Best Neural Network Approximation by using Bernstein Polynomials

with GRNN Learning Application

Hawraa Abbas Almurieb

Department of Mathematics, College of Education for Pure Sciences, University of Babylon, Hilla, Iraq

Anwar Anwer Hamody

Hudaibiya School, Babylon Education Directorate, , Hilla, Iraq

الخلاصة

تعد كثيرات حدود برنشتاين إحدى أهم الأدوات الرئيسية لتقريب الدوال. علاوة على ذلك ، تحتوي الشبكات العصبية على العديد من التطبيقات المفيدة في التقريب وغيره من المجالات. ندرس في هذا البحث، إمكانية الاستفادة من خصائص متعددات حدود برنشتاين لتعريف صيغة جديدة من الشبكات العصبية ، والتي يمكن أن تتلاءم مع الدوال التقريبية مع درجة تقريب بدلالة معامل الاستمرارية. من ناحية أخرى ، الشبكات العصبية هي تقديرات تقريبية عامة ، لذلك يمكن اعتبار الدوال كشبكات عصبية باستخدام التقريب العددي. لهذا الغرض ، نستخدم خوارزمية GRNN لتقريب الدوال بشكل منتظم باستخدام برنامج الماتلاب ، مع إعطاء بعض الأمثلة التي تؤكد جودة التقريب.

الكلمات المفتاحية:

الشبكات العصبية ، متعددات حدود بيرنشتاين، التقريب الدالي، معيار الاستمرارية، التعلم.

Abstract

Bernstein polynomials are one of the first and main tools for function approximation. On the other hand, neural networks have many useful applications in approximation and other fields as well. In this paper, we study how we benefit from properties of Bernstein polynomials to define a new version of neural networks, that can be fit approximating functions in terms of modulus of continuity. Numerically, we use neural networks to approximate some types of continuous functions. For that purpose, we use GRNN algorithm to approximate functions uniformly by using Matlab, giving some examples that confirm good rate approximation.

Keywords:

Neural networks, Bernstein Polynomials, Function Approximation, Modulus of Continuity, GRNN, Learning.

1. Introduction

In many papers, authors define many formulas of neural networks for purposes of function approximation. For examples, (1–6) studied the existence of best function approximation in C[a,b] and Lp spaces by neural networks with several formulas. When a new formula is defined, it should have the appropriate properties to best fit the target function. Moreover, the degree of that best approximation is estimated in terms of several criteria, especially modulus of continuity. Now, we begin with some main definitions about approximation,

Definition 1.1. (7) Let $(X, \|\cdot\|)$ be a normed space, then the best approximation of $x \in X$, from $Y \subseteq X$, is $y_0 \in Y$, that satisfies

$$\|x - y_0\| = \inf_{y \in Y} \|x - y\|. \tag{1}$$

The uniform norm (7), is the one that we use here, which is given by

$$\|\boldsymbol{x}\|_\infty = \max_{1 \le i \le n} |x_i|, \tag{2}$$

for any $\boldsymbol{x} = (x_i)_{i=1}^n \in R^n$.

If we consider $C[a, b]$, to be the set of continuous functions on $[a, b]$, then the following norm over any $f \in C[a, b]$ is given by

$$\|f\|_\infty = \max_{a \le x \le b} |f(x)|, \tag{3}$$

Function approximation begun with polynomials by P. L. Chebyshev (8), the problem is to find some polynomial on $[a, b]$ that minimize

$$\max_{a \le x \le b} |f(x) - p(x)|. \tag{4}$$

Later, many forms of polynomials were built to get the best approximation as in (4) $\max_{a \le x \le b} |f(x) - p(x)|$. , see for example (9–11), they approximated continuous functions with several polynomials, famously, Bernstein polynomials (10). Bernstein introduced his polynomials that best approximate continuous functions, and later it was improved with modulus of continuity, for more details, see (7).

Definition 1.2.(7) The general form of Bernstein polynomials is given by

$$\big(B_n(f)\big)(x) = \sum_{k=0}^{n} f\left(\frac{k}{n}\right)\binom{n}{k} x^k (1-x)^{n-k}, \tag{5}$$

for $0 \leq x \leq 1$.

On the other hand, neural networks are universal approximations for any continuous function by Cybenko (12), he used very primitive formula of neural network in his earliest proof. Until today, one can find many versions of neural networks among papers (3,13–17). Some defined new activation functions, while others gave special weights that fit their needs for function approximation.

Few little papers dealt with neural networks via Bernstain polynomials, far away from approximation approaches. We define a special formula of neural network on $[-1,1]^d$, as follow

Definition 1.3. For any input $\boldsymbol{x}$ and a weight $\boldsymbol{w}$, *s.t.* $\boldsymbol{x}, \boldsymbol{w} \in [-1,1]^d$, define the neural network

$$N_n(\boldsymbol{x}) = \sum_{j=0}^{d}\sum_{k=0}^{n} c_{j,i}\, \sigma_j\big(\langle \boldsymbol{w}.\boldsymbol{x}\rangle + b_j\big)\,, \tag{6}$$

where $\langle \boldsymbol{w}.\boldsymbol{x}\rangle$ is the inner product between the vectors $\boldsymbol{w}$ and $\boldsymbol{x}$, $c_{j,i} = \frac{(-1)^i\, n!}{i!(n-k)!(k-i)!} f\left(\frac{k}{n}\right)$, $j = 1, \ldots\ldots, d$, $k = 0, \ldots\ldots, n$, and 1_j and $x_j \in [-1,1]$, are the jth component of $\mathbf{1} = (1_1, \ldots, 1_d)$ and $\boldsymbol{x} = (x_1, \ldots, x_d)$, respectively. The activation function of jth component is given by

$$\sigma_j(\boldsymbol{x}) = x_j^k\big(1_j - x_j\big)^{n-k}, \tag{7}$$

Set $\varkappa$ to the set of neural networks of the form $Nn(\boldsymbol{x}) = \sum_{j=0}^{d}\sum_{k=0}^{n} c_{j,i}\, \sigma_j\big(\langle \boldsymbol{w}.\boldsymbol{x}\rangle + b_j\big)$,).

Definition 1.4. The best neural approximation of $\boldsymbol{f} \in C^d[-1,1]$ is $N_0 \in \varkappa$, that satisfies

$$\|\boldsymbol{f} - N_0\|_\infty = \inf_{N\in\varkappa} \|\boldsymbol{f} - N\|_\infty \tag{8}$$

Moreover, the degree of best uniform approximation of $\boldsymbol{f} \in C^d[-1,1]$ from the set $\varkappa$, is

$$E_n(\boldsymbol{f})_\infty = \inf_{N\in\varkappa} \|\boldsymbol{f} - N\|_\infty \tag{9}$$

To measure $E_n(\boldsymbol{f})_\infty$, then modulus of continuity is there, it gives better degree of approximation than ε. To learn more about modulus of continuity, here is the following definition from (7)

Definition 1.5. For any bounded function f on $[a,b]$,the modulus of continuity of f is given by

$$\omega_f([a,b];\delta) = sup\{|f(x) - f(y)| : x, y \in [a,b], |x-y| \le \delta\} \tag{10}$$

2. Auxiliary Lemmas

Now, we need some properties of Bernstein's polynomials to prove our main result. In general, Bernstein polynomials satisfies the following properties

Lemma 2.1. Properties of Bernstein polynomials (7)

1. $\sum_{k=0}^{n} \binom{n}{k} x^k (1-x)^{n-k} = 1$
2. $\sum_{k=0}^{n} \left(\frac{k}{n} - x\right)^2 \binom{n}{k} x^k (1-x)^{n-k} \le \frac{1}{4n}$
3. For $\delta > 0$, and $\left|\frac{k}{n} - x\right| \ge \delta$, implies $\sum_{k=0}^{n} \binom{n}{k} x^k (1-k)^{n-k} \le \frac{1}{4n\sigma}$

Lemma 2.2. Properties of $\boldsymbol{\omega_f}$ (7)

1- If f is uniformly continuous then $\omega_f(\delta) \longmapsto 0 \ \ as\ \delta \longmapsto 0^+$

2- If $\delta` < \delta, than\ \omega_f(\delta`) \le \omega_f(\delta)$

3- $\omega_f(n\delta) \le n\ \omega_f(\delta)$, for $n \in N$

4- $\omega_f(\lambda\delta) \le (1+\lambda)\omega_f(\delta), for\ \lambda > 0.$

3.Existence of Best Approximation

This is the main section of the paper, it concludes when and how we find best approximation with our Bernstein neural network,

3.1. Existence Theorem

Let $\boldsymbol{f} \in C^d[-1,1]$, then for any $n \in N$, with $d \leq n$, there exist a neural network of $\boldsymbol{f}$

$$N_n(\boldsymbol{f}) = \sum_{j=0}^{d} \sum_{k=0}^{n} c_{j,i}\, \sigma_j(\langle \boldsymbol{w}.\boldsymbol{x} \rangle + b_j)\,, \tag{11}$$

that satisfies

$$E_n(\boldsymbol{f})_\infty \leq c\, \omega_k(\boldsymbol{f}, \delta) \tag{12}$$

4. Proof of Existence Theorem

Let $n \in \boldsymbol{N}$ *and* $d \leq n$, then by (8), (6), (10), Lemma 2.1, and Minkowski Inequality, we have

$$|f - N_{n,d}(f)| = \left| \sum_k \sum_j f(y_j) - f\left(\frac{k}{n}\right) \frac{n!}{k!\,(n-k)!} {y_j}^k (1-y_j)^{n-k} \right|$$

$$\leq \sum_k \sum_j \left| f(y_j) - f\left(\frac{k}{n}\right) \frac{n!}{k!\,(n-k)!} {y_j}^k (1-y_j)^{n-k} \right|$$

$$\leq \sum_k \sum_j \left(\left| \left(f(y_j) - f\left(\frac{k}{n}\right) \right) \frac{n!}{k!\,(n-k)!} {y_j}^k (1-y_j)^{n-k} \right| \right)$$

$$\leq \sum_k \sum_j \omega_f \left(\left| y_j - \frac{k}{n} \right| \right) \frac{n!}{k!\,(n-k)!} {y_j}^k (1-y_j)^{n-k}$$

By using $\lambda = \sqrt{n} \left| y - \frac{k}{n} \right|$ and $\delta = \frac{1}{\sqrt{n}}$ in property (3) of Lemma 2.1, then by using Cauchy-shwarz inequality, we get

$$|f - N_{n,d}(f)| \leq \omega_f\left(\frac{1}{\sqrt{n}}\right) \sum_k \sum_j \left[1 + \sqrt{n} \left| y_j - \frac{k}{n} \right| \right] \frac{n!}{k!\,(n-k)!} {y^k}_j (1-y_j)^{n-k}$$

$$= \omega_f\left(\frac{1}{\sqrt{n}}\right) \left[1 + \sqrt{n} \sum_k \sum_j \left| y_j - \frac{k}{n} \right| \frac{n!}{k!\,(n-k)!} {y^k}_j (1-y_j)^{n-k} \right]$$

$$\leq \omega_f\left(\frac{1}{\sqrt{n}}\right) \left[+ \sqrt{n} \sum_j \left\{ \sum_k \left| y_j - \frac{k}{n} \right| \frac{n!}{k!\,(n-k)!} y_j^k\, (1-y_j)^{n-k} \right\}^{\frac{1}{2}} \left\{ \sum_k \left| y_j - \frac{k}{n} \right| \frac{n!}{k!\,(n-k)!} y_j^k\, (1 - y_j)^{n-k} \right\}^{1/2} \right]$$

$$\leq \omega_f\left(\frac{1}{\sqrt{n}}\right)\left[1+\sqrt{n}\sum_{j}^{d}\frac{1}{4n}\right]^{\frac{1}{2}}$$

$$\leq c\ \omega_f\left(\frac{1}{\sqrt{n}}\right)$$

By taking maximum over j to both sides, we finish the proof■

5. Function Approximation by Learning Neural Networks

Many algorithms use weighting technique to adjust the neural network to get the desired target. In this section, we use GRNN (General Regression Neural Networks) algorithm to find a neural approximation for some different continuous functions to train it closer to the target function. Mean Square Error (MSE) is the measure of how well the function approximation is. The optimization problem is to minimize MSE as to reach the target error.

GRNN is an FNN (Feedforward Neural Network) with a radial basis layer and the next linear layer. It generates a good tool for function approximation. It is simply an input-output with a structure of Radial Basis Network in the first layer, but with a different input to the transfer function. The Euclidean distance is applied to the input $\boldsymbol{x}$ and the weight $\boldsymbol{w}$ as follow

$$\|\boldsymbol{x}-\boldsymbol{w}\| = \left(\sum_{j=1}^{d}(x_j - w_j)^2\right)^{1/2},$$

where $\boldsymbol{x} = (x_1, x_2, \cdots, x_d)$ and $\boldsymbol{w} = (w_1, w_2, \cdots, w_d)$.

And then, the result is activated by Gaussian RBF as follow

$$\sigma(x_i) = e^{-{x_i}^2/\rho^2}$$

Also, set

$$Y(\boldsymbol{x}) = \frac{\sum_{i=1}^{d} Y(x_i)e^{-(x_i-w_i)^2/\rho^2}}{\sum_{i=1}^{d} e^{-(x_i-w_i)^2/\rho^2}},$$

where ρ^2 is the smoothing parameter of the Gaussian kernel (Default 1). $Y(\boldsymbol{x})$ represents the normal distribution.

Theoretically, RBF maps an input X to get a weight W, so we could measure how much the function is the approach to the target by calculating $\|\boldsymbol{X} - \boldsymbol{Y}\|$. Thus, the result is trained again in the same procedure.

6. Experimental Results

One of the most important neural network algorithms for function approximation is the GRNN algorithm. We introduce some practical results for our algorithm, including function examples, by using Matlab. In addition, all calculations and figures are done by Matlab too.

6.1. Continuous Exponential Function

We apply GRNN to approximate a continuous exponential function which is of one variable $x \in [0,1]$

$$y = (x + 1)\ exp\ (-3x + 3)$$

The following figure shows both the target function and the approximated one using GRNN, and the MSE is 0.0042

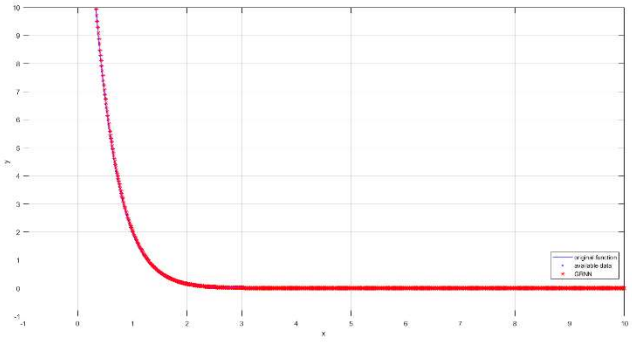

Figure -1 Approximation of $y = (x + 1)\ exp\ (-3x + 3)$ by GRNN

6.2. Continuous Periodic Function

We apply GRNN to approximate the function of a continuous periodic function, which is of one variable $x \in [0,1]$

$$y = sin(4x)exp(-|5x|)$$

The following figure shows both the target function and the approximated one using GRNN; the MSE is 0.0031

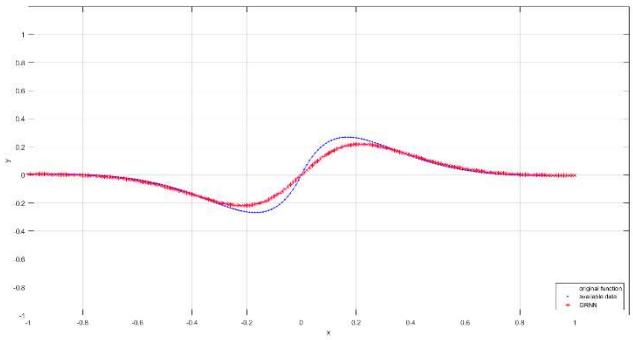

Figure-2 Approximation of $y = sin(4x)exp(-|5x|)$ by GRNN

6.3. Santner Function

Another famous function that is used for approximation is the Lim Function (18). It is given by

$$y = f(x) = e^{-1.4x}\cos(3.5\pi x)\,, x \in [0,1]$$

The following figure shows both the target function and the approximated one using GRNN; the MSE is 1.4637e-05

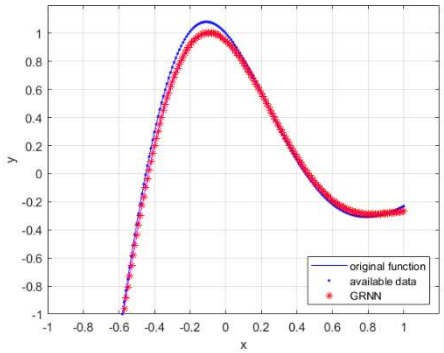

Figure-3 Approximation of $f(x) = e^{-1.4x}\cos(3.5\pi x)$ by GRNN

6.4. MSE Results

Finally, we collect our results of MSE between output and target of each function in the examples above with the following table,

Function	MSE
$\boldsymbol{y = (x+1)exp(-3x+3)}$	0.0042
$\boldsymbol{y = sin(4x)exp(-\lvert 5x\rvert)}$	0.0031
$\boldsymbol{y = e^{-1.4x}cos(3.5\pi x)}$	1.4637e-05

7. Conclusions

This paper includes a special type of neural network that we defined in terms of Bernstein polynomials, we proved that the uniform approximation exists. Moreover, the degree of best approximation is concluded. Numerically, we find best approximation for some continuous functions by using GRNN algorithm and calculate the MSE of each. More improvements and generalizations are possible in this topic.

8. References

1. Almurieb HA, Bhaya ES. Best neural simultaneous approximation. Indonesian Journal of Electrical Engineering and Computer Science. 2020 Dec 1;20(3):1584–90.
2. Dingankar A, Phatak D, County B. Simultaneous Approximation with Neural Networks. 2000;(June 2014).
3. Bhaya ES, Sharba ZA. L_p_ approximation by ReLU neural networks. Karbala International Journal of Modern Science. 2020;6(4):414–9.
4. Lin S, Guo X, Cao F, Xu Z. Approximation by Neural Networks with Scattered Dataq. Applied Mathematics and Computation [Internet]. 2015;224(June):29–35. Available from: http://dx.doi.org/10.1016/j.amc.2013.08.014
5. Liu B. Optimal Function Approximation with Relu Neural Networks. 2019;
6. Almurieb HA, Bhaya ES. Monotone approximation by quadratic neural network of functions in Lp spaces for p<1. Iraqi Journal of Science. 2020;61(4):870–4.

7. Carothers NL. A Short Course on Approximation Theory. Citeseer; 1998.

8. Chebyshev PL. Théorie des mécanismes connus sous le nom de parallélogrammes". Mémoires des Savants étrangers présentés à l'Académie de Saint-Pétersbourg (in French). 1854;7:539–86.

9. Cao, F.L., Xiong JY: Steckin-Marchaud-Type Inequality in Connection with Lp Approximation for Multivariate Bernstein-Durrmeyer Operators. Chinese Contemporary Mathematics. 2001;22(2):137–42.

10. Bernstein S. Sur L'ordre de la Meilleure Approximation des Fonctions Continues par des Polynômes de Degré Donné. Hayez, imprimeur des acad{\'e}mies royales. 1912;4.

11. Steffens K georg. The History of Approximation Theory From Euler to Bernstein.

12. Cybenko G. Continuous Valued Neural Networks: Approximation Theoretic Results. In: Computer Science and Statistics: proceedings of the 20th Symposium on the Interface,. 1988. p. 174–83.

13. Hornik K., Stinchcombe M. WH. Universal Approximation of an Unknown Mapping and its Derivatives Using Multilayer Feedforword Networks. Neural Networks. 1990;3(5):551–60.

14. Chen T, Chen H. Universal Approximation to Nonlinear Operators by Neural Networks with Arbitrary Activation Functions and Its Application to Dynamical Systems. IEEE Transactions on Neural Networks. 1995;6(4):911–7.

15. Suzuki S. Constructive Function Approximation by Three-Layer Artificial Neural Networks. Neural Networks. 1998;11(6):1049–58.

16. Wang J, Xu Z. New Study on Neural Networks: The Essential Order of Approximation. Neural Networks [Internet]. 2010;23(5):618–24. Available from: http://dx.doi.org/10.1016/j.neunet.2010.01.004

17. Almurieb HA, Bhaya ES. SoftMax Neural Best Approximation. In: Series, I O P Conference Science, Materials. 2020.

18. Lim YB, Sacks J, Studden WJ, Welch WJ. Design and Analysis of Computer Experiments when the Output is Highly Correlated over the Input Space. Can J Stat. 2002;30(1):109–126.