Application of Neural Network Approach to Numerical Integration

Gregory A. Shipunov[1,*], Oksana I. Streltsova[1,2] and Yuriy L. Kalinovskiy[1,2]

[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 one 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 without knowing the anti-derivative. But as the dimensions of the integral function's grow, the complexity of those methods also grows as well as the error bound. 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 an inspiration for 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

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); strel@jinr.ru (O. I. Streltsova); kalinov@jinr.ru (Y. L. Kalinovskiy)
0009-0007-7819-641X (G. A. Shipunov); 0000-0003-4522-6735 (O. I. Streltsova); 0000-0002-7596-5531 (Y. L. Kalinovskiy)

the numerical integration has the main goal of the development of a software library for the Python programming language with functionality of numerical integration based on neural network approach.

This paper contains several parts. Firstly, we will give the description of one of the possible neural network approaches to the numerical integration as well as highlight other potential approaches. 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 problem we are investigating in this paper is the calculation of approximate value of integral of a function, i.e. numerical integral with usage of neural network approach.

Let a continuous real function f be defined as $f : \mathbb{R}^n \to \mathbb{R}$. Let Ω be a compact subset of $\mathbb{R}^n$ and let G be a bounded convex subset of Ω. Let, also, $\mathbf{x}$ be a vector of n dimensions in Ω. Than

$$I(f) = \int_G f(\mathbf{x}) d\mathbf{x}, \tag{1}$$

is a definite integral of f across set G. Therefore, the problem is to get the $I(f)$ value calculated with use of neural network approach.

The work of Lloyd et al.[1] proposes, that function $f(\mathbf{x})$ can be approximated using a MLP (multilayer perceptron) neural network. Let this network contain of 3 layers with sizes n, k and 1 accordingly. n is defined above as a number of variables, k is the hidden-layer size which is set to minimize the approximation error and 1 neuron of the output layer accumulates the approximated $f(\mathbf{x})$ value. While this value is not equal to the value $f(\mathbf{x})$ with given $\mathbf{x}$, let us denote it as $\hat{f}(\mathbf{x})$ and further on we will refer to the MLP network as $\hat{f}(\mathbf{x})$. Figure 1 depicts the structure of the $\hat{f}(\mathbf{x})$ network.

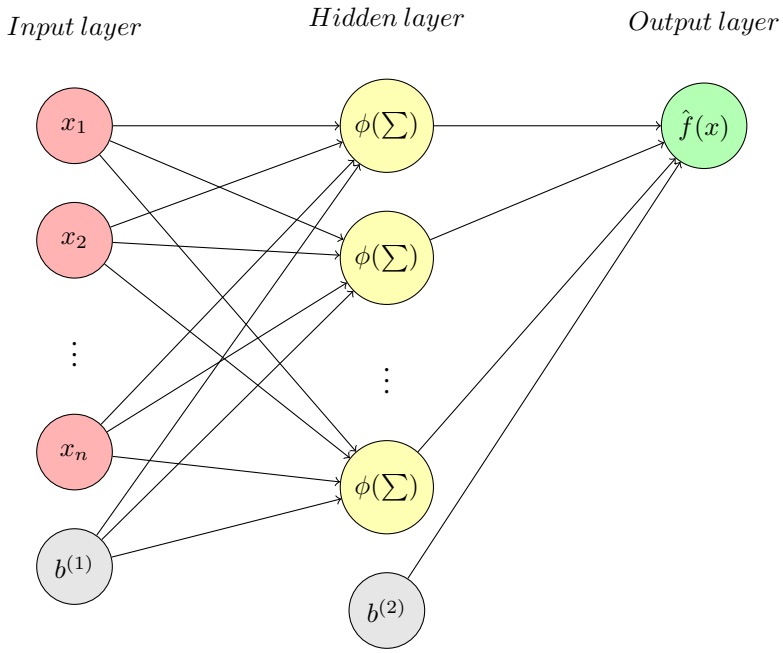


Figure 1: The MLP structure used in the neural network approach

MLP network can have activation functions applied to the neuron values before forward

propagation. In our case both input and output layer has *linear activation function* applied and the hidden layer has *logistic sigmoid function* applied, which has the following definition:

$$\phi(z) = \frac{1}{1 + \exp(-z)}. \tag{2}$$

Given network structure's mathematical form is:

$$\hat{f}(x) = b^{(2)} + \sum_{j=1}^{k} w_j^{(2)} \phi(b_j^{(1)} + \sum_{i=1}^{n} w_{ji}^{(1)} x_i). \tag{3}$$

So here Lloyd's et al. work puts the proposition to use the trained neural network $\hat{f}(\mathbf{x})$ to get the approximate value of integral (1):

$$\hat{I}(f) = \int_G \hat{f}(\mathbf{x}) d\mathbf{x}. \tag{4}$$

To integrate the (1) and get the value of $\hat{I}(f)$ let us apply following substitution:

$$-Li_0(-\exp(z)) = \frac{1}{1 + \exp(-z)} = \phi(z), \tag{5}$$

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

$$Li_0(u) = \frac{u}{1 - u}. \tag{6}$$

This substitution is required, because it is easy to integrate polylogarithm functions. With this in mind, the Lloyd's et al. propose this formulae as a numerical method to integrate $f(\mathbf{x})$ over given $G : [\alpha_1, \beta_1] \times ... \times [\alpha_n, \beta_n]$.

1. Let
$$I(f) = \int_G f(x) dx, \tag{7}$$
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]):

$$\hat{I}(f) = b^{(2)} \prod_{i=1}^{n} (\beta_i - \alpha_i) + \sum_{j=1}^{k} w_j^{(2)} [\prod_{i=1}^{n} (\beta_i - \alpha_i) + \frac{\Phi_j}{\prod_{i=1}^{n} w_{ij}^{(1)}}], \tag{8}$$

$$\Phi_j = \sum_{r=1}^{2^n} \xi_r Li_n(-exp[-b_j^{(1)} - \sum_{i=1}^{n} w_{ij}^{(1)} l_{i,r}]), \tag{9}$$

$$\xi_r = \prod_{d=1}^{n} (-1)^{[r/2^{n-d}]}, \tag{10}$$

$$l_{i,r} = \begin{cases} \alpha_i, & \text{if } [r/2^{n-d}]\%2 = 0 \\ \beta_i, & \text{if } [r/2^{n-d}]\%2 \neq 0, \end{cases} \tag{11}$$

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:
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:

$$\hat{f}(x) = b^{(2)} + \sum_{j=1}^{k} w_j^{(2)} \phi(b_j^{(1)} + \sum_{i=1}^{n} w_{ji}^{(1)} x_i), \tag{12}$$

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 (2).

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 (8-11). 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:

$$-Li_0(-\exp(z)) = \frac{1}{1+\exp(-z)} = \phi(z), \tag{13}$$

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

$$Li_0(u) = \frac{u}{1-u}. \tag{14}$$

With the help of this substitution the integral of the neural network $\hat{f}(x)$ of any number of variables is calculated simply using (8-11) 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.) The whole calculus of the integral of (12) is a broad topic with requirement of its own theory but it is beyond the scope of this paper, but it may be observed in Lloyd's et al. work[1].

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

1. generate the dataset to train, validate and test the neural network model (12) to approximate given integral function;
2. train, validate and test (12) to approximate given integral function;
3. extract parameters of the network (12) and use them with (8-11) to calculate the numerical integral.

3. The problem of modeling of the particles formation and their properties in the NICA experiment

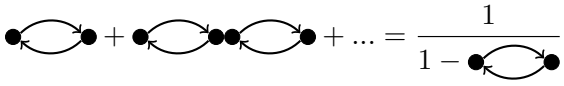
It is mentioned in the introduction to this paper, that the source to the development of the neural network approach was the problem of modeling of the particles formation and their properties in the NICA experiment.

The main task of this problem is to model particle formation ant their properties in the NICA experiment. The problem requires to build a model, which will describe the properties of light and heavy mesons. This will be a nonlocal interactions model!!!!!![2][3]. The model concludes to the fact that meson (quark-antiquark pair) formation proceeds with a gluon exchange. But the gluon exchange cannot be described as a diagram sum, because perturbation theory cannot be applied here!!!!!!!.

Let ● be the *form-factor* and let us use Gaussian-type model:

$$F(p^2) = \exp(\frac{-p^2}{\lambda^2}), \tag{15}$$

so the form-factor size will be defined by the $F(p^2)$ as shown above (15). To describe the particles we need to know masses, decay constants and the decay process. Let us denote the interaction diagram as  and than we can use it in the equations:


$$= \frac{1}{1 - \text{}}, \tag{16}$$


$$\rightarrow \int_0^\infty \frac{dp}{2\pi^4} F(p^2) \frac{1}{u_1 u_2} = \int_0^\infty \frac{dp}{2\pi^4} F(p^2) \frac{1}{[(p+q_1)^2 + m_1^2][(p+q_2)^2 + m_2^2]}. \tag{17}$$

To calculate these integrals let us use the equations:

$$\frac{1}{(p+q_1)^2 + m_1^2} = \int_0^\infty dt\{\exp(-t[(p+q_i)^2 + m_i^2])\}, \tag{18}$$

$$F(p^2) = \int_0^\infty ds\{\exp(-sp^2)F(s)\}. \tag{19}$$

The cases of more than two mesons is described using triple and quadruple integrals.

The amplitudes of particular decays and the squared amplitudes of particular decays are calculated using FORM (a software for analytical calculation). And the key problem is to calculate integrals of functions of multiple variables.

In the framework of this problem there is a particular iterated integral:

$$\int_0^1 d\alpha\{\alpha^a(1-\alpha)^b\} \int_0^\infty dt\{\frac{t^m}{(1+t)^n} F[z_0]\} \equiv I(a,b,m,n;F[z_0]), \tag{20}$$

$$F[z_0] = \exp[-2z_0], \tag{21}$$

$$z_0 = tD + \frac{t}{1+t}R^2, \tag{22}$$

$$D = \alpha_1(b_1^2 P^2 + m_1^2) + \alpha_2(b_2^2 P^2 + m_2^2), \tag{23}$$

$$R^2 = (\alpha_1^2 b_1^2 + \alpha_2^2 b_2^2 + 2\alpha_1\alpha_2 b_1 b_2)P^2, \tag{24}$$

$$b_1 = -\frac{m_1}{m_1 + m_2}, \tag{25}$$

$$b_2 = \frac{m_2}{m_1 + m_2}, \tag{26}$$

$$\alpha_1 = \alpha, \alpha_2 = 1 - \alpha. \tag{27}$$

This integral is used in multiple calculations in the problem. As it can be seen in (20) it is denoted as $I(a, b, m, n; F[z_0])$, and with different combination of parameters it is used multiple times in the calculations. The integral (20) also is interesting in that case, that it depends on two variables only, so it is can be used to test the integration framework we've been developing early in the development cycle.

4. Usage of Neural Network Approach

As stated in the previous part, the integral (20) has been already used to test the neural network approach.

It was firstly interpreted as a product of two integrals of single-variable functions ($\alpha_{1,2}$ were treated as constants). The first integral of function $\alpha^a(1-\alpha)^b$ has a very simple analytical anti-derivative with any integer parameter a and b so it was easy to check the result of the integration. In most cases the neural network approach provided error bound $\varepsilon < 10^{-5}$. On the other hand, the second integral of function $\frac{t^m}{(1+t)^n}F[z_0]$ has a much more complex anti-derivative, so in this case the goal was to make the approximation bound of neural network $\hat{f}(x)$ was $\epsilon < 10^{-5}$ and this requirement was met.

Secondly, the iterated integral (20) was interpreted as a double integral:

$$\int_0^1 \alpha^a(1-\alpha)^b d\alpha \int_0^\infty \frac{t^m}{(1+t)^n} F[z_0] dt = \int_0^1 \int_0^\infty \frac{t^m}{(1+t)^n} F[z_0] \alpha^a (1-\alpha)^b dt d\alpha, \tag{28}$$

and this double integral function was used to train the neural network model to approximate it. The approximation error bound of $\epsilon < 10^{-5}$ was also met, but currently the software implementation of the formulae (8-11) requires further improvement and testing.

Also both 1-dimensional ans 2-dimensional functionality was tested on simple arbitrary functions, e.g. transcendental functions.

5. Future development

Currently, the further development of the neural network approach to the numerical integration continues. The resulting software will be tested on the integrals from the meson interaction modeling problem, which provides plenty of cases where numerical integration of high-dimensional functions is required. Also, we hope, the final product - the software library for numerical integration with use of numerical integration approach called *Skuld* will help to solve difficult calculation problems in the meson interaction modeling problem domain as well, as in other fields, both theoretical and applied.

6. Acknowledgments

Authors of this work express their acknowledgments to Sara Shadmehri and Tatevik Bezhanyan from Meshcheryakov Laboratory of Information Technologies, JINR for their advices both in the field of neural networks and English language.

References

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

[2] D. Blaschke, H. Grigorian, Y. L. Kalinovsky, Meson form-factor scheme for the chiral lagrangian approach to j/ψ breakup cross sections motivated by a relativistic quark model, Physics of Particles and Nuclei Letters 9 (2012) 7–17.
[3] P. Costa, M. C. Ruivo, Y. L. Kalinovsky, Pseudoscalar neutral mesons in hot and dense matter, Physics Letters B 560 (2003) 171–177.