The Role of Systems Theory in Control Oriented Learning

Mario Sznaier * **Alex Olshevsky** ** **Eduardo D. Sontag** *

* *ECE Department, Northeastern University, Boston, MA 02115.*
Emails: msznaier@coe.neu.edu, e.sontag@northeastern.edu
** *ECE, Boston University, Boston, MA 02215. Email: alexols@bu.edu*

Abstract: This extended abstract discusses the role that systems theory plays in unveiling fundamental limitations of learning algorithms and architectures when used to control a dynamical system, and in suggesting strategies for overcoming these limitations. As an example, a feedforward neural network cannot stabilize a double integrator using output feedback. Similarly, a recurrent NN with differentiable activation functions that stabilizes a non-strongly stabilizable system must be itself open loop unstable, a fact that has profound implications for training with noisy, finite data. A potential solution to this problem, motivated by results on stabilization with periodic control, is the use of neural nets with periodic resets, showing that indeed systems theoretic analysis is instrumental in developing architectures capable of controlling certain classes of unstable systems. The abstract finishes by arguing that when the goal is to learn control oriented models, the loss function should reflect closed loop, rather than open loop model performance, a fact that can be accomplished by using gap-metric motivated loss functions.

Keywords: Control Oriented Learning, Neural Nets, Reinforcement Learning.

1. INTRODUCTION AND MOTIVATION.

Despite recent advances in Machine Learning (ML), the goal of designing control systems capable of fully exploiting the potential of these methods remains elusive. Modern ML methods can leverage large amounts of data to learn powerful predictive models, but such models are not designed to operate in a closed-loop environment. Recent results on reinforcement learning offer a tantalizing view of the potential of a rapprochement between control and learning, but so far proofs of performance and safety are mostly restricted to limited cases (e.g. finite horizon LQR/LQG or iterative tasks). Thus, learning elements are often used as black boxes within the loop, with limited interpretability and less than completely understood properties. Further progress hinges on the development of a principled understanding of the limitations of control-oriented machine learning. This extended abstract presents some initial results unveiling the fundamental limitations of some popular learning algorithms and architectures when used to control a dynamical system. For instance, it shows that even though feed forward neural nets are universal approximators, they are unable to stabilize some simple systems. Along these lines we also show that a recurrent neural net with differentiable activation functions that stabilizes a non-strongly stabilizable system must be itself open loop unstable, and discuss the implications of this fact for training with noisy, finite data. On the other hand, this difficulty can be overcome by using either time varying architectures or architectures with periodic resets. We also present some empirical evidence that conventional, off the shelf Reinforcement Learning will fail to stabilize non-strongly stabilizable plants. The extended abstract finishes by arguing that when the goal is to learn stabilizing controllers, the loss function should reflect closed loop performance, a fact that can be accomplished by using gap-metric motivated loss functions.

[1] This work was partially supported by NSF grant CNS–2038493, AFOSR grant FA9550-19-1-0005, and ONR grant N00014-21-1-2431.

1.1 Fundamental limitations of Feed Forward NN.

Even though Feed Forward NN (FFNN) are routinely used as controllers, there are fundamental obstructions that may prevent the existence of stabilizing FFNN controllers with continuous activation functions (Sontag and Sussmann (1980)). In this portion of the extended abstract we present some simple examples illustrating these limitations.

Single Hidden Layer FNN. Recall the single hidden layer FNN can approximate arbitrarily well any continuous functions (Cybenko (1989). However, as shown in (Sontag (1992)), there exists an asymptotically controllable system that has the origin as a locally asymptotically stable equilibrium point of the zero input dynamics and yet it cannot be be stabilized on compact sets using a single hidden layer FNN, even with discontinuous activation functions. This limitation arises from the fact that the (one sided) inverse needed to implement a stabilizing controller cannot be generically approximated by a linear combinations of scalar functions of linear combinations, even when the forward mapping is continuous.

Similarly, single hidden layer FNN cannot control nonholonomic systems due to their inability to implement Lie Brackets. On the other hand, since continuous-time

periodic controllers can overcome topological obstructions (Khaneja and Brockett (1980)) we conjecture that if $\dot{x} = f(x) + g(x)u$ is stabilizable, there is a recurrent NN (RNN) with (continuous) activation ReLU, state z, and input x, and a feedback $u = k(z, x)$ so that $\{(0, z)\}$ is asymptotically stable.

The discussion above illustrates the limitations of single hidden layer FFNNs when used as controllers. However, this leaves open the question of whether multi-layer FFNN can be used as universal controllers. In the next section we show that this is not the case.

1.2 Inadequacy of Deep FFNNs for output feedback

In this section we illustrate with a simple example the limitations of FFNNs when used to implement output feedback controllers. To this effect, consider the stabilization of a double integrator using output feedback,

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(x_1)\,, \end{aligned} \tag{1}$$

where $f(x_1)$ is implemented by a FFNN. Such a controller can never render the origin a globally asymptotically stable equilibrium point. To see this, consider the "energy" function $V(x_1, x_2) = \frac{1}{2}x_2^2 - \int_0^{x_1} f(\lambda)d\lambda$. Since $dV/dt \equiv 0$, trajectories starting at any $(0, x_2)$,$x_2 \neq 0$ cannot asymptotically approach the origin $(0, 0)$. To be precise, suppose $f(x_1)$ is locally bounded and Lebesgue measurable. Then $F(\xi) := \int_0^{\xi} f(\lambda)d\lambda$ is locally Lipschitz, and $x_1(t)$ is absolutely continuous (a.c.), so also $F(x_1(t))$ is a.c., so V is a.c. Thus, the chain rule can be applied, and V is constant along trajectories. (The a.c. property rules out examples such as the Cantor function, where derivatives can be identically zero yet the function is not constant.)

1.3 RNNs and non-strongly stabilizable systems

This portion of the extended abstract discusses the challenges in using RNNs to control non-strongly stabilizable plants, that is, Linear Time Invariant (LTI) plants that cannot be stabilized by open loop stable LTI controllers. These plants are interesting both on their own and because their relationship to the problem of simultaneous stabilization (Doyle et al. (1992)). Recall that a SISO plant is strongly stabilizable if it satisfies the parity interlacing property Doyle et al. (1992): the number of real poles in the right half plane (RHP) (counted according to their multiplicity) in between every pair of RHP zeros (including those at infinity) is even.

Proposition 1. *If a RNN with differentiable activation functions stabilizes a non-strongly stabilizable plant, then the RNN must be open loop unstable.*

The proof follows by considering the controller obtained by linearizing the input/output (between time series) mapping implemented by the NN.

A more interesting case arises if we allow for recurrent NN that implement non-smooth mappings. To investigate this case, consider an ideal setting where a known, open loop unstable internally stabilizing controller is used to train the neural net (Fig. 1). This scenario arises for instance when seeking to optimize performance. In this situation,

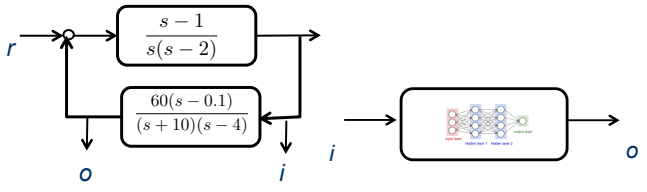

Fig. 1. Using the closed-loop signals generated by a stabilizing controller to train a NN.

one can use this pre-trained NN as an initial controller and then adjust its weights (for instance via gradient descent) to improve performance.

Let $i(t)$ and $o(t)$ denote the input output signals. Since the controller stabilizes the loop, it follows that $i(t), o(t) \in \mathcal{L}_\infty$, so in principle these bounded signals could be used to train a NN. Nevertheless, as we show next, a NN that interpolates all input/ouput pairs generated by an open loop unstable controller, has to be open loop unstable.

Proposition 2. *Consider an unbounded mapping $\mathcal{C}\colon \mathcal{L}_\infty \to \mathcal{L}_{\infty,e}$. Let $\mathcal{I}_b \subseteq \mathcal{L}_\infty$ denote the set of essentially bounded inputs that result in bounded outputs, i.e.*

$$\mathcal{I}_b \doteq \{r \in \mathcal{L}_\infty \colon (s \doteq \mathcal{C}r) \in \mathcal{L}_\infty\}$$

Then, if an operator $\mathcal{NN}$ is such that $\mathcal{NN}r = \mathcal{C}r$ a.e. for all $r \in \mathcal{I}_b$, $\mathcal{NN}$ must be open loop unstable, in the sense that there exists some $r_o \in \mathcal{L}_\infty$ such that $\mathcal{NN}r_o \notin \mathcal{L}_\infty$

From the observation above it follows that the NN can be trained in open-loop using the closed loop signals generated by an open loop controller only in the ideal case that these signals are perfectly known. This is a consequence of the fact that, since the NN is open-loop unstable, a suitably chosen perturbation of the input signal will lead to unbounded outputs. The discussion above leaves open the question of whether the NN can be trained in closed loop. As we show next, if the NN has differentiable activations, closed loop training is also likely to fail, due to the sensitivity of the parameters with respect to the observed outputs. For simplicity, we consider a SISO tracking scenario where the NN implements an LTI controller and the goal is to find the controller parameters θ that minimize some function $\mathcal{L}[e(\theta, u)]$ of the output e corresponding to a given input u, that is:

$$\theta^o = \operatorname*{argmin}_{\theta} \mathcal{L}[e(\theta, u)]$$

where

$$e(\theta) = \frac{u(s)}{1 + P(s)C(s, \theta)}$$

To illustrate the difficulties arising when the controller is open loop unstable, we will compute the gradient of the loss function with respect to the parameters of the controller. Let $C = \frac{N}{D}$ denote a coprime factorization of C, parameterized directly in term of its poles and zeros, that is $D(s) = D_o\Pi(s-\theta_i)$ and $N(s) = N_o\Pi(s-\psi_i)$. Since C is open loop unstable, at least one $\theta_i > 0$ and

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \theta_i} &= \frac{\partial \mathcal{L}}{\partial e}\frac{\partial e}{\partial C}\frac{\partial C}{\partial \theta_i} = \frac{\partial \mathcal{L}}{\partial e}\frac{-P}{(1+PC)^2}\frac{\partial C}{\partial \theta_i}u(s) \\ &= \frac{\partial \mathcal{L}}{\partial e}\frac{P}{(1+PC)^2}\left(\frac{C}{s-\theta_i}\right)u(s) \end{aligned} \tag{2}$$

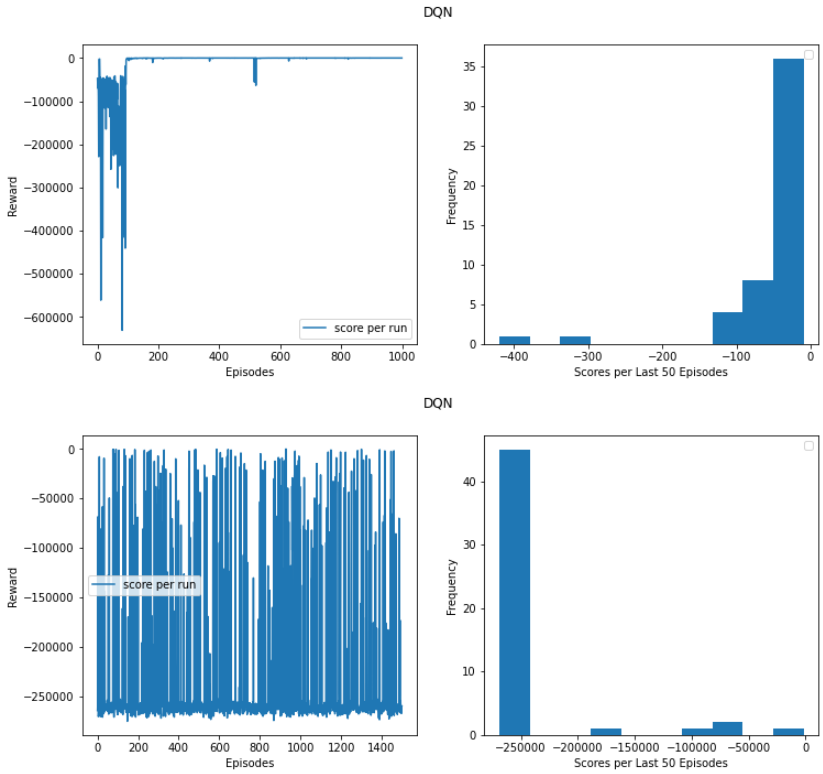

Fig. 2. Deep Reinforcement applied to a double integrator (top) and a non-strongly stabilizable plant (botton).

In the ideal case where $\frac{\partial C}{\partial \theta_i}$ can be exactly computed the poles of $\frac{\partial C}{\partial \theta_i}$ are cancelled by the zeros of $\frac{-P}{(1+PC)^2}$ and the overall system is stable. On the other hand, if only approximate values of the gradient are available (due for instance to finite and/or discrete time approximations), then this exact pole-zero cancellation no longer holds, leading to an unstable mapping $\frac{\partial C}{\partial \theta_i} \to \frac{\partial \mathcal{L}}{\partial \theta_i}$.

The developments above raise the question of whether a non-strongly stabilizable plant can be stabilized by an open loop stable controller. An affirmative answer to this question was given in Savkin and Petersen (1997), showing that this is indeed possible when using linear time varying, infinite dimensional controllers. An alternative, simpler controller is presented below:

Proposition 3. *Consider a non-strongly stabilizable LTI plant P and an LTI stabilizing controller with state space realization: A_c, B_c, C_c, D_c. Then the controller*

$$\mathcal{C}(y) = \begin{cases} \dot{x}_c = A_c x_c + B_c y \\ x_c(t^+) = x_c(t^-),\ t \neq kT \\ x_c(kT) = 0 \\ u = C_c x_c + D_c y \end{cases} \tag{3}$$

is open loop stable and stabilizes P.

Intuitively, the states of the controller are reset every T seconds to prevent them from growing too large. At the same time, since for $t \in (kT, (k+1)T)$ the LTI controller is acting, T can be chosen so that at the end of each cycle the state of the plant satisfies $\|x(kT+T)\| < \|x(kT)\|$. While in principle this avoids the difficulties entailed in training an open-loop unstable controller, at the moment is unclear how to implement and train such a controller using available NN architectures.

1.4 Reinforcement Learning

Next, we present some experiments illustrating the difficulties of using Reinforcement Learning to control non-strongly stabilizable plants. Consider the problem of stabilizing a plant using Deep Reinforcement Learning. To this effect, we considered a neural network architecture consisting of two hidden layers with leaky ReLu activations and a set of discrete actions $\mathcal{U}$. The NN takes an observation (i.e., $y_k = Cx_k$) and outputs a vector $q_k = V_\theta(y_k)$ of the same dimension as the number of actions, where each entry is a prediction of the value from taking the corresponding action. The next control action u_k is selected as the one corresponding to the maximal entry in q, with probability $1 - \epsilon_k$, or a random action with probability ϵ_k, where $\epsilon_k = \max\{\epsilon_{\min}, 0.99 * \epsilon_{k-1}\}$ The reward corresponding to the action u_k at state x_k is set to be $-\|x_k\|_2^2$.

The neural net was trained with Q-learning as follows. Let $u_{\text{taken,k}}$ denote the action taken at step k, and let q_{k+1} be the vector obtained by applying the neural network to the next observation $y_{k+1} = Cx_{k+1}$. We then set $\hat{q}_k$ to be the vector obtained by replacing the entry in q_k corresponding to $u_{\text{taken,k}}$ with $-||x_k||^2 + \gamma \max\{q_{k+1}\}$, where max applied to a vector denotes the largest entry. Finally, we perform a gradient descent step on θ, the weights of the NN, with objective $\|V_\theta(y_k) - \hat{q}_k\|^2$. Note that while knowledge of the true states was used in training (through the reward), the policy here depends only on the observations y_k. We applied this approach on both an "easy" plant (a double integrator with state feedback)

$$\begin{aligned} x(k+1) &= \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k) \\ y(k) &= x(k) \end{aligned} \tag{4}$$

and a "hard" one (not strongly stabilizable)

$$x(k+1) = \begin{bmatrix} 1.2 & 0 \\ 0.1 & 1 \end{bmatrix} x(k) + \begin{bmatrix} 0.1 \\ 0 \end{bmatrix} u(k) \qquad (5)$$
$$y(k) = [1 \ -1]\, x(k)$$

As shown in Fig. 2 the Deep RL algorithm described above stabilizes the "easy" case but fails to do so for the non-strongly stabilizable one.

1.5 Open Loop vs Closed Loop Distances

In this portion of the extended abstract we argue that when using a NN to model a plant, the loss function used to train it should take into account the closed-loop distance between the the unknown plant and its model, rather than the open loop one. Consider the open-loop unstable plant $G_1 = \frac{100}{2s-1}$. Modelling this plant with a NN such that the open loop distance, measured in terms of the induced norm $\|(G_1 - NN)\|_{\ell_i \to \ell_o}$ is finite, will require an open loop unstable net. On the other hand, when the loop is closed with the simple controller $K = 1$, the original plant G_1 and the open-loop stable plant $G_2 = \frac{100}{2s+1}$ have virtually indistinguishable performance (Fig 3) Thus, if the goal

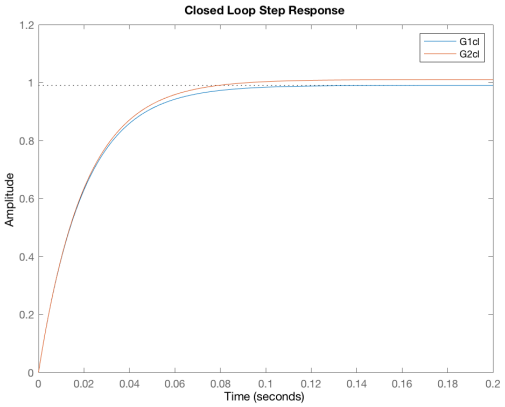

Fig. 3. Closed loop step responses of G_1 and G_2.

is to designing controllers, the stable plant G_2, which is substantially easier to model using a NN, can be used as a proxy for G_1 in the design process. This observation suggests that, when training a NN, one should try to minimize a closed-loop distance, rather than an open loop one. One such metric is the gap metric (see for instance Zhou and Doyle (1998)). Given two plants G_1, G_2 with normalized coprime factorizations $G_i = \frac{N_i}{D_i}$, $i = 1, 2$ the ν-gap δ_ν is defined by

$$\delta_\nu(G_1, G_2) = sup_w | - N_2(jw)M_1(jw) + \\ + M_2(jw)N_1(jw)|$$

Plants with small δ_ν can be stabilized by the same $\mathcal{H}_\infty$ optimal controller and have similar closed loop transfer functions (see Zhou and Doyle (1998) for a formalization of this statement). For instance, for the example above $\delta_\nu(G_1, G_2) = 0.02$, which explains the virtually indistinguishable closed loop responses. This suggest that one should learn coprime factorizations, rather than plants, and then perform a model (in)validation step, as proposed in Steele and Vinnicombe (2001) to estimate the gap between the learned model and the true plant. This approach has the additional advantage that it can handle unstable plants. While learning coprime factors directly from data is an open problem, the results below suggest that, at least in the noiseless case, this can be accomplished by solving two convex Nevanlinna Pick interpolation problems.

Proposition 4. *Given input/output pairs* $\{r(z_i), y(z_i)\}_{i=1}^n$ *there exist stable transfer functions* $N(z), M(z)$ *such that* $y(z_i) = \frac{N(z_i)r(z_i)}{M(z_i)}$ *if and only if there exist* $u(z)$ *such that following conditions hold:*

$$P_N = \left[\frac{r(z_i)r^*(z_j) - u(z_i)u^*(z_j)}{1 - (z_i z_j^*)^{-1}}\right]_{i,j} \succeq 0$$
$$P_M = \left[\frac{y(z_i)y^*(z_j) - u(z_i)u^*(z_j)}{1 - (z_i z_j^*)^{-1}}\right]_{i,j} \succeq 0$$

Using Schur complements, these conditions can be transformed into convex LMIs in u. Once the Pick matrices P_N and P_M have been found, state space realizations for N and M can be obtained using for instance the formulas in Parrilo et al. (1998).

1.6 Conclusions

This extended abstract illustrates the challenges entailed in using ML to control dynamical systems. As shown here, learning stabilizing controllers places additional constraints on the architectures and on the algorithms used to train them. Thus, we argue that control-agnostic ML is unlikely to succeed in controlling challenging systems. Rather, the choice of representations and training has to be guided by systems theory.

REFERENCES

Cybenko, G. (1989). Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 303–314.

Doyle, J.C., Francis, B.A., and Tannenbaum, A. (1992). *Feedback Control Theory*. Macmillan, Toronto.

Khaneja, N. and Brockett, R. (1980). Dynamic feedback stabilization of nonholonomic systems. In *Proc. IEEE CDC, Phoenix, Dec.1999*, 1640–1645.

Parrilo, P.A., Sznaier, M., Sanchez-Pena, R.S., and Inanc, T. (1998). Mixed time/frequency-domain based robust identification. *Automatica*, 34(11), 1375–1389.

Savkin, A. and Petersen, I. (1997). A method for simultaneous strong stabilization of linear time-varying systems. *IFAC Proceedings Volumes*, 30(16), 165–169.

Sontag, E. (1992). Feedback stabilization using two-hidden-layer nets. *IEEE Trans. Neural Networks*, 3, 981–990.

Sontag, E. and Sussmann, H. (1980). Remarks on continuous feedback. In *Proc. IEEE CDC, Albuquerque, Dec.1980*, 916–921.

Steele, J. and Vinnicombe, G. (2001). Closed-loop time-domain model validation in the nu-gap metric. In *Proceedings of the 40th IEEE CDC*, volume 5, 4332–4337.

Zhou, K. and Doyle, J.C. (1998). *Essentials of Robust Control*. Prentice Hall.