Constructing Multilayer Perceptrons as Piecewise Low-Order Polynomial Approximators: A Signal Processing Approach

Ruiyuan Lin, Suya You, Raghuveer Rao and C.-C. Jay Kuo

Abstract—**The construction of a multilayer perceptron (MLP) as a piecewise low-order polynomial approximator using a signal processing approach is presented in this work. The constructed MLP contains one input, one intermediate and one output layers. Its construction includes the specification of neuron numbers and all filter weights. Through the construction, a one-to-one correspondence between the approximation of an MLP and that of a piecewise low-order polynomial is established. Comparison between piecewise polynomial and MLP approximations is made. Since the approximation capability of piecewise low-order polynomials is well understood, our findings shed light on the universal approximation capability of an MLP.**

Index Terms—**multilayer perceptron, feedforward neural network, approximation theory, piecewise polynomial approximation.**

I. Introduction

Piecewise low-order polynomial approximation (or regression) is a classic topic in numerical analysis. It fits multiple low-order polynomials to a function in partitioned intervals. The approximation capability of piecewise low-order polynomials is well understood. In contrast, the representation power of neural networks is not as obvious. The representation power of neural networks has been studied for three decades, and universal approximation theorems in various forms have been proved [1], [2], [3], [4], [5]. These proofs indicate that neural networks with appropriate activations can represent a wide range of functions of interest. Their goals were to make their theorems as general as possible. The proofs were based on tools from functional analysis (e.g., the Hahn-Banach theorem [6]) and real analysis (e.g., Sprecher-Kolmogorov theorem [7], [8]), which is less well known in the signal processing community. Furthermore, although they show the existence of a solution, most of them do not offer a specific MLP design. Lack of MLP design examples hinders our understanding of the behavior of neural networks.

A feedforward construction of an MLP with two intermediate layers was presented to solve a classification problem in [9]. As a sequel to [9], this work addresses the regression problem. For ease of presentation, our treatment will focus on the approximation of a univariate function. We adopt a signal processing (or numerical analysis) approach to tackle

Ruiyuan Lin and C.-C. Jay Kuo are with Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, 3740 McClintock Avenue, Los Angeles, USA, emails:{ruiyuanl,jckuo}@usc.edu
Suya You and Raghuveer Rao are with Army Research Laboratory, Adelphi, Maryland, USA, emails:{suya.you.civ,raghuveer.m.rao.civ}@mail.mil

it. Through the design, we can explain how MLPs offer universal approximations intuitively. The construction includes the choice of activation functions, neuron numbers, and filter weights between layers. A one-to-one correspondence between an MLP and a piecewise low-order polynomial will be established. Except for the trivial case of the unit step activation function, results on MLP construction and its equivalence to piecewise low-order polynomial approximators are new. Another interesting observation is the critical role played by activation functions. That is, activation functions actually define kernels of piecewise low-order polynomials. Since the approximation error of piecewise low-order polynomials is well understood, our findings shed light on the universal approximation capability of an MLP as well.

The rest of this paper is organized as follows. The construction of MLPs corresponding to pieceswise constant, linear and cubic polynomial approximations is detailed in Sec. II. The generalization from the 1-D case to multivariate vector functions is sketched in Sec. III. Concluding remarks and future research directions are given in Sec. IV.

II. Proposed MLP Constructions

Without loss of generality, we consider a continuous function $f(x)$ defined on interval $x \in [0, 1]$, which is uniformly divided into N sub-intervals. Let $h = N^{-1}$ be the length of the subinterval and $x_i = ih$, where $i = 0, 1, \cdots, N$. We show how to construct an MLP so that it serves as a piecewise constant or a piecewise linear approximation to $f(x)$ in this section.

A. Piecewise Constant Approximation

A piecewise constant approximation of $f(x)$ can be written as

$$f(x) \approx f_c(x) = \sum_{i=0}^{N-1} f(x_i)b_i(x), \tag{1}$$

where

$$b_i(x) = \begin{cases} 1, & x_i \leq x < x_{i+1} \\ 0, & \text{otherwise}. \end{cases} \tag{2}$$

is the unit box function. We can rewrite the unit box function as the difference of two unit step functions

$$b_i(x) = u(x - x_i) - u(x - x_{i+1}), \tag{3}$$

where

$$u(x) = \begin{cases} 1, & x \geq 0 \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

is the unit step function. Then, we can rewrite Eq. (1) as

$$f_c(x) = f(x_0)u(x-x_0) + \sum_{i=1}^{N-1}[f(x_i)-f(x_{i-1})]u(h^{-1}(x-x_i)).$$

$$(5)$$

Our designed MLP consists of one input node denoting $x \in [0,1]$, one output node denoting $f(x)$, and N intermediate nodes (or neurons), denoted by R_j, where $j = 0, 1, \cdots, N-1$. We use α_j and ξ_j to denote the weight and bias between the input node, R_{in} and R_j and $\tilde{\alpha}_j$ and $\tilde{\xi}_j$ to denote the weight and bias between R_j and the output node R_{out}, respectively. The response at R_j before activation, denoted by y_j, and the response at R_{out}, denoted by z, can be written as

$$y_j = \alpha_j x + \xi_j, \tag{6}$$

$$z = \sum_{j=0}^{N-1} \tilde{\alpha}_j \text{Act}(y_j) + \tilde{\xi}_j, \tag{7}$$

where Act is the activation function, α_j and $\tilde{\alpha}_j$ are weights and ξ_j and $\tilde{\xi}_j$ are biases.

With the unit step function as activation and Eq. (5), it is easy to verify that we can choose the following weights and biases for the MLP:

$$\begin{aligned} R_{in} \Rightarrow R_j : \quad & \alpha_j = h^{-1}, & \xi_j = -h^{-1}x_j, \\ R_j \Rightarrow R_{out} : \quad & \tilde{\alpha}_j = f(x_j) - f(x_{j-1}), & \tilde{\xi}_j = 0, \end{aligned}$$

$$(8)$$

where $f(x_{-1}) \equiv 0$. The above derivation was given in [5], and it is repeated here for the sake of completeness.

B. Piecewise Linear Approximation

A commonly used nonlinear activation function is the rectified linear unit (ReLU), which can be written as

$$r(x) = \max(0, x). \tag{9}$$

We will construct an MLP using $r(x)$ as the nonlinear activation function and show its equivalence to a piecewise linear approximation to $f(x)$. The latter can be expressed as

$$f(x) \approx f_l(x) = \sum_{i=0}^{N-1} f(x_i)t_i(x), \tag{10}$$

where

$$t_i(x) = t(h^{-1}(x - x_i)) \tag{11}$$

and where $t(x)$ is the unit triangle function in form of

$$t(x) = \begin{cases} x + 1, & \text{if } -1 \leq x \leq 0 \\ -x + 1, & \text{if } 0 \leq x \leq 1 \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

With one input node denoting $x \in [0,1]$ and one output node denoting $f(x)$, we construct an MLP that has 4 neurons in two pairs, denoted by $R_{j,k}$, $k = 1, \cdots, 4$, in the intermediate layer to yield the same effect of t_j. We use $\alpha_{j,k}$ and $\xi_{j,k}$ to denote the weight and bias from the input node R_{in} to $R_{j,k}$, respectively. In our design, they are specified as:

$$\begin{aligned} R_{in} \Rightarrow R_{j,1} : \quad & \alpha_{j,1} = h^{-1}, & \xi_{j,1} = -h^{-1}x_{j-1}, \\ R_{in} \Rightarrow R_{j,2} : \quad & \alpha_{j,2} = h^{-1}, & \xi_{j,2} = -h^{-1}x_j, \\ R_{in} \Rightarrow R_{j,3} : \quad & \alpha_{j,3} = -h^{-1}, & \xi_{j,3} = h^{-1}x_{j+1}, \\ R_{in} \Rightarrow R_{j,4} : \quad & \alpha_{j,4} = -h^{-1}, & \xi_{j,4} = h^{-1}x_j. \end{aligned}$$

$$(13)$$

The responses of neurons $R_{j,k}$, $k = 1, 3$ and $k = 2, 4$ after ReLU nonlinear activation are shown in Figs. 1 (a) and (b), respectively, where the dotted lines indicate those of individual neurons and the solid lines show those of combined responses. Finally, the combination of all responses of all 4 neurons at the output node is shown in Fig. 1 (c). It is a unit triangle plus a vertical shift.
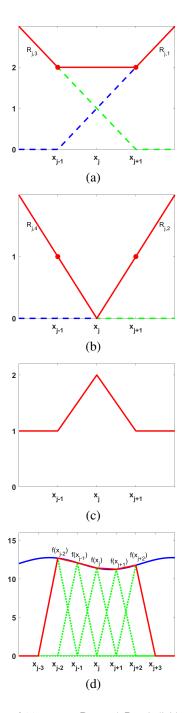

(a)

(b)

(c)

(d)

Fig. 1. Responses of (a) neurons $R_{j,1}$ and $R_{j,2}$ individually (dotted lines) and jointly (solid lines), (b) neurons $R_{j,3}$ and $R_{j,4}$, (c) all four neurons combined, and (d) the approximation of $f(x)$ with a sequence of weighted triangle functions.

Similarly, we use $\tilde{\alpha}_{j,k}$ and $\tilde{\xi}_{j,k}$ to denote the weight and bias from $R_{j,k}$ to the output node R_{out}, respectively. They

are specified below:

$$R_{j,k} \Rightarrow R_{out}: \quad \tilde{\alpha}_{j,k} = f(x_j), \qquad k = 1, 3$$
$$R_{j,k} \Rightarrow R_{out}: \quad \tilde{\alpha}_{j,k} = -f(x_j), \qquad k = 2, 4 \qquad (14)$$
$$R_{j,k} \Rightarrow R_{out}: \quad \tilde{\xi}_{j,k} = -0.25f(x_j), \quad 1 \le k \le 4.$$

It is easy to verify from the Fig. 1 (d) that these four neurons are nothing but a weighted triangle centered at x_i with its base between x_{i-1} and x_{i+1} and height $f(x_i)$ and $f(x)$ can be approximated by a sequence of such weighted triangles.

C. Piecewise Cubic Approximation

Next, we design an MLP that offers a piecewise cubic approximation to $f(x)$. Besides incoming and outgoing weights/biases, we will design the unit cubic activation (see Fig. 2(a)) in form of:

$$q(x) = \begin{cases} 0 & x \le -1, \\ a_3 x^3 + a_2 x^2 + a_1 x + a_0, & -1 \le x \le 1, \quad (15) \\ 1 & x \ge 1, \end{cases}$$

where $q(x)$, $-1 \le x \le 1$, satisfies two constraints: i) $q(-1) = 0$ and, ii) $q(x)$ is anti-symmetric with respect to $(0, 0.5)$; i.e.,

$$q(x) + q(-x) = 1. \qquad (16)$$

Note that these two constraints imply $q(0) = 0.5$ and $q(1) = 1$. By substituting $q(-1) = 0$, $q(0) = 0.5$, $q(1) = 1$ into Eq. (15), we get $a_2 = 0$, $a_1 + a_3 = 0.5$, and $a_0 = 0.5$. There is one degree of freedom left. To complete the design, one choice is to specify the first-order derivative at the inflection point: $(x, f(x)) = (0, 0.5)$. As shown in Fig. 2(a), the inflection point has the maximum first-order derivative and its second-order derivative is zero.

For interval $[x_{j-2}, x_{j+2}]$ centered at x_j, we can use two neurons to build a third-order unit bump function plus a vertical shift as shown in Fig. 2(b). The weight and bias from the input node R_{in} to $R_{j,k}$ are specified as:

$$R_{in} \Rightarrow R_{j,1}: \quad \alpha_{j,1} = h^{-1}, \quad \xi_{j,1} = -h^{-1}x_{j-1},$$
$$R_{in} \Rightarrow R_{j,2}: \quad \alpha_{j,2} = -h^{-1}, \quad \xi_{j,2} = h^{-1}x_{j+1} \qquad (17)$$

where $R_{j,1}$ and $R_{j,2}$ are activated by the unit cubic function, $q(x)$. The weights and biases from $R_{j,k}$, $k = 1, 2$, to output node R_{out} are:

$$R_{j,k} \Rightarrow R_{out}: \quad \tilde{\alpha}_{j,k} = g(x_j), \quad \tilde{\xi}_{j,k} = -0.5g(x_j), \qquad (18)$$

where $g(x_j)$, $j = 0, \cdots, N$, is the solution to the following linear system of $(N+1)$ equations:

$$g(x_j) + 0.5[g(x_{j-1}) + g(x_{j+1})] = f(x_j), \ 0 \le j \le N, \quad (19)$$

with boundary conditions $g(x_{-1}) = g(x_{N+1}) = 0$. Given the target regression function $f(x_j)$ at the right-hand-side of (19), one can solve the system for $g(x_j)$, $j = 0, \cdots, N$, uniquely. If $f(x)$ is smooth in a local region, $g(x) \approx 0.5f(x)$ as shown in Fig. 2(c).

There is an alternative design as shown in Fig. 2(d), where we increase the spacing between two adjacent bumps from h to $2h$. The weight and bias from the input node R_{in} to $R_{j,k}$
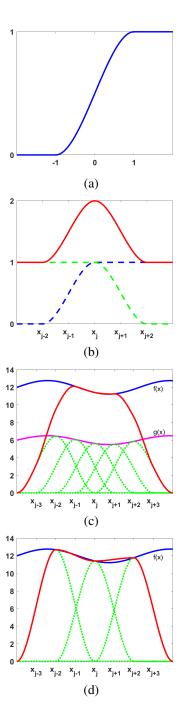

(a)

(b)

(c)

(d)

Fig. 2. (a) The unit cubic activation function, (b) responses of $R_{j,1}$ and $R_{j,2}$ individually (dotted lines) and jointly (solid lines), (c) and (d) two designs for the approximation to $f(x)$ with a sequence of weighted third-order bumps.

still follow Eq. (17) with $j = 2j'$, where $j' = 0, 1, \cdots, N/2$, while Eq. (18) should be changed to

$$R_{j,k} \Rightarrow R_{out}: \quad \tilde{\alpha}_{j,k} = f(x_j), \quad \tilde{\xi}_{j,k} = -0.5f(x_j). \qquad (20)$$

This is fine since there is no interference between $f(x_{j-2})$, $f(x_j)$ $f(x_{j+2})$. Furthermore, using the Taylor series expansion[1] and based on the fact x_{j+1} is the inflection point of two

[1] under the assumption that low-order derivatives of $f(x)$ are bounded

associated cubic activation functions, we can derive

$$0.5[f(x_{j+2}) + f(x_j)] - f(x_{j+1}) \approx O(h^4), \qquad (21)$$

The design in Fig. 2(c) demands $2N$ neurons while that in Fig. 2(d) demands N neurons only in total. The latter design appears to be simpler.

D. Discussion

This section builds a bridge between Piecewise low-order polynomial and MLP approximations. The approximation errors of piecewise low-order polynomials can be derived using Taylor's series expansion. As $N = h^{-1}$ goes to infinity, the errors of piecewise constant, linear and cubic polynomials converge to zero at a rate of $O(h)$, $O(h^2)$ and $O(h^4)$. The same conclusion applies to the corresponding designed MLPs.

There is however difference between them. To give an example, consider the kernel density estimation problem [10]:

$$\hat{f}_h(x) = \sum_{j=1}^{N} \omega_j K(h^{-1}(x - x_j)), \qquad (22)$$

where $K(x)$ is a kernel. The unknown function $f(x)$ at given point x can be expressed as the summation of weighted kernels, where weights ω_j can be solved by a regression method (say, linear regression) for given $K(x)$. The box, triangle, and third-order unit bump functions are kernels to smooth statistical variations in local intervals. The pre-selected kernel could be too local and rigid to be efficient in handling long- and mid-range correlations in $f(x)$. In contrast, parameters in the first stage of an MLP allow a richer choice of kernels. Our MLP design does not exploit this flexibility fully. For example, we may adopt more different $\alpha_{j,k}$ values (see Fig. 3).
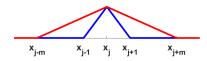

Fig. 3. Two kernels of different supports centered at x_j, which can be achieved by adding more neurons to the intermediate layer of an MLP.

One focal point of neural network research is to understand the role of activation in a neuron. Historically, earliest activation was the simple unit step function [11]. It offers "on" and "off" two states with an abrupt jump. Such activation has zero derivatives except at $x = 0$ and, as a result, backpropagation is not able to update parameters. Several variants such as the sigmoid, ReLU, leaky ReLU activations have been developed later. Paired activations are used to build kernels of low-order polynomials here. As presented earlier, higher-order nonlinear activations are not fundamental in the universal approximation capability of neural networks. They affect the kernel shape only. Instead, the role of nonlinear activation is interpreted as a tool to resolve the sign confusion problem caused by the cascade of multiple layers in [12] and [9].

III. GENERALIZATION TO MULTIVARIATE VECTOR FUNCTIONS

Our presentation has focused on the case of 1D input/output so far. In general, the regression problem can be a q-dimensional vector function of p multivariables:

$$F : \mathbf{x} \to \mathbf{f}(\mathbf{x}), \text{ where } \mathbf{x} \in [0,1]^p \text{ } \mathbf{f} \in R^q, \qquad (23)$$

where $p, q \geq 1$. We will provide a sketch to the generalization from the univariate scalar function to the multivariate vector function in this section. Generalization of our proposed design to a vector function is straightforward since we can design one MLP for each output dimention.

The generalization from the univariate case to the multivariate case using 1D piecewise low-order polynomials, we can proceed as follows:
1) Partition $[0,1]^p$ into p-D cells of volume size h^p;
2) Conduct the 1D approximation separately;
3) Perform the tensor product of 1D kernels to form p-D kernels.

For the MLP design, we cannot perform the tensor product in the third step directly. Instead, we can use the tensor product result to find the weighted contributions of neighboring grid points to the center of each cell and construct an MLP accordingly.

In most practical machine learning applications, the multivariate vector function, $\mathbf{f}(\mathbf{x})$, it is assumed to be sufficiently smooth. That is, its lower-order partial derivatives exist. Consequently, Talyor series expansion for multivariable functions can be adopted to estimate the convergence rate to the target regression function.

IV. CONCLUSION AND FUTURE WORK

A bridge between piecewise low-order polynomials and three-layer MLP approximators was built in this work. Instead of following the traditional path of universal approximation theorem of neural networks based on functional or real analysis, we proposed a signal processing approach to construct MLPs. The construction includes the choice of activation functions, neuron numbers, and filter weights between adjacent layers. Our design decomposes an MLP into two stages: 1) kernel construction in the first stage and 2) weight regression in the second stage. We can explain how an MLP offers an universal approximation to a univariate function intuitively by this approach. Kernel design plays an important role in many machine learning and statistical inference problems. It was shown to be related to the shape of the activation function here.

Given an MLP architecture, backprogagation is used to solve kernel construction and weight regression problems jointly via end-to-end optimization. It is however an open problem to find the optimal number of neurons in the intermediate layer. On the other hand, the traditional approach decouples the whole problem into two subproblems and solve them one by one in a feedforward manner. It is interesting to study the relationship between the kernel design and the first-stage filter weight/bias determination of MLPs. It is also desired to exploit prior knowledge about data and applications for flexible kernel design in feedforward learning systems.

REFERENCES

[1] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[2] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[3] C. K. Chui and X. Li, "Approximation by ridge functions and neural networks with one hidden layer," *Journal of Approximation Theory*, vol. 70, no. 2, pp. 131–141, 1992.

[4] T. Chen and H. Chen, "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems," *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 911–917, 1995.

[5] F. Scarselli and A. C. Tsoi, "Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results," *Neural networks*, vol. 11, no. 1, pp. 15–37, 1998.

[6] W. Rudin, "Functional analysis," 1973.

[7] D. A. Sprecher, "On the structure of continuous functions of several variables," *Transactions of the American Mathematical Society*, vol. 115, pp. 340–355, 1965.

[8] A. N. Kolmogorov, "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition," in *Doklady Akademii Nauk*, vol. 114, no. 5. Russian Academy of Sciences, 1957, pp. 953–956.

[9] R. Lin, Z. Zhou, S. You, R. Rao, and C.-C. J. Kuo, "From two-class linear discriminant analysis to interpretable multilayer perceptron design," *arXiv preprint arXiv:2009.04442*, 2020.

[10] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[11] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[12] C.-C. J. Kuo, "Understanding convolutional neural networks with a mathematical model," *Journal of Visual Communication and Image Representation*, vol. 41, pp. 406–413, 2016.