IJHSR
ijhighschoolresearch.org
RESEARCH ARTICLE

Calculation of the approximate value of an arbitrary function on a limited interval by the Chebyshev rounding method using Back-Propagation Neural Network

Illarion Illarionov Zervas

Lyceum 'Second School' n.a. V.D. Ovchinnikov in Moscow, Russia, 119333 ; illarionillarionovzervas@gmail.com

ABSTRACT: This research paper is devoted to including new methods of approximation in Theory Of Approximation using Chebyshev Polynomials. The research focuses on creating a potential model for approximation using Neural Networks. This work enables mathematical analysis of the Chebyshev polynomials and activation functions. One of the goals of this paper was to minimize the time spent on the learning process. However, implementing every term of Chebyshev Polynomial could only be realized in some frameworks.

KEYWORDS: Machine learning, Algebra, Interpolation, Approximation, Chebyshev polynomials.

Introduction

Artificial neural networks (ANNs) and their machine learning have an excellent ability to learn and generalize based on data, which provides a wide range of applications with excellent development capabilities to meet simple everyday needs. Among various computing schemes, analog information processing using dynamic systems is of increasing interest since this approach allows us to effectively solve various problems in many branches, including physics and engineering. Neural Networks (NN) could also be used in mathematical problems. Even though function approximation has been an urgent problem for centuries, implementing NNs has never been compared with any arbitrary function using Chebyshev's approximation. This leads to a well-established algorithmic path to find an approximate value. At the same time, it is worth noting that some functions cannot be recognized due to their complexity and insufficient data on the construction and theory of their origin.

In situations where it is necessary to replace a function on a specific segment with an approximating polynomial, in which the deviation of the function from the polynomial on a given set will be the smallest, it is proposed to use the theory of the best approximation of functions created by Chebyshev. Approximating polynomials using Chebyshev's polynomials (CPs) have several remarkable properties. Chebyshev's polynomials make it possible to construct a generalized polynomial with the required accuracy but shorter than the existing polynomials. This is more convenient for calculations; the decomposition converges faster than other polynomials.

In this type of problem, a Back-Propagation Neural Network (BPNN) with Feed-Forward learning could find a solution by analyzing arbitrary functions on a geometric plot. The purpose of this work will be to determine the ability of a neural network to calculate the value of any arbitrary function using Chebyshev's approximation.

Annotation:

Chebyshev polynomials were discovered by Pafnuty Lvovich Chebyshev (1821-1894). He first announced his invention at the St. Petersburg Academy in 1853 in his memoirs "The Theory of Mechanisms known as parallelograms," inspired by communication with European scientists, namely French mathematicians and geometricians.[1] At the same time, this work did not fully reveal the details. It contained only mathematical ideas and theories, which contributed to the creation of his second work, "On the issues of minima related to the approximate representation of functions", in which he revealed the approximation and the interpolation method.[2] Chebyshev was the first person who fully understood the concept of orthogonal polynomials.[3]

In the 19th century, interpolation and approximation were actively used in Europe in such areas as the construction of geographical maps, rules for determining approximate distances on the earth's surface, and the kinematics of mechanisms. Orthogonal polynomials are an important part of computational mathematics.[4] Later in the 20th century, they began to be actively used by computers in developing antenna arrays for industries such as radio astronomy, aviation, telecommunications, etc.

Studying the topic of approximation of trigonometric functions, a gap was identified in the part of training neural networks by Chebyshev polynomials; it was this section that interested our team. This paper will operate on the first type of Chebyshev polynomials. Suppose a successful neural network training option is found. In that case, a more accessible way of calculating complex mathematical operations without using computers opens up, particularly when creating water filtration systems with complex structures.

The first kind could be represented in two ways:

Recursive*:

$T_0(x) = 1$,

$T_1(x) = x$,

DOI: 10.36838/v5i6.4

*The recursive representation works only in segment x ∈ [-1, 1]. Out of this range, the polynomial will grow to +∞ or -∞(depending on the parity of degree of the function)

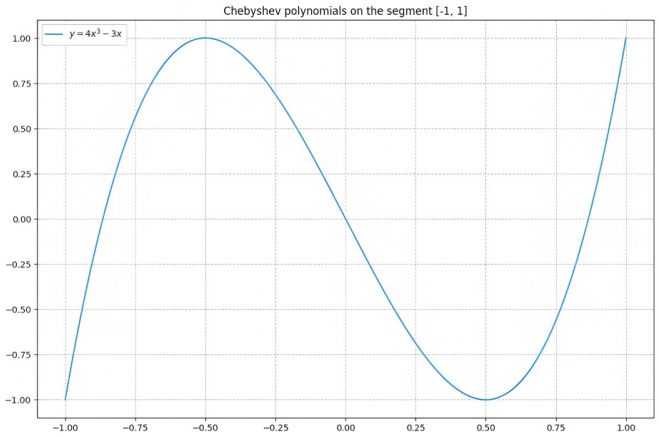

Figure 1: Chebyshev Polynomial: n = 4.

On the plot represented the $T_4(x)=4x^3-3x$. On the particular segment, the function is similar to $f(x) = sin(x)$. However, outside the specific range function has nothing similar to any trigonometric function:

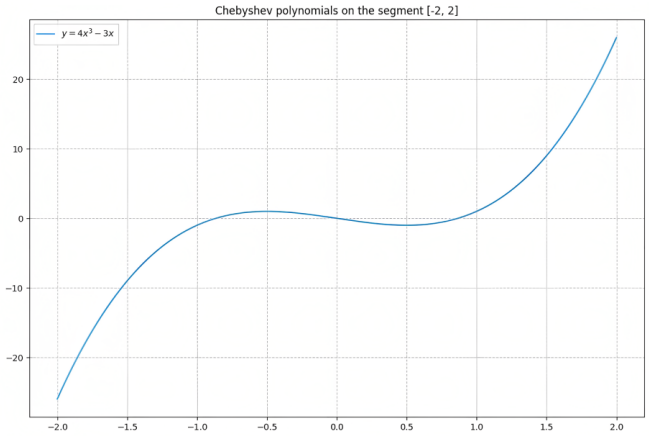

Figure 2: Chebyshev Polynomial: n = 4 on the segment [-2, 2].

Trigonometric:*

$T_n(x) = cos(n*arccos(x))$

*Also works only on the segment of [-1, 1].

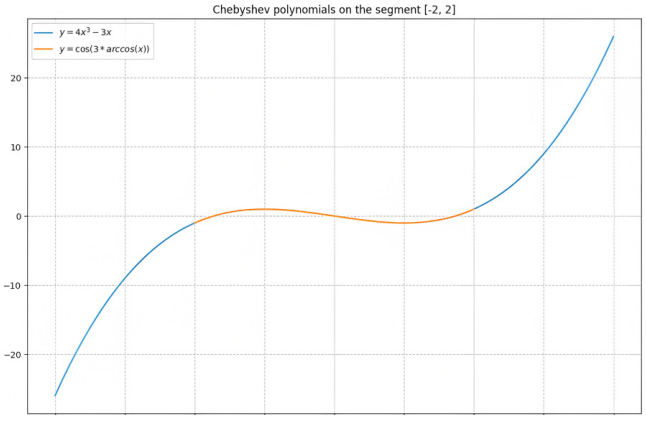

Figure 3: Chebyshev Polynomial: n = 4, trigonometric and recursive forms.

The graph shows a 100% similarity with both representatives on the segment [-1, 1]. The trigonometric version of Chebyshev polynomials gives us an understanding of polynomial behavior. When $n = 1 \rightarrow f(x) = cos(arccos(x)) = x$. The $arccos(x)$ is a limit for a function on the segment [-1, 1]. With the increase of the n, the frequency of the cosine function will also grow.

There are Chebyshev polynomials from 3rd to 7th degree:

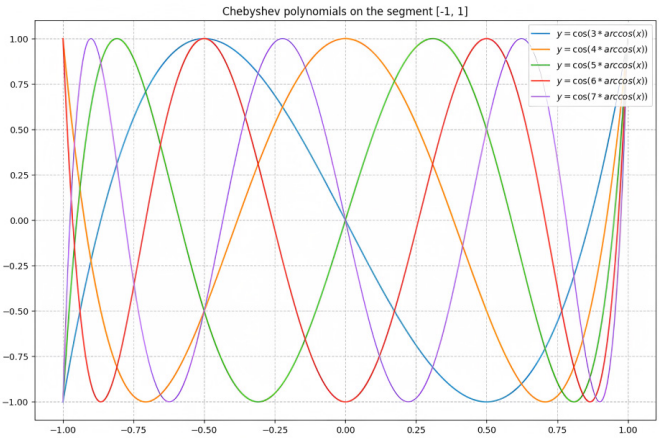

Figure 4: Chebyshev Polynomials: n = [3, 7], trigonometric representatives

Literature Review

There are a few works that are trying to analyze the problem of calculation of Chebyshev's polynomials.[5-7] Their main goal is to find approximate solutions using computational methods. A large number of research papers in this field, which you can find in open space, is connected with the theoretical usage of the Chebyshev experience, which includes interpolating an arbitrary function. Nowadays, the problem of function interpolation is old and multifaceted. With the increased popularity of neural networks, the discussion for creating new solutions with a higher precision is as relevant as ever. There are already important papers in this field of research. For instance, the paper included Chebyshev Approximation(CA) for the Wine problem using Modified Multi-Output Chebyshev-Polynomial Feed-Forward Neural Network(MOCPFFN) MOCPFFN includes N input layers with M output layers, which is similar to the BPNN model.[7] Researchers from China University wanted to find a correlation between the quality of wine and geographic position. Few works generally describe problems with calculating Chebyshev polynomials using NN models.[8,9]

Notwithstanding all that, there are already tons of works assigned to solving mathematical problems, such as generic algorithms or finding the divider of an arbitrary number, and many others in different areas of math. The amount of this research gives an understanding of NN's potential in solving practical problems. NN was already used in the approximation of functions. One of the most popular and earliest was the Fourier Neural Network.[10] Despite this, the similarity with our problem is only with a particular topic; the calculation algorithm has a different structure. Sometimes, it is

tion. Algorithmic solution for approximation is well-known; however, trained NN could outperform classical interpolation solution. That's the case where NN could be more efficient than algorithms by analyzing geometrical plots.

Well-trained feed-forward NN would recognize patterns much faster and with higher accuracy. This work is devoted to clarifying the potential of NN in approximation algorithms.

Methodology

Function Approximation:

The function approximation is a method to simplify the difficult polynomial to well-known small-term polynomials.

The CPs are perfectly fit for the function's simplification because sigmoidal functions are easier to structure. The opportunity to change the frequency of the cosine function would replace any complex functions (with the condition of future CP expansion).

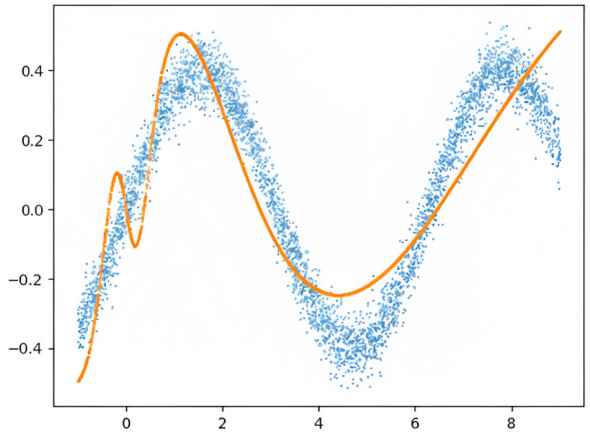

Figure 5: NN Interpolation using Chebyshev polynomials.

Main aspects:

Chebyshev's approximation is known for finding solutions on intervals from -1 to 1 for n-degree polynomials, which could be expanded to [-∞, +∞] (by Pade Approximant[11]). But, our goal is to recognize the function on the geometry plot and find its value as precisely as possible. We could achieve that by training our NN for n degrees of CP. This way, you can approximate any function for $T_n(x)$ and find its value in every point (or expand it).

Coding Aspects:

Our team used the Python library "Keras," commonly used for building BPNN with highly precise settings. Using this particular method gives opportunities for different types of activations and approximations.

Different activation functions could be used for approximation of a function. The functions that could be used for approximation of sinusoidal functions are sigmoid and the hyperbolic tangent. However, some of them are extremely sensitive to parity.

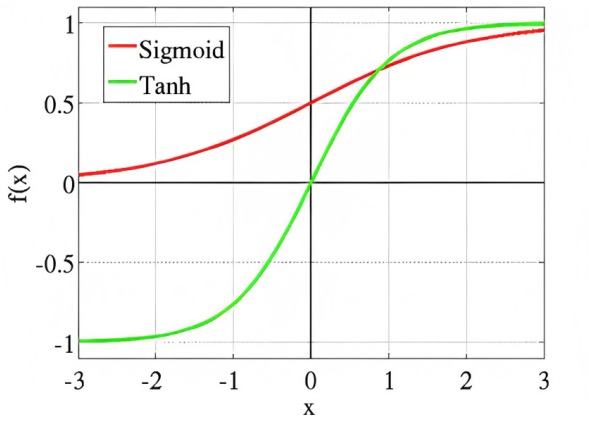

Figure 6: Comparison of tanh and sigmoid functions.

Where sigmoid function represented as $f(x) = \frac{1}{e^{-x}+1}$ and tahn $f(x) = \frac{e^{x}-e^{-x}}{e^{-x}+e^{x}}$

The problem of parity could be caused by missing terms in TS. This could be fixed by using the ReLU activation function. ReLU is represented in this form: $f(x)=max(0,x)$.

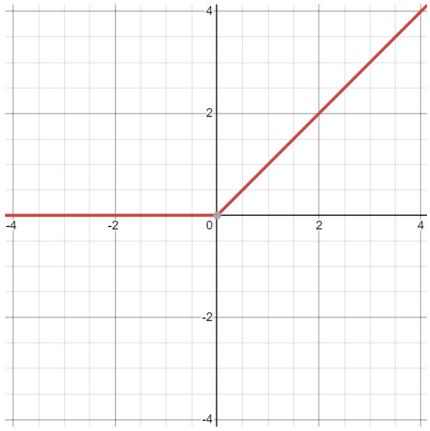

Figure 7: ReLU function.

Usage of this activation function could recover terms in TS and remove unexpected errors that pop up during the learning process.

Results and Discussion

The problem has plenty of details that are important to consider. That's why a model should be as precise as possible. There are important points which we should comply with:

- CPs could be divided into even and odd functions. That is significant to take into account the origins of activation functions. For instance, the hyperbolic tangent is an odd function that could give us issues analyzing even functions. That also leads to the expansion of a polynomial in a Taylor series. The goal is to avoid mistakes connected with wrong expansion in a TS where some terms could be missed and, as a result, break the structure of polynomials. Activation functions are influenced by the parity of the degree of the polynomial.

For instance, the CP of 3rd term in the Taylor series has the next representative:

$$f(x) = -3x+4x^3$$

However, the 4th term of CP:

DOI: 10.36838/v5i6.4

$$f(x) = 1\text{-}8x^2\text{+}8x^4$$

From this representative, it is evident that functions have different parity, which could lead us to problems with pattern recognition.

- The best way to describe CPs is the cosine function, which increases frequency with each degree of the polynomial. The nearest approximation for CPs is sigmoid and tanh functions. As mentioned earlier, it's important to feel the parity of function. Tanh and sigmoid functions are of different parity. We should consider the usage of only one type of activation function(a.f.), which could increase MSE and throw out bugs during the testing period.

- On the other hand, combining different activation functions without any balance and structure is useless. We tried to combine two layers of ReLU and two layers of sigmoid in our model. However, it's not that obvious due to the complexity of NN. In some cases, we met the impossibility of unification of some layers. The simultaneous implementation of "ReLU" and "tanh" activation functions led to missing every odd degree, giving only a straight line. The graph below is a photo from the first experiments, illustrating a perfect example of this type of mistake.

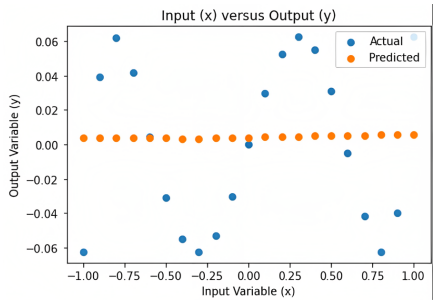

Figure 8: The problem of missed degree in Taylor series expansion.

- We are also dependent on time because CA already has an algorithmic solution. However, it is time-consuming and requires high-capacity computers. Our goal is to find a solution that minimizes time.

The model consists of 3 hidden layers with different types of activation and amounts of neurons(which is a variation for rational numbers on the segment [0, 350]). Our model works perfectly on the segment [0, 20] in degrees of CP. The accuracy also influences our model because a high Mean Squared Error(MSE) will cause incapability of our project. MSE should be at most 2-3 %. Otherwise, the approximation wouldn't be as precise as we need.

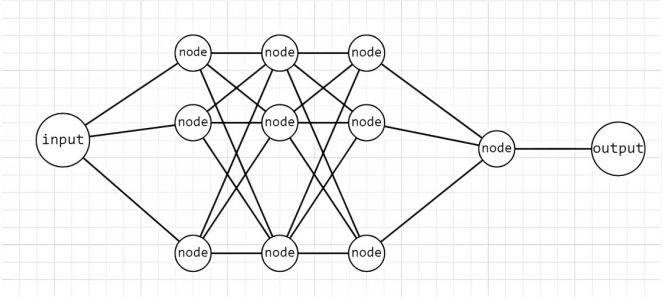

Figure 9: The model of NN based on Chebyshev polynomials.

In this figure, we demonstrate the concept of our model. From the input, layer data divides into "n" amounts of neuron groups with different node activations. In our model, hidden layers are placed in the following order: sigmoid, sigmoid, and sigmoid. The choice of these layers will be explained in the next chapter.

The goal is to find an analytic model for best approximation. Summarizing all the issues above, composing a perfect model for a function is important. The prediction should be working for any degree and term of CPs. However, our predictions were wrong, and the model Sigmoid-ReLU-Sigmoid-ReLU(SRSR) with 20 nodes in each layer didn't work for odd functions. Misunderstanding the roots of this behavior led us to increase epochs, which did not affect the current situation. We tried to analyze the behavior of our functions and marked that ReLU should not recover missed terms in TS. In other words, the ReLU activation function only interrupted the learning process and denied any sigmoid pattern on the segment [-1, 0]. From this point, our model will not conclude the ReLU activation function. Deliverance of different activation functions indicated that the sigmoid activation function perfectly works for any parity of functions. Anyways, the amount of the terms of CPs where MSE lower than 3% decreased to eleven. In a small amount of research, it was discovered that the epochs were at least 3000. The configuration of 3 layers of Sigmoid-Sigmoid-Sigmoid (SSS) achieves our goal.

"Approximation theory" uses different methods to approximate functions. Chebyshev's method of approximation should be used on small degrees of CPs. Anyways, our NN could be expanded to any term of CPs. It's important to mention that increases in degree do not affect overall time. The average time spent on learning for ten terms of the CPs is around 180-250 seconds for 2500 epochs.

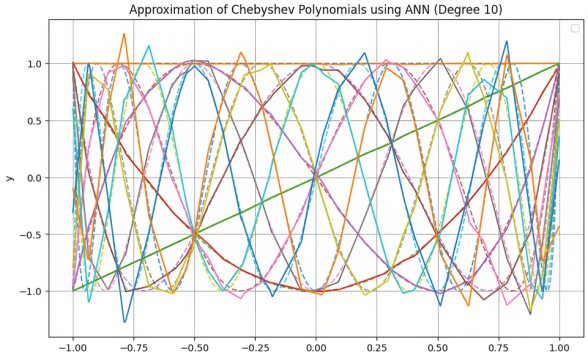

Figure 10: The final results of the plot. The neural network can predict the first 10 CPs with an average of 9% MSE.

Conclusion

This research found a model that accurately approximates the first degrees of CPs. One of our theses implied a non-static amount of layers. Through our work, we found out that more does not mean better, especially in ML. That is why we changed our focus from layers question to time question due to the issue's relevance. The result of time minimization, which we achieved, is promising but should be overlooked. Two hundred fifty seconds is the average time for learning

any term of CPs. This result is not the fastest one, and this problem is still urgent.

The intermediate outcome of comparing "tanh" and "sigmoid" a.f. was intriguing, and further research should be done. Our team mentioned that the "tanh" activation function has lesser MSE than the "sigmoid" activation function. Despite this, on higher terms of CPs, the "tanh" a.f. was a total outsider in contrast to the "sigmoid" a.f., which was much more precise. It's also important that it is connected with the even degrees of the functions. The result could be used as an example of a better approximation of sinusoidal functions with high frequency. The main goal of this paper was to take a new perspective on the theory of approximation. Implementing ML in approximation theory still needs to be explored due to the need for more work in this sphere. The infrequent usage of NN in computational science or theory of Approximation could be explained by already existing algorithms, which require only high machine capacity. The potential of NN has not been reached in this mathematical sphere. The topic of approximation with NN could be explored further. The focus of this work was more on approximation than on interpolation using Chebyshev polynomials. A separate study requires a deeper consideration of the interpolation of functions using CP since the unexpected discovery and optimization of algorithms in this area will improve the entire production sphere of the world. Looking at the question of approximation through Chebyshev polynomials, it is worth considering other models of the structure of NN. The interpolation using the CP question should be the next topic of the research. From the results shown, it is evident that approximation by NN is possible. NN was trained for different types of interpolation (such as data or function interpolation). However, NN wasn't trained on n-dimensional interpolation. Only in 2022, the problem of 4D interpolation was solved. The idea of introducing NN Chebyshev interpolation and expanding it to n dimensions is still relevant.

To sum up, we found a potential way to imply the theory of approximation by using BPNN. The results of this research could enhance the experience of engineering high-accuracy technologies and include new methods in computational mathematics.

Acknowledgments

I would like to express my sincere gratitude to Dr. Eric Sakk for his invaluable guidance and support throughout the study. His experience, meaningful feedback, and dedication greatly contributed to the success of this work. I am grateful to him for the opportunities to expand my knowledge and for his mentoring, which played a significant role in shaping the direction of this research project.

References

1. Chebyshev PL. *Théorie Des Mécanismes Connus Sous Le Nom De Parallélogrammes*. St.-Pétersbourg: Imprimerie de l'Académie impériale des sciences; 1853.
2. CHEBYSHEV, P. L.: Sur les Questions de Minima qui se Ratachent a la Représentation Approximative des Fonctions. Mem. Imperial Acad. Sci. St. Petersburg, 6th Ser., Math. and Phys.7, 199–291 (1859).
3. V. L. Goncharov. 2000. The Theory of Best Approximation of functions. J. Approx. Theory 106, 1 (Sept. 2000), 2–57. https:// doi.org/10.1006/jath.2000.3476
4. G.M. Kobelkov. 2020. Numerical Methods, pt. 1.
5. *Scott A. Sarra* Chebyshev Interpolation: An Interactive Tour // Journal of Online Mathematics and Its Applications
6. A. B. Bogatyrev, "Effective computation of Chebyshev polynomials for several intervals," Sb. Math., 190:11 (1999), 1571–1605
7. L. Jin, Z. Huang, Y. Li, Z. Sun, H. Li and J. Zhang, "On Modified Multi-Output Chebyshev-Polynomial Feed-Forward Neural Network for Pattern Classification of Wine Regions," in *IEEE Access*, vol. 7, pp. 1973-1980, 2019, doi: 10.1109/ACCESS. 2018.2885527.
8. Convolutional Neural Networks on Graphs with Chebyshev Approximation, Revisited: arXiv:2202.03580
9. Training Neural Networks for and by Interpolation: arXiv:1906.0566
10. Ngom, M., Marin, O., Fourier neural networks as function approximators and differential equation solvers, Stat Anal Data Min: *The ASA Data Sci Journal*. 14(2021), 647– 661.
11. Weisstein, Eric W. "Padé Approximant." From MathWorld--A Wolfram Web Resource
12. V.A. Ilyin, V.A. Sadovnichiy, Bl. H. Sendov "Mathematical Analysis" Part 1, Ed. 3, ed. A.N. Tikhonov, Prospect 2004.

Author

Illarion Illarionov-Zervas is a high school student of the Faculty of Computer Science of the Physics and Mathematics Lyceum "Second School" in Moscow, Russia. He plans to specialize in machine learning and is engaged in research in computer science, applying mathematical methods to a neural network.

DOI: 10.36838/v5i6.4