Trabalho apresentado no XLI CNMAC, Unicamp - Campinas - SP, 2022.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics
Preprint

Deep Neural Networks as Optimal Control Problems

Diego Dias Sobrinho[1]
UFRJ, Rio de Janeiro, Brazil
Rafael Benchimol Klausner
PSR Energy Consulting and Analytics, Rio de Janeiro, Brazil
Roberto Machado Velho
Sarpen Quant Investments, São Paulo, Brazil

Deep Neural networks play an important role in regression and classification problems. A new approach [1] regards the training of such networks via optimal control problems. Let

$$u^{[j]} := \left(K^{[j]}, \beta^{[j]}\right), \quad j = 0, \ldots, N-1, \quad u = \left(u^{[0]}, \ldots, u^{[N-1]}\right),$$

with u a combined vector of values to be determined, where $K^{[j]}$ is a $n \times n$ matrix of weights and $\beta^{[j]}$ are the biases, for a network of N layers. Given the dataset $(x_i, z_i)_{i=1}^m$, where $x_i \in \mathbf{R}^n$ are the inputs and $z_i \in \mathbf{R}$ the outputs, the training process is translated in solving the optimization problem:

$$\min_{y,u,W,\mu} \quad \sum_{i=1}^{m} \left| \left(W y_i^{[N]} + \mu\right) - z_i \right|^2, \tag{1}$$

where $W \in \mathbb{R}^{1\times n}$ a weight vector and $\mu \in \mathbb{R}$ a scalar bias, subject to the constraints

$$y_i^{[j+1]} = y_i^{[j]} + \Delta t \; f(y_i^{[j]}, u^{[j]}), \qquad j = 0, \ldots, N-1, \qquad y_i^{[0]} = x_i, \qquad \Delta t = 1. \tag{2}$$

Generally, the function f has the form $f\left(y_i^{[j]}, u^{[j]}\right) := \sigma\left(K^{[j]} y_i^{[j]} + \beta^{[j]}\right)$, σ denoting the activation function. The approach to train this neural network is solving the previous discrete optimization problem. In contrast, one regard the training as a continuous control problem. Suppose that $y_i = y_i(t)$ and $u = u(t) = (K(t), \beta(t)), t \in [0, T]$. Taking $\dot{y}(t) = f\left(y(t), u(t)\right)$, (2) becomes the explicit Euler method for the initial value problem $\dot{y_i}\left(t\right) = f\left(y_i(t), u(t)\right)$, $y_i(0) = x_i$. One can see that the optimization problem (1) is the discretization of the optimal control problem:

$$\min_{y,u,W,\mu} \quad \sum_{i=1}^{m} \left|(W y_i(T) + \mu) - z_i\right|^2, \text{ subject to } \; \dot{y_i}\left(t\right) = f\left(y_i(t), u(t)\right); \; y_i(0) = x_i. \tag{3}$$

The solution to (3) can be understood as a discretization method for the time evolving ordinary differential equation (ODE) in (3), allowing us the use of multiple ODE solvers, as the classical Runge-Kutta method.

We devised the following experiment for comparing the training of a neural network via both approaches. We set the function $g(s) = 2s + \sin(2s) + 3$ on the interval $[0, 5]$ as the reference one and from $m = 251$ input points s_i equally distributed along the previous interval, we produce outputs $z_i = g(s_i) + \epsilon_i$, with ϵ_i a random Gaussian noise. The set $(x_i, z_i)_{i=1}^m$, $x_i = (s_i, 0, 0) \in \mathbb{R}^3$ is the dataset of interest for our experiment. In order to fit such dataset, we choose a neural network with $n = 3$ neurons per layer. We opt for $N = 3$ layers and the hyperbolic tangent as activation function. Such network was trained via the classical method using the ADAM optimizer routine contained in the package `Flux.jl`, while the continuous version used gradient descent with backtracking and a forward stepper Euler Integrator with $T = 3$, both running $20k$ iterations and implemented in Julia.[2]. All the parameters (weights and biases) were started with uniformly

[1] diegodias3468@gmail.com, rafa.bench@gmail.com, roberto.velho@gmail.com
[2] All the code is available at https://github.com/rafabench/OCDNN.jl

random inputs. In Figure 1 we compare the performance of both methods ("Neural Network" vs. "Euler Network") regarding the decay of the loss measured via the mean-squared error (mse) - the expression to be minimized in 1 - and the fit to the data (after $20k$ iterations). The Euler Network tends to require fewer iterations than the Neural Network to achieve the same loss. Such performance motivates the study of other numerical methods for ODEs, what we performed using Runge-Kutta methods of order two to four (RK2, RK3, RK4). We repeat the previous regression problem (with new randomly initial parameters) comparing such solvers and present the results in Figure 2. The results show the trend that the higher the order of the ODE solver, the fewer iterations are required to achieve a certain loss. Further investigation is necessary to understand the deeper relation between the ODE solver and the necessary number of iterations to achieve a good quality of data fit. This would contrast with the original paper [1], where the studied classification problem did not show improvement of the fit as the ODE solver was changed.

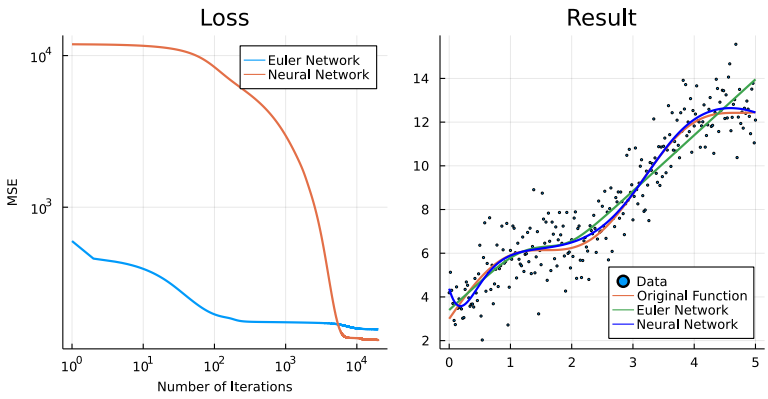

Figure 1: Loss evolution along iterations and fit of data for the Neural and Euler networks.

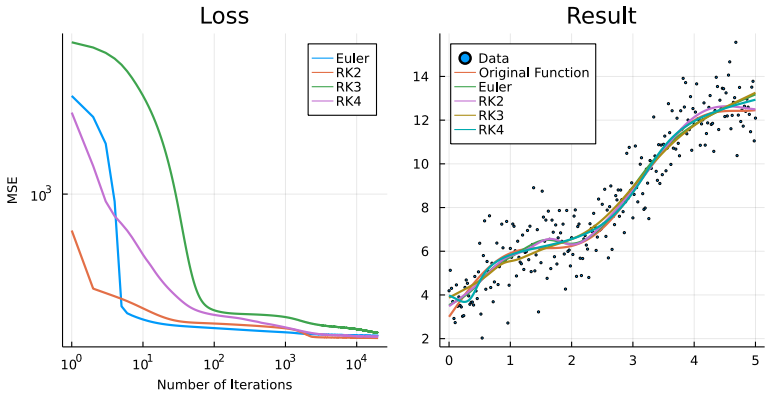

Figure 2: Loss evolution along iterations and fit of data for different ODE solvers.

References

[1] Martin Benning et al. “Deep learning as optimal control problems: Models and numerical methods”. In: **Journal of Computational Dynamics** 6.2 (2019), pp. 171–198.