function as $J_L = \sum_{i=1}^N g_L(f(x^i), y_L^i)$, where $g_L(\cdot)$ is a kind of error metric, measuring the distance from y_L^i to $f(x^i)$. Furthermore, the loss function can be regularized by adding a penalty term $J_R = \sum_{k=0}^{L-1} g_k(\theta_k)$. Therefore, the supervised learning problem can be reformulated as an optimal control problem:

$$\begin{aligned} \min_{\{\theta_k\}} \quad & J_L + \alpha J_R \\ \text{subject to} \quad & y_k^i = F(k, y_{k-1}^i, \theta_k), y_0^i = x^i \end{aligned}$$

Remark 9. *Pontryagin's maximum principle can be applied to solve the optimal control problem, which shall be detailed in the next subsection. For more details, refer to [3] and references therein.*

3.3 Maximum Principle in Discrete time

Pontryagin's maximum principle is proposed by Lev Pontryagin and his students to find the optimal control for a dynamical system under some constraints, by which the problem is reduced to a maximization of Hamiltonian associated with the original optimal control problem. For more, see [4][5]. Based on the optimal control problem formulated in last section, we are now in a position to detail Maximum principle in a discrete time setting.

In general, we consider a discrete time dynamical system: given an initial point $x_0 \in \mathbb{R}^n$ and $f : \mathbb{R}^{n+m} \to \mathbb{R}^n$

$$x_{k+1} = f_k(x_k, \mu_k), \quad k = 0, \cdots N-1,$$

where $x_k \in \mathbb{R}^n, \mu_k \in \mathbb{R}^m$. The objective is to find an optimal control $(\mu_0, \mu_1, \ldots, \mu_{N-1})$ and a corresponding state sequence $(x_0, x_1, \ldots, x_N)$ such that the following objective function is minimized

$$J(\mu_0, \mu_1, \ldots, \mu_{N-1}) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k), \tag{1}$$

and straightforward computation gives[1]:

$$\begin{aligned}
\nabla_{\mu_{N-1}} J =& \nabla_{\mu_{N-1}} g_N(f_{N-1}(x_{N-1}, \mu_{N-1})) + \nabla_{\mu_{N-1}} g_{N-1}(x_{N-1}, \mu_{N-1}) \\
&= \nabla_{\mu_{\mathrm{N}-1}} f_{N-1} \cdot \nabla_{x_N} g_N + \nabla_{\mu_{N-1}} g_{N-1}, \\
\nabla_{\mu_{N-2}} J =& \nabla_{\mu_{N-2}} g_N(f_{N-1}(f_{N-2}(x_{N-2}, \mu_{N-2}), \mu_{N-1})) \\
&+ \nabla_{\mu_{N-2}} g_{N-1}(f_{N-2}(x_{N-2}, \mu_{N-2}), \mu_{N-1}) + \nabla_{\mu_{N-2}} g_{N-2}(x_{N-2}, \mu_{N-2}) \\
=& \nabla_{\mu_{N-2}} f_{N-2} \cdot \nabla_{x_{N-1}} f_{N-1} \cdot \nabla_{x_N} g_N \\
&+ \nabla_{\mu_{N-2}} f_{N-2} \cdot \nabla_{x_{N-2}} g_{N-1} + \nabla_{\mu_{N-2}} g_{N-2} \\
=& \nabla_{\mu_{N-2}} f_{N-2} \left(\nabla_{x_{N-1}} f_{N-1} \cdot \nabla_{x_N} g_N + \nabla_{x_{N-2}} g_{N-1} \right) + \nabla_{\mu_{N-2}} g_{N-2}, \\
\nabla_{\mu_{N-3}} J =& \nabla_{\mu_{N-3}} g_N(f_{N-1}(f_{N-2}(f_{N-3}(x_{N-3}, \mu_{N-3}), \mu_{N-2}), \mu_{N-1})) \\
&+ \nabla_{\mu_{N-3}} g_{N-1}(f_{N-2}((f_{N-3}(x_{N-3}, \mu_{N-3}), \mu_{N-2}), \mu_{N-1}) \\
&+ \nabla_{\mu_{N-3}} g_{N-2}((f_{N-3}(x_{N-3}, \mu_{N-3}), \mu_{N-2}) + \nabla_{\mu_{N-3}} g_{N-3}(x_{N-3}, \mu_{N-3}) \\
=& \nabla_{\mu_{N-3}} f_{N-3} \cdot \nabla_{x_{N-2}} f_{N-2} \cdot \nabla_{x_{N-1}} f_{N-1} \cdot \nabla_{x_N} g_N \\
&+ \nabla_{\mu_{N-3}} f_{N-3} \cdot \nabla_{x_{N-2}} f_{N-2} \cdot \nabla_{x_{N-2}} g_{N-1} \\
&+ \nabla_{\mu_{N-3}} f_{N-3} \cdot \nabla_{x_{N-2}} g_{N-2} + \nabla_{\mu_{N-3}} g_{N-3}(x_{N-3}, \mu_{N-3}) \\
=& \nabla_{\mu_{N-3}} f_{N-3}(\nabla_{x_{N-2}} f_{N-2} \cdot \nabla_{x_{N-1}} f_{N-1} \cdot \nabla_{x_N} g_N + \nabla_{x_{N-2}} f_{N-2} \cdot \nabla_{x_{N-1}} g_{N-1} + \nabla_{x_{N-2}} g_{N-2}) \\
&+ \nabla_{\mu_{N-3}} g_{N-3}(x_{N-3}, \mu_{N-3}) \\
&\vdots \\
\nabla_{\mu_k} J =& \nabla_{\mu_k} f_k(\nabla_{x_{k+1}} f_{k+1} \cdot \ldots \cdot \nabla_{x_{N-1}} f_{N-1} \cdot \nabla_{x_N} g_N \\
&+ \nabla_{x_{k+1}} f_{k+1} \cdot \ldots \cdot \nabla_{x_{N-2}} f_{N-2} \cdot \nabla_{x_{N-1}} g_{N-1} \\
&+ \ldots + \nabla_{x_{k+1}} g_{k+1}) + \nabla_{\mu_k} g_k.
\end{aligned}$$

[1]In order to better present the iterative scheme for computing $\nabla_{\mu_k} J$, we are not following the normal way to present chain rule, and this doesn't affect the deduction.A more rigorous discussion about compatibility of dimension is attached as appendix

Hence, we have a compact representation for $\nabla_{\mu_k} J$:

$$\begin{aligned}\nabla_{\mu_k} J &= \mathbf{1}^T \left[\begin{array}{l} \nabla_{\mu_k} f_k \cdot \nabla_{x_{k+1}} f_{k+1} \cdot \dots \cdot \nabla_{x_{N-1}} f_{N-1} \cdot \nabla_{x_N} g_N \\ \nabla_{\mu_k} f_k \cdot \nabla_{x_{k+1}} f_{k+1} \cdot \dots \cdot \nabla_{x_{N-2}} f_{N-2} \cdot \nabla_{x_{N-1}} g_{N-1} \\ \vdots \\ \nabla_{\mu_k} f_k \cdot \nabla_{x_{k+1}} g_{k+1} \end{array}\right] + \nabla_{\mu_k} g_k \\ &= \mathbf{1}^T \left(\nabla_{\mu_k} f_k \cdot \left[\begin{array}{l} \nabla_{x_{k+1}} f_{k+1} \cdot \dots \cdot \nabla_{x_{N-1}} f_{N-1} \cdot \nabla_{x_N} g_N \\ \nabla_{x_{k+1}} f_{k+1} \cdot \dots \cdot \nabla_{x_{N-2}} f_{N-2} \cdot \nabla_{x_{N-1}} g_{N-1} \\ \vdots \\ \nabla_{x_{k+1}} g_{k+1} \end{array}\right] \right) + \nabla_{\mu_k} g_k. \qquad (2)\end{aligned}$$

If we define P_k as

$$P_k = \left[\begin{array}{l} \nabla_{x_k} f_k \cdot \dots \cdot \nabla_{x_{N-1}} f_{N-1} \cdot \nabla_{x_N} g_N \\ \nabla_{x_k} f_k \cdot \dots \cdot \nabla_{x_{N-2}} f_{N-2} \cdot \nabla_{x_{N-1}} g_{N-1} \\ \vdots \\ \nabla_{x_k} f_k \cdot \nabla_{x_{k+1}} g_{k+1} \\ \nabla_{x_k} g_k \end{array}\right],$$

and with a little abuse of notation, we introduce the tensor sum $\oplus$ as following(assume that the compatibility for dimension is always guaranteed):

$$P_k \oplus g = \left[\begin{array}{c} P_k \\ g \end{array}\right],$$

then we obtain following equations:

$$\nabla_{\mu_k} J = \mathbf{1}^T (\nabla_{\mu_k} f_k P_{k+1}) + \nabla_{\mu_k} g_k, \qquad (3)$$
$$P_k = \nabla_{x_k} f_k P_{k+1} \oplus \nabla_{x_k} g_k, \quad k = 1, 2, \cdots, N-1, \qquad (4)$$
$$P_N = \nabla_{x_N} g_N. \qquad (5)$$

In fact, (3) tells that the gradient of J equals to the gradient of Hamiltonian H_k, i.e. $\nabla_{\mu_k} J = \nabla_{\mu_k} H_k$, where H_k is defined as

$$H_k \triangleq g_k(x_k, \mu_k) + \mathbf{1}^T (f_k(x_k, \mu_k) P_{k+1}).$$

Suppose that $(\mu_0^*, \mu_1^*, \dots, \mu_{n-1}^*)$ is an optimal control and $(x_0^*, x_1^*, \cdots \quad x_N^*)$ is the corresponding state trajectory. Assume that the constraint sets U_k are convex. Then for all $k = 0, \dots, N-1$, first order condition gives

$$\nabla_{\mu_k}^T H(x^*, \mu_k^*, P_{k+1})(\mu_k - \mu_k^*) \geq 0, \forall \mu_k \in U_k,$$

where $P_1, \dots, P_N$ are obtained from the adjoint equation (4)(5). Also, as proposed in [2], by pushing the compositional approach to an infinitesimal limit, it is possible for us to produce nonlinear functions approximation using continuous dynamical systems, which offers more flexibility and in this case the associated optimal control problem is introduced as

$$\begin{array}{ll} \text{minimize} & V(t) = \int_{t_0}^{t_1} l(x(t), u(t), t) dt + M(x(t_1)) \\ \text{subject to} & \dot{x(t)} = f(x(t), u(t), t), \quad x(t_0) = x_0 \in \mathbb{R}^n \end{array}$$

where $V(t)$ can be interpreted as the running cost and $M(x(t_i))$ as the error metric, as suggested in [3].

Appendix A

In this appendix, we shall address the dimension-compatibility issue in section 3.3. Assume that $x_k \in \mathbb{R}^n, \mu_k \in \mathbb{R}^m$ and $f_k : \mathbb{R}^{n+m} \to \mathbb{R}^n, g_k : \mathbb{R}^{n+m} \to \mathbb{R}^n$ for $k = 0, \cdots, N-1$, whereas $g_N : \mathbb{R}^n \to \mathbb{R}$. Then, we have $\nabla_{x_k} f_k \in \mathbb{R}^{n\times n}, \nabla_{\mu_k} f_k \in \mathbb{R}^{n\times m}, \nabla_{x_k} g_k \in \mathbb{R}^{1\times n}, \nabla_{\mu_k} g_k \in \mathbb{R}^{1\times m}$ and $\nabla_x g_N \in \mathbb{R}^{1\times n}$. As shown in the computation of (1), chain rule yields

$$\nabla_{\mu_{N-1}} J = \nabla_{\mu_{\mathrm{N}-1}} f_{N-1} \cdot \nabla_{x_N} g_N + \nabla_{\mu_{N-1}} g_{N-1}.$$

However, it is impossible to multiply a $n \times m$ matrix with a $n-$dimensional vector. Actually, by chain rule, the gradient of J with respect to μ_{N-1} should be

$$\nabla_{\mu_{N-1}} J = \nabla_{x_N} g_N \cdot \nabla_{\mu_{\mathrm{N}-1}} f_{N-1} + \nabla_{\mu_{N-1}} g_{N-1},$$

and this is also true for all $\nabla_{\mu_k} J$. With the same reason, P_k, actually should be written as

$$P_k = \begin{bmatrix} \nabla_{x_N} g_N \cdot \nabla_{x_{N-1}} f_{N-1} \dots \cdot \nabla_{x_k} f_k \\ \nabla_{x_{N-1}} g_{N-1} \cdot \nabla_{x_{N-2}} f_{N-2} \cdot \dots \cdot \nabla_{x_k} f_k \\ \vdots \\ \nabla_{x_{k+1}} g_{k+1} \cdot \nabla_{x_k} f_k \\ \nabla_{x_k} g_k \end{bmatrix} \in \mathbb{R}^{(N-k+1)\times n},$$

With $\nabla_{\mu_k} f_k \in \mathbb{R}^{n\times m}$, we have

$$P_k \nabla_{\mu_k} f_k = \begin{bmatrix} \nabla_{x_N} g_N \cdot \nabla_{x_{N-1}} f_{N-1} \dots \cdot \nabla_{x_k} f_k \nabla_{\mu_k} f_k \\ \nabla_{x_{N-1}} g_{N-1} \cdot \nabla_{x_{N-2}} f_{N-2} \cdot \dots \cdot \nabla_{x_k} f_k \nabla_{\mu_k} f_k \\ \vdots \\ \nabla_{x_{k+1}} g_{k+1} \cdot \nabla_{x_k} f_k \nabla_{\mu_k} f_k \\ \nabla_{x_k} g_k \nabla_{\mu_k} f_k \end{bmatrix} \in \mathbb{R}^{(N-k+1)\times m},$$

Finally, we obtain the a more rigorous representation of $\nabla_{\mu_k} J$ (the counterpart is (3)):

$$\nabla_{\mu_k} J = \mathbf{1}^T (P_{k+1} \nabla_{\mu_k} f_k) + \nabla_{\mu_k} g_k,$$

where $\mathbf{1} \in \mathbb{R}^{N-k+1}$ and $\nabla_{\mu_k} g_k \in \mathbb{R}^{1\times m}$. Similarly, (4) in fact should be

$$P_k = P_{k+1} \nabla_{x_k} f_k \oplus \nabla_{x_k} g_k, \quad k = 1, 2, \cdots, N-1.$$

In a nutshell, with a little abuse of notation, our proof and deductions are still valid.

References

[1] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming," in *Optimization and neural computation series*, 1996.

[2] W. E, "A Proposal on Machine Learning via Dynamical Systems," *Communications in Mathematics and Statistics*, vol. 5, no. 1, pp. 1–11, 2017.

[3] Q. Li, L. Chen, C. Tai, and E. Weinan, “Maximum principle based algorithms for deep learning,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 5998–6026, 2017.

[4] V. Boltyanskiy, R. V. Gamkrelidze, and L. Pontryagin, “Theory of optimal processes,” tech. rep., JOINT PUBLICATIONS RESEARCH SERVICE ARLINGTON VA, 1961.

[5] L. S. Pontryagin, *Mathematical theory of optimal processes.* Routledge, 2018.