ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

On Learning a Control System without Continuous Feedback

Georgi Angelov[*] and Bogdan Georgiev[†‡]

Abstract.
We discuss a class of control problems by means of deep neural networks (DNN). Our goal is to develop DNN models that, once trained, are able to produce solutions of such problems at an acceptable error-rate and much faster computation time than an ordinary numerical solver. In the present note we study two such models for the Brockett integrator control problem.

1 Introduction

The recent work of Breen et al [1] approached the solutions of a chaotic system (the well-known 3-body problem) through a Multilayer Perceptron architecture (MLP). There it was shown that an ensemble of solutions given by an appropriate numerical integration scheme can be utilized to train a DNN that provides acceptable solutions at a fixed computational cost - in fact, after training the DNN performs much faster than the corresponding numerical solver.

Motivated by this observation, we study a certain class of control problems, where a non-trivial feature is that one cannot prescribe continuous control procedures (control law). It is interesting to ask to what extent one could effectively "learn" the solutions of such control systems. In this direction, we investigate the well-known Brockett integrator system and explore models for solutions' learning and prediction [1]. As in the case of [1], after training we observe an acceptable error-rate and much faster computing performance than the numerical solvers.

1.1 Related Work

The literature on learning techniques, control theory and their interaction is vast - without aim of being exhaustive we mention a few relevant sources. An overview and a certain convex approach for neural network models for optimal control are discussed in [2]. We also refer to references therein. It should be noted that various frameworks and a deep neural network architectures for stochastic controls by means of recurrent and fully connected layers have also been proposed.

Concerning the theory of discontinuous control law systems, Brockett [3] constructs a dynamical system that cannot be asymptotically null stabilized by a continuous feedback. For this system Krastanov [5] proposes an approach

[*]Sofia University.
[†]Fraunhofer IAIS, the ML2R project and the Research Center for ML, Fraunhofer IAIS.
[‡]Equal Contribution.
[1]Code for our experiments is available at: https://github.com/bogeorgiev/ml-and-optimal-control.

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

for the construction of a discontinuous feedback control law that asymptotically stabilizes the system. Clarke [6] uses a "sample-and-hold" approach based on nonsmooth control Lyapunov functions and Vdovin et. al. [7] utilize methods of optimal control theory.

2 Control Systems with and without continuous feedback

Let us consider the following control system:

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0 \in \mathbb{R}^n, \quad f(0,0) = 0, \tag{1}$$

with $u(\cdot) \in U$, closed subset of $\mathbb{R}^m$, and $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a smooth map. It is well known that if f is linear, the system (1) is globally asymptotically null controllable if and only if it is stabilizable by a linear feedback [4]. In the nonlinear case, even for systems with $n = m = 1$, continuous feedback may not exits. In the seminal paper [3], Brockett provides a topological condition that is necessary for the existence of a continuous stabilizing feedback law.

In many areas of control theory and in practice discontinuous feedback arises naturally. One can easily construct examples that fail to be locally asymptotically stabilizable by continuous feedback, but that are stabilizable by discontinuous feedback. A systems that is null controllable, but does not satisfy the Brockett necessary condition is [3]:

$$\dot{x} = u, \quad \dot{y} = u, \quad \dot{z} = xv - yu, \tag{2}$$

where $(u, v) \in \mathbb{R}^2$ and $(x, y, z) \in \mathbb{R}^3$. This seemingly simple system has become a benchmark example for nonlinear control methods and for decades many authors have discovered control strategies for the Brockett integrator (2) as we previously pointed out.

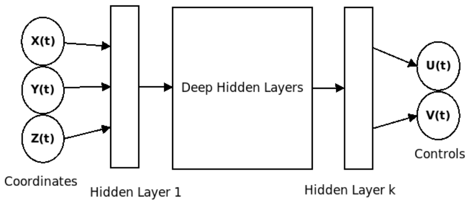
3 Model Exploration and Experimental Results

3.1 Trajectory Sampling

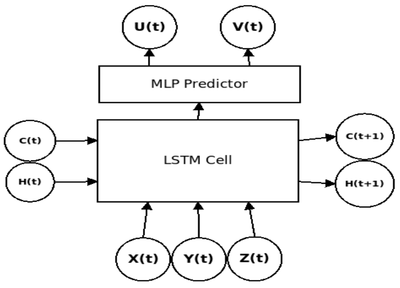
In order to create the dataset and generate trajectories that approach the origin we solve numerically an optimal control problem via optimization solver using Brockett's integrator as system dynamics. Let us formulate the optimal control problem:

$$\min \int_{t_0}^{T} x^2 + y^2 + z^2 dt, \quad \text{s.t.} \quad \dot{x} = u, \dot{y} = v, \dot{z} = xv - yu. \tag{3}$$

Here at time t_0 the starting point is $(x_0, y_0, z_0) \in \mathbb{R}^3$ and one aims to find controls (u, v) that steer the system to the origin $(0, 0, 0)$. By transforming the



(a) An MLP predictor: The current coordinates are fed forward through a fully connected multilayer perceptron to obtain the corresponding controls.



(b) An LSTM predictor: The individual one-step MLP predictors are assembled into an LSTM chain.

Fig. 1: An overview of the control prediction architectures.

optimal control problem (3) to a discrete optimization problem we could obtain a discontinuous control law that steers the trajectory to null.

We consider the discrete approximation to problem (3) that is obtained by solving the Brockett intergrator using a Runge-Kutta integration scheme. In [8] the authors use a s-stage Runge-Kutta scheme and prove convergence and error rate of $O(h^2)$ to the optimal solution. We use a second-order scheme suggested by [8].

Using an uniform mesh of width $h = 1/N$, where $N \in \mathbb{N}$ denotes the number of steps, the second-stage Runge-Kutta discretization for (3) that we use is:

$$\min h \sum_{k=0}^{N-1} (x_{k+1/2}^2 + y_{k+1/2}^2 + z_{k+1/2}^2) \tag{4}$$

$$\begin{cases} x_{k+1/2} = x_k + \frac{h}{2} u_{k+1/2}, \quad x_{k+1} = x_k + h u_{k+1/2}, \\ y_{k+1/2} = y_k + \frac{h}{2} v_{k+1/2}, \quad y_{k+1} = y_k + h v_{k+1/2}, \\ z_{k+1/2} = z_k + \frac{h}{2}(x_k v_{k+1/2} - y_k u_{k+1/2}), \\ z_{k+1} = z_k + h(x_{k+1/2} v_{k+1/2} - y_{k+1/2} u_{k+1/2}) \end{cases} \tag{5}$$

and (x_0, y_0, z_0) is the initial point. The nonlinear optimization problem (5) can be numerically solved by various algorithms. We use a limited-memory quasi-Newton algorithm ”L-BFGS-B” described in [9]. For the generation of the dataset we sample initial points with coordinates in the interval $[0, 1)$ and for every random initial point we solve (5). From the solution we obtain the control strategies $u_{k+1/2}$ and $v_{k+1/2}$ for $k = 0, ..., N-1$, this are the intermediate controls that should be used between two mesh points defined by the step h.

In our analysis we use $N = 70$ and generate a dataset of 5000 trajectories, generated by the solutions of (5) with different uniformly picked initial points. We partition the dataset so that 80% are used for training and validation, the other 20% are used for testing.

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

3.2 One-step control prediction by means of an MLP

We have first tested the one-step prediction capabilities of a Multilayer Perceptron (MLP). The basic architecture is summarized in Figure (1a). Given the initial point (x_0, y_0, z_0) of a trajectory, the goal was to learn the corresponding control quantities (u_0, v_0). To this end, we used a plain MSE loss function and, motivated by initial complexity experiments, we have selected an MLP-architecture consisting of 4 fully-connected hidden layers with 70 neurons and standard ReLU activations. We utilized an Adam optimizer and a learning rate of $\sim 10^{-4}$ with batches of 1000 samples. After training, the model was evaluated on the test set by requiring that the predicted controls steer the initial spatial point close to the next spatial point. This is formalized by the following metric:

$$M_1^t := \|(x_1^t, y_1^t, z_1^t) - F\left(\mathrm{MLP}(x_0^t, y_0^t, z_0^t)\right)\|, \tag{6}$$

where (x_i^t, y_i^t, z_i^t) denotes the i-th point of the trajectory t; MLP denotes the MLP one-step predictor and F denotes the map that given a current spatial point and controls uses the system's dynamics (i.e. (5)) to compute the next spatial point. Analyzing M_1^t over the test set T, we computed a mean of ~ 0.0129 and standard mean deviation of ~ 0.0187 - these could be considered on an acceptable order of $1/N$.

However, initial experiments with MLP architectures for a multiple-steps prediction instead of single one-step prediction appeared to be more challenging. This is motivation behind augmenting the model with a recurrent structure.

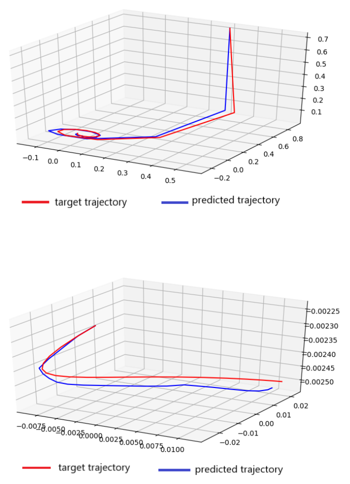


Fig. 2: Trajectory Comparison: First and Last 20 steps.

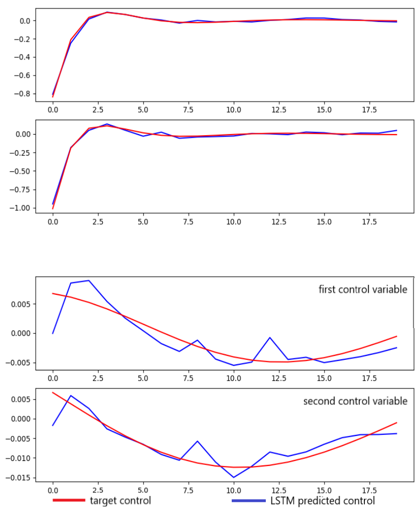


Fig. 3: Controls Comparison: First and Last 20 steps.

3.3 An LSTM approach for multiple-step prediction

In a way, we assembled the individual MLP one-step predictors into a recurrent scheme. We opted for a basic LSTM (Long-Short-Term-Memory) model whose cell computation is sketched in Figure (1b): at each time step t the LSTM produces a hidden-state $H(t)$ that is fed forward through an MLP one-step predictor. Among other advantages LSTMs are assumed to be robust against the vanishing-gradients problem [10]. Given a trajectory $\{(x_i, y_i, z_i)\}_{i=0}^{N-1}$ the LSTM model is supposed to learn and predict the corresponding controls $\{(u_i, v_i)\}_{i=0}^{N-1}$. We trained the LSTM model in a supervised way using the following loss function:

$$\mathcal{L} := \lambda_1 \, \mathrm{MSE}\left(\{(u_i, v_i)\}_{i=0}^{N-1}, \{\mathrm{LSTM}_i\}_{i=0}^{N-1}\right) \tag{7}$$

$$+ \lambda_2 \, \mathrm{MSE}\left(\{(x_i, y_i, z_i)\}_{i=0}^{N-1}, \{F\left(\mathrm{LSTM}_i\right)\}_{i=0}^{N-1}\right). \tag{8}$$

Here LSTM_i denotes the predicted controls at the i-th time step; F is the mapping given by (5) as above and λ_1, λ_2 are hyperparameters (after experimentation, we set the ratio λ_2/λ_1 to 3). The motivation for introducing the second term on the RHS lies in dealing with dynamics F that might have low regularity; furthermore, intuitively it also speeds up the learning of the controls' "spatial meaning". We note that given only an initial point (x_0, y_0, z_0), the LSTM model produces the other trajectory points by using the mapping F: from (x_0, y_0, z_0) the model computes (u_0, v_0), then $F(u_0, v_0)$ gives the next point (x_1, y_1, z_1), and so the process continues.

After training, we evaluated the metrics M_i^t from (6) for each trajectory point i. To ease the presentation, we first analyzed $\{M_i^t\}_{i=0}^{20}$ over the test set and computed mean and standard mean deviation - the results are summarized in the following table:

i:	0	3	6	9	12	15	18
mean(M_i):	0	0.0259	0.0203	0.0176	0.0159	0.0155	0.0184
std(M_i):	0	0.0136	0.0104	0.0095	0.0079	0.0080	0.0088

The results for the last points $\{M_i^t\}_{i=50}^{70}$ are summarized in the table:

i:	51	54	57	61	64	67	70
mean(M_i):	0.0027	0.0026	0.0022	0.0025	0.0026	0.0027	0.0035
std(M_i):	0.0016	0.0015	0.0014	0.0015	0.0015	0.0015	0.0021

Similarly to the one-step prediction case the error rates can be seen of the order of $1/N$. A random test sample is visualized in Figures (2) and (3). The red curves denote the trajectory/controls computed by a numerical solver; whereas blue curves denote the ones computed by the LSTM model. since the trajectories converge very quickly to the origin, it eases the visualization to consider the initial and the last points separately. We note that the controls produced the LSTM model might exhibit small-scale perturbations, but, however, follow the trend of the solver-produced controls closely.

4 Conclusion and Further Work

We studied models for one-step and multiple-step predictions for a control system without continuous feedback. We showed that a reasonable error-rate of the order of $1/N$ where N denotes the number of time steps could be produced: once trained the approach is much more computationally faster than using a state-of-art numerical solver. In an upcoming work we plan to build upon the structure of our models, the complexity of the studied problem and theoretical foundation. We consider the present note as just the beginning step towards further (both theoretical and practical) work in the subject.

References

[1] Philip G. Breen, Christopher N. Foley, Tjarda Boekholt, Simon Portegies Zwart: Newton vs the machine: solving the chaotic three-body problem using deep neural networks, preprint, arXiv:1910.07291.

[2] Yize Chen and Yuanyuan Shi and Baosen Zhang, Optimal Control Via Neural Networks: A Convex Approach, International Conference on Learning Representations, 2019

[3] R. W. Brockett: Asymptotic stability and feedback stabilization. In R. W. Brockett, R. S. Millman, and H. J. Sussmann, editors, Differential Geometric Control Theory, pages 181-191. Birkhauser, Boston, 1983.

[4] Sontag, E.: Mathematical Control Theory: Deterministic Finite Dimensional Systems, Texts in Applied Mathematics 6, Springer-Verlag, New York, Berlin, Heidelberg, London, Paris, Tokyo, Hong Kong (1990).

[5] Krastanov M.I. (2006) On the Synthesis of a Stabilizing Feedback Control. In: Lirkov I., Margenov S., and Waśniewski J. (eds) Large-Scale Scientific Computing. LSSC 2005. Lecture Notes in Computer Science, vol 3743. Springer, Berlin, Heidelberg.

[6] F. Clarke. Discontinuous feedback and nonlinear systems. In Proceedings of IFAC Conf. Nonlinear Control (NOLCOS), Bologna, pages 1-29, 2010.

[7] S. A. Vdovin, A. M. Taras'yev, and V. N. Ushakov. Construction of the attainability set of a brockett integrator. Journal of Applied Mathematics and Mechanics, 68:631-646, 2004.

[8] Dontchev, A. L., Hager, W. W., Veliov, V. M. (2000). Second-Order Runge-Kutta Approximations in Control Constrained Optimal Control. SIAM Journal on Numerical Analysis, 38(1), 202-226.

[9] R. H. Byrd, P. Lu and J. Nocedal. A Limited Memory Algorithm for Bound Constrained Optimization, (1995), SIAM Journal on Scientific and Statistical Computing, 16, 5, pp. 1190-1208.

[10] Sepp Hochreiter, Jürgen Schmidhuber: Long short-term memory, In: Neural Computation (journal), vol. 9, issue 8, S. 1735-1780, 1997.