Deep Learning for PDEs

Paolo Joseph Baioni*

March 24, 2020

[1]paolojoseph.baioni@mail.polimi.it

Acknowledgements

Contents

Introduction

The numerical solution of partial derivative equations (PDEs) plays a fundamental role in applied mathematics, science and engineering.
The recent advances in machine learning (ML) and the successes obtained by the application of these techniques in various areas suggest the possibility of using ML in solving PDEs. This approach has considerable potential compared to other numerical methods: from the possibility of writing mesh-free algorithms to that of solving data-learned equations, whose explicit functional form is unknown; however the theory is not yet developed and consequently important results of convergence and stability are missing, as well as general rules that would allow to identify the optimal parameters for the design of numerical codes. Finally, some intrinsic characteristics of the method make it considerable as a tool which is complementary to traditional numerical methods, finding application in the field of real-time control.
In this project we proposed ourselves to get into deep learning techniques and study their possible applications to PDEs, also reporting some useful theoretical results, and to implement from scratch a neural network based C++ program, so to have a low-level, detailed and complete understanding of the method.
The structure of the report is as follows.
In chapter 1 we present the general architecture of a Deep Neural Network (DNN) and the main idea of functioning of the algorithm that allows it to learn from the data (Deep Learning), consisting of two phases, called *forward propagation* and *backward propagation*. Therefore, some sector-specific issues are considered in detail, such as: distinction between train, development and test sets, problems related to overfitting, possible regularization techniques aimed at reducing it, different optimization algorithms and an overview of the main parameters that must be tuned adequately to get good results.
In chapter 2 two possible approaches to solving PDEs via DNNs are exposed, also highlighting some choices that can be made in the formulation of the problem and in treating the boundary conditions. We then deepen the comparison between DNNs and the piecewise continuous linear function which

are used as bases of finite element spaces of order one, in order to provide a greater intuition of the reasons why the DNN-based method works and to identify, albeit in this particular case, some general indications on the ideal number of nodes and layers of the DNN.
In chapter 3 we explain the programming architectural choices and the structure of the developed code, freely available on GitHub[1], as well as possible extensions.
In chapter 4, after providing instructions on how to compile, link and run the program, we show some examples by means of benchmark cases.

[1] *https://github.com/pjbaioni/neural-net*

Chapter 1

Neural Networks and Deep Learning

1.1 Architecture and operation of a Deep Neural Network

1.2 Further insights

Chapter 2

Application to PDEs

2.1 The Poisson-Dirichlet problem

2.2 Some more in-depth theoretical results

Chapter 3

Implementation of neural-net

Chapter 4

Benchmarks examples

Conclusions and further developments

Bibliography

[Andrew NG] Professor Andrew NG's Deep Learning MOOC at Coursera, *https://www.coursera.org/specializations/deep-learning*, consulted in 2019.

[cppreference.com] An online reference of the C++ language, *https://en.cppreference.com/w/cpp*

[Eigen] Eigen C++ template library for linear algebra, *https://eigen.tuxfamily.org/*

[GetPot] GetPot command line parser, *http://getpot.sourceforge.net/*

[gnuplot] A GNU graphing utility, *http://www.gnuplot.info/*

[gnuplot-iostream] A C++ interface to gnuplot, *https://github.com/dstahlke/gnuplot-iostream*

[Jinchao Xu et al.] Juncai He, Lin li, Jinchao Xu, Chunyue Zheng, *ReLU Deep Neural Networks and Linear Finite Elements*, 2018, found at *https://arxiv.org/abs/1807.03973*

[Kailai et al.] Kailai Xu, Bella Shi, Shuyi Yin, code and technical report of the project for the course CS230 *Deep Learning, Winter 2018, Stanford University*, found at *https://github.com/kailaix/nnpde*

[Kingma-LeiBa] Diederik P. Kingma, Jimmy Lei Ba,*Adam: a method for stochastic optimization*, 2015, found at *https://arxiv.org/abs/1412.6980*

[Make] GNU Make doc at *https://www.gnu.org/software/make/manual*

[Weinan-Bing] Weinan E, Bing Yu *The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems*, 2017, found at *https://arxiv.org/abs/1710.00211*