Using Artificial Neural Network to solve ODE and PDE

POLITECNICO DI MILANO

Marco Di Giovanni[1]

Marco Brambilla[1], Pavlos Protopapas[2]

marco.digiovanni@polimi.it, marco.brambilla@polimi.it, pavlos@seas.harvard.edu

[1] *Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy*

[2] *Institute for Applied Computational Science, Harvard University, Boston, United States*

ABSTRACT

Classical approaches for the numerical solution of ordinary and partial differential equations are widely used due to their effectiveness and simplicity.

In this project, we investigate the application of artificial neural networks in the field of numerical approximation of differential equations, since the Universal approximation theorem states that a shallow feed-forward neural network is able to approximate any continuous function to a given precision if the number of hidden neurons is big enough.

METHOD

We start from a general differential equation of the form:

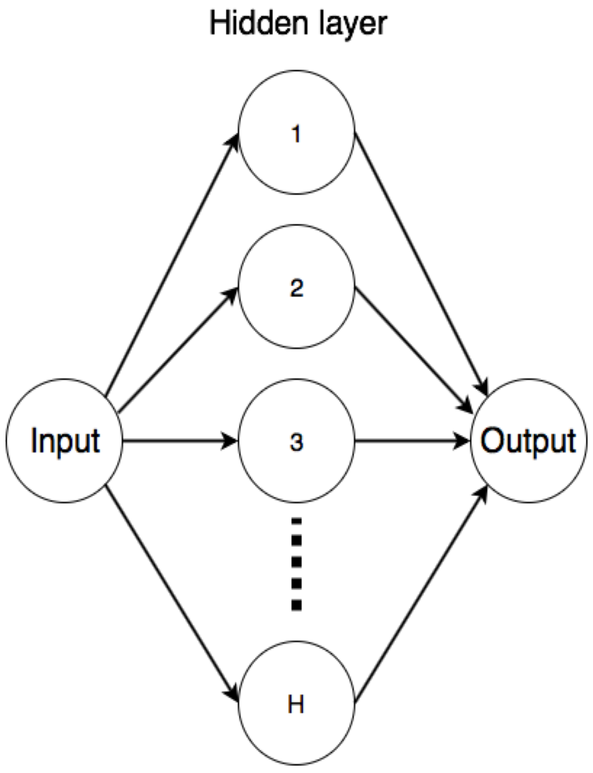
$$f(t, x, x', x'', x''', \dots) = 0$$

where $x' = \frac{\partial x}{\partial t}$, …

with fixed boundary conditions: $x(t_0) = x_0$, …

The goal is to learn $x(t) \to \hat{x}(t, \theta)$ using a deep learning approach, where θ are the parameters of the NN

As a starting point, we chose an ANN with a simple architecture: a shallow fully connected NN with one hidden layer of H neurons, one input (t) and one output (x(t)), see figure.



The activation function chosen is a Logsigmoid:

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

The LBFGS algorithm is used, a very memory intense optimizer, since the quantity of data is small enough.

The NN is trained to minimize the loss function

$$\text{loss} = \left| f(t, \hat{x}(t), \widehat{x'}(t), \dots) \right|$$

in a set of fixed points (randomly selected in the interval previously defined).

This an unsupervised approach: there is no need of training labels $x(t)$.

$\hat{x}(t)$ can be defined to satisfy automatically the boundary conditions and ease the training.

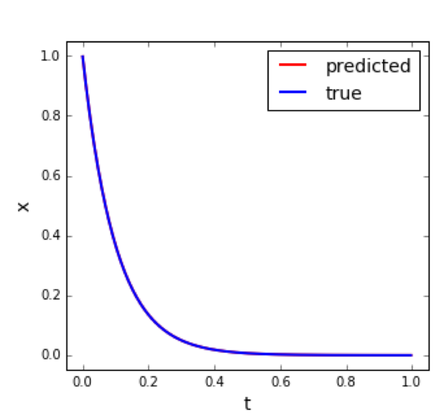
OBJECTIVE

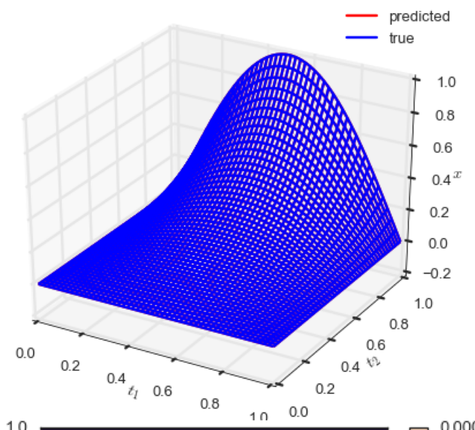
The proposed approach differ from the classical one for the following reasons:

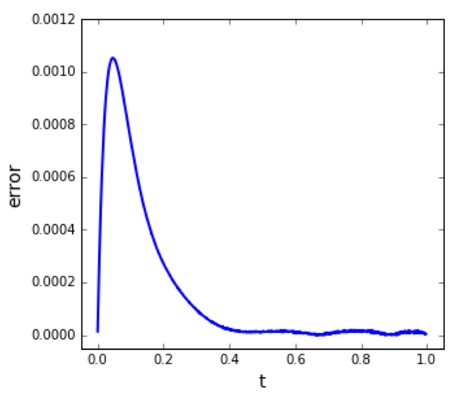
- differentiable: the solution is a neural network that is easily differentiable with relation to the input variables (and not only to the parameters) through autograd methods;
- complete: the solution is defined in the whole domain, not only in a fixed set of points;
- generic: not constrained to the type of the equation;
- robust: not heavily dependent to the differential equation;
- parallelizable: since a NN is easily parallelizable.

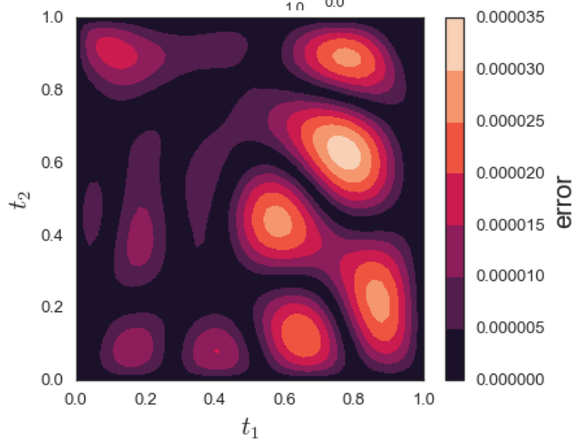
RESULTS

Results of ODE and PDE



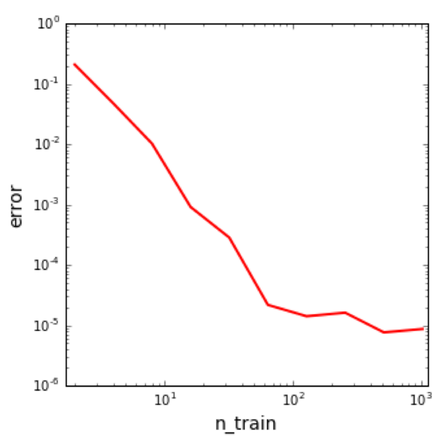


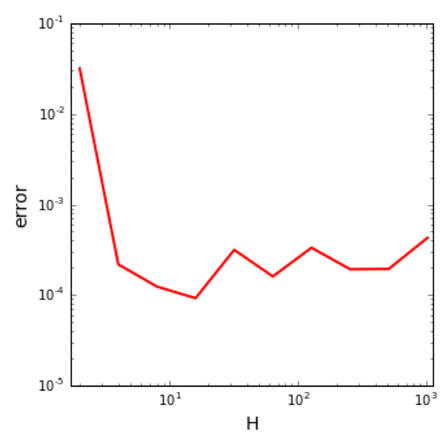




$$f(t, x, \frac{\partial x}{\partial t}) = \frac{\partial x}{\partial t} - \lambda x \qquad f(t_1, t_2, x, \frac{\partial x}{\partial t_1}, \frac{\partial x}{\partial t_2}, \frac{\partial^2 x}{\partial t_1^2}, \frac{\partial^2 x}{\partial t_2^2}) = \frac{\partial^2 x}{\partial t_1^2} + \frac{\partial^2 x}{\partial t_2^2}$$

The error is very low, thus the NN is a good approximator of the solution of the DEs.





The error decreases increasing the number of *training* points, while, increasing H, it remains stable, suggesting that a large training set is needed but too many hidden neurons are not useful.