Application of Neural Network Approach to Numerical Integration

Gregory Alexandrovich Shipunov[1,*], Oksana Ivanovna Streltsova[2] and Yuriy Leonidovich Kalinovskiy[2]

[1]Dubna State University, 19 Universitetskaya St, Dubna, 141980, Russian Federation
[2]Joint Institute for Nuclear Research, 6 Joliot-Curie St, Dubna, 141980, Russian Federation

Abstract

This paper is dedicated to the description of the application of neural network approach to numerical integration of functions of one and multiple variables. The essence of the approach is to train a neural network model to approximate the integral function and than use the parameters of the model to numerically calculate the value of the integral using the formulae based on those parameters. The usage of the approach will reduce the amount of calculations (and time) required to get a numerical integration result when the number of integral function's variables is big. Where the common numerical methods become too complex the numerical approach allows calculations to be less demanding of the computational time and resources. This approach is being tested within the framework of a physics problem of modeling of the particles formation and their properties in the NICA experiment. In this experiment the key problem is to calculate integrals of functions of multiple variables. Currently the author of this paper is developing the framework for integration of functions of two variables. The main goal of the project though is to develop a Python library for numerical integration based on neural network approach.

Keywords

neural networks, numerical integration, meson, NICA, python library

1. Introduction

The problem of numerical integration appears in both theoretical and applied calculations and potentially raises two obstacles. The first is the complexity of the analytical form of the anti-derivative of the integral function (which is required to get the value of the definite integral). Sometimes it is just hard to produce the anti-derivative of the function and sometimes there are none. In general case there is no general algorithm to get a given function's anti-derivative. Methods of numerical integration exist (e.g. Simpson's method) to solve this problem by providing a formulae to get the value of an integral with some error. But when the number of the integral function's variables grow the complexity of those methods also grows as well as the errors. In this case the neural network approach to the numerical integration may help reduce the complexity as well as increase accuracy of the calculation.

Currently, there were several works done in the field of application of neural networks to the numerical integration problem. We here will highlight the paper "Using neural networks for fast numerical integration and optimization" by Lloyd et al.[1]. This work was served as a source to the development of the neural network approach. Another source was the physics problem of modeling of the particles formation and their properties in the NICA experiment which involves solving the equations containing integrals of 7-dimensional, 10-dimensional and even higher-dimensional integrals. The development of this neural network approach to the numerical integration has the main goal of the development of a Python programming

Information and Telecommunication Technologies and Mathematical Modeling of High-Tech Systems 2025 (ITTMM 2025), Moscow, April 07–11, 2025

*Corresponding author.
shgregory3@gmail.com (G. A. Shipunov); !!! (O. I. Streltsova); !!! (Y. L. Kalinovskiy)
0009-0007-7819-641X (G. A. Shipunov); 0000-0003-4522-6735 (O. I. Streltsova); !!! (Y. L. Kalinovskiy)

language software library with functionality of numerical integration with usage of neural networks approach.

This paper contains several parts. Firstly, we will give the description of the neural network approach to the numerical integration. Secondly, the physics problem of modeling of the of the particles formation and their properties in the NICA experiment will be depicted. Thirdly, we will give the observation on the results already reached in the development of the neural network approach to the numerical integration. Lastly, the insight on the future of this project will be given.

2. Neural Network Approach to Numerical Integration

The theory of the neural network approach can be summed in the following key points:

1. Let
$$I(f) = \int_G f(x)dx, \tag{1}$$
be a numerical integral of a given function $f(x)$ across region G.
2. $f(x)$ can be approximated using a MLP (multi-layer perceptron) neural network $\hat{f}(x)$.
3. $\hat{f}(x)$ can be trained to approximate $f(x)$ with an arbitrary ϵ error bound.
4. The mathematical form of $\hat{f}(x)$ can be integrated across given boundaries to produce $\hat{I}(f)$ – the value of the integral, with the following form (this from is derived from the Lloyd's et al. work[1]):
$$\hat{I}(f) = b^{(2)} \prod_{i=1}^{n}(\beta_i - \alpha_i) + \sum_{j=1}^{k} w_j^{(2)}[\prod_{i=1}^{n}(\beta_i - \alpha_i) + \frac{\Phi_j}{\prod_{i=1}^{n} w_{ij}^{(1)}}], \tag{2}$$
$$\Phi_j = \sum_{r=1}^{2^n} \xi_r Li_n(-exp[-b_j^{(1)} - \sum_{i=1}^{n} w_{ij}^{(1)} l_{i,r}]), \tag{3}$$
$$\xi_r = \prod_{d=1}^{n} (-1)^{[r/2^{n-d}]}, \tag{4}$$
$$l_{i,r} = \begin{cases} \alpha_i, & \text{if } [r/2^{n-d}]\%2 = 0 \\ \beta_i, & \text{if } [r/2^{n-d}]\%2 \neq 0, \end{cases} \tag{5}$$
where $b^{(1)}, b^{(2)}, w^{(1)}, w^{(2)}$ – neural network's parameters, $\alpha_i, \beta_i, 1 \leq i \leq n$ – integration boundaries, n – number of $f(x)$ variables.
5. The $\hat{I}(f)$ value can be interpreted as a value of the given numerical integral with an integration error bound ε.

The said MLP architecture should be described in details. The MLP neural network is a simple yet efficient neural network, which contains of several generally 1-dimensional layers l_i. Each of the layers contain 1 or many neurons and each neuron form l_i is connected to the each neuron of the l_{i+1} layer. Each neuron value can be changed using an activation function, which modifies the forward-propagated value of the neuron.

In our case the MLP architecture has 3 layers. The first, input layer, contains of n neurons, where n is the number of integral function variables. The second, hidden layer, contains of k neurons, where k is an arbitrary number greatly influenced by the neural network optimization process. This is the only layer which contains an activation function. The function is a logistic sigmoid function with the form:
$$\phi(z) = \frac{1}{1 + \exp(-z)}. \tag{6}$$

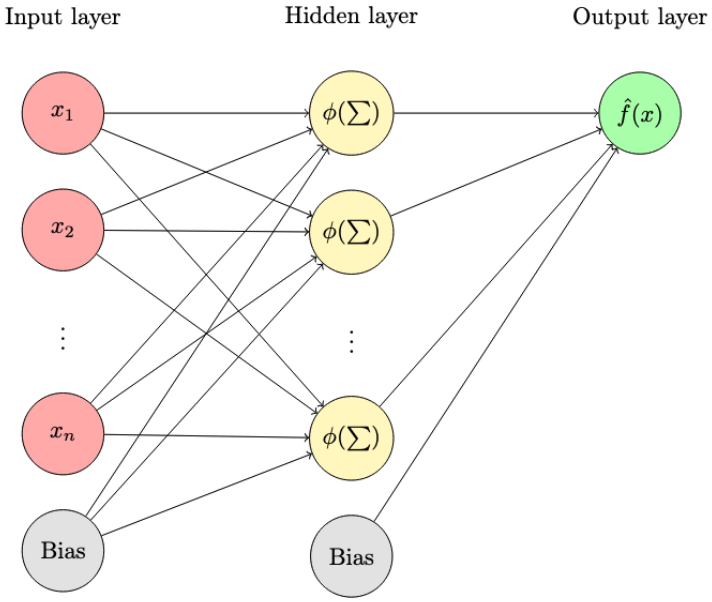

Figure 1: Described MLP structure

The third, output layer, contains only one neuron where the $\hat{f}(x)$ value is accumulated. Figure 1 shows the described architecture.

The mathematical form of this MLP structure is:

$$\hat{f}(x) = b^{(2)} + \sum_{j=1}^{k} w_j^{(2)} \phi(b_j^{(1)} + \sum_{i=1}^{n} w_{ji}^{(1)} x_i), \tag{7}$$

where $b^{(1)}, b_j^{(2)}, w_{ji}^{(1)}, w_j^{(2)}$ – neural network's parameters, k – number of hidden layer neurons, n – number of input layer neurons, ϕ – sigmoid function (6).

It is shown in item 3 in the key points of the neural network approach to the numerical integration theory that neural network $\hat{f}(x)$ can be integrated and this operation produces formulae (2-5). The integration is done quite simply, because mathematical form of the MLP contains of simple arithmetic operation except the sigmoid function. But it also can be integrated simply by substitution:

$$-Li_0(-\exp(z)) = \frac{1}{1+\exp(-z)} = \phi(z), \tag{8}$$

where $Li_0(u(z))$ is a Jonquière's function or the polylogarithm of order 0:

$$Li_0(u) = \frac{u}{1-u} = -\frac{1}{1-u^{-1}}. \tag{9}$$

With the help of this substitution the integral of the neural network $\hat{f}(x)$ of any number of variables is calculated simply using (2-5) where, as it can be clearly seen, the polylogarithm is of the same order n as the number of the integration function's variables. (NB: in Python programming language there is mpmath library which contains the required functionality to calculate such an exotic function as the polylogarithm and our numerical integration method also utilizes it.)

To conclude, the neural network approach to numerical integration is:

1. generate the dataset to train, validate and test the neural network model (7) to approximate given integral function;

2. train, validate and test (7) to approximate given integral function;
3. extract parameters of the network (7) and use them with (2-5) to calculate the numerical integral.

3. Physics task

text

4. Usage of Neural Network Approach in the Physics task

text

5. Future development

text

References

[1] S. Lloyd, R. A. Irani, M. Ahmadi, Using neural networks for fast numerical integration and optimization, IEEE Access 8 (2020) 84519–84531.