Geometrical derivation of a single layer neural network with RELU activation

Germán González, gonzalezv.germanh@javeriana.edu.co

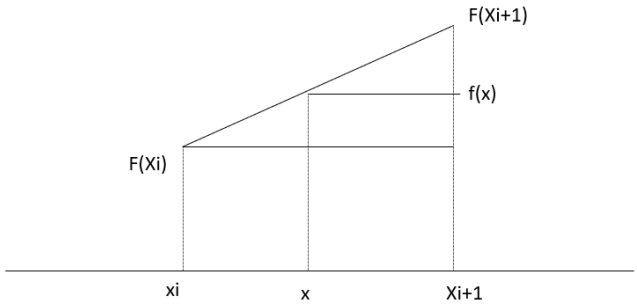

Fig. 1: A point between tow know points in a line without using parametric equations can expresses as in equation (1).

***Abstract*—The geometrical deduction of a single layer neural network is developed as well as the analysis of why the precision of it depends on the number of neurons is presented.**

***Index Terms*—Kolgomorov, Sigmoidal , Relu.**

I. Introduction

RELU activation in deep neural netwoks have proved to work better than sigmoidal and tangh functions. Since the last ones suffer from vanishing gradients. Also, RELU functions as easy to compute. In this article it will be explored a geometrical derivation of a single layer neural network with RELU activation that can be used to aproximate any function. It will also provide a python code that resembles the mathematical formulation so that it can be used quickly in any project.

II. Previous work

In the introductio to neural networks it is always mentioned that a fucntion can be aproximated with a very wide single layer neural network. Examples of that are demonstrated from the works of Kolgomorov [3] and Cybenko[2]. being the lastest one a simpler version of the first one but still requiring to have sound understanding of postgraduate studies in math on teh field of measure theory as the demostration is based on σ-algebra.

III. One-segment linear approximation

A fuction can be aproximated as a line between two points. In the case of the figure it is not done parametrically but as a point **x** between to known points.

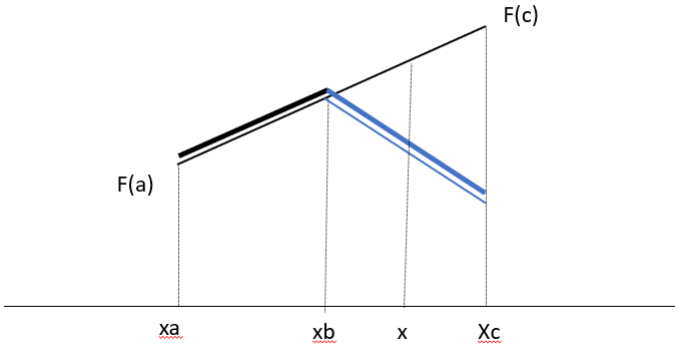

Fig. 2: Two segments can be approximated with two lines. The first part of the figure is the black thick line going from x_a to x_b and the second part is the blue thick line going from x_b to x_c. However, the black lines continues and has to be taken out. To do that the Relu has to make the blue line be zero before x_b and make make the portion of the black line be the the exact opostite of the continuation of the frst segment so that when adding the two segments only the thick segements survive (**??**).

Form triangle equivalences:

$$\frac{F(x_{i+1})-F(x_i)}{x_{i+1}-x_i}=\frac{f(x)-F(x_i)}{x-x_i}=\frac{F(x_{i+1})-f(x)}{x_{i+1}-x} \quad (1)$$

Then, droping the first fraction and arranging the second and third:

$$f(x)\,(x_{i+1}-x)=F(x_{i+1})x-F(x_{i+1})x_i-f(x)(x-x_i)+F(x_i)(x_{i+1}-x) \quad (2)$$

Finally:

$$f(x)(x_{i+1}-x_i)=F(x_{i+1})(x-x_i)+F(x_i)(x_{i+1}-x) \quad (3)$$

Making $\epsilon = x_{i+1}-x_i$. Then we arrive to the expression from:

$$f(x)=\epsilon^{-1}[(x-x_i)F(x_{i+1})+(x_{i+1}-x)F(x_i)] \quad (4)$$

IV. Two-segment linear approximation

For this part and avoid confusion with subindices the X_{i+2}, X_{i+1}, X_i will be replaced with X_c, X_b, X_a as in the figure 2. Then, the equation will be about adding the two segments and substracting the portion in the second half that comes from the first segment as in figure 2.

$$F(x)=\epsilon^{-1}[\frac{(x_c-x)F(x_b)+(x-x_b)F(x_c)}{(x_c-xb)}]-\epsilon^{-1}[\frac{(x_b-x)F(x_a)+(x}{(x_b-x_a} \quad (5)$$

Simplifiying the equation, taking into account that $x_b - x_a$ is equal to $x_b - x_a$:

$$f(x) = (x-x_b)F(X_c)+(x_c-x-x-x_a)F(x_b)+(x-x_b)F(x_a) \tag{6}$$

$$f(x) = (x-x_b)F(X_c)+(x_c-2x-x_a)F(x_b)+(x-x_b)F(x_a) \tag{7}$$

Noticing that $x_c + x_a$ is equal to $2x_b$:

$$f(x) = \epsilon^{-1}[F(X_c) - 2F(x_b) + F(x_a)](x - x_b) \tag{8}$$

The full equation then can be expressed as a sequence or sum of segments as follows:

$$f(x) = \sum_{k=1}^{m} \alpha(k) max(x - b(k), 0) + \beta \tag{9}$$

Where $\alpha(k)$ is:

$$\alpha(k) = \epsilon^{-1}[F(x_c) - 2F(x_b) + F(x_a)] \tag{10}$$

Or, in terms of x_i:

$$\alpha(k) = \epsilon^{-1}[F(x_{i+2}) - 2F(x_{i+1}) + F(x_i)] \tag{11}$$

So, the result is the same as in [1] but with the geometrical deduction. Now, the following is just a sample of how precsise the aproximation can be done. In the following code, the number of neurons is equal to m as in equation (9).

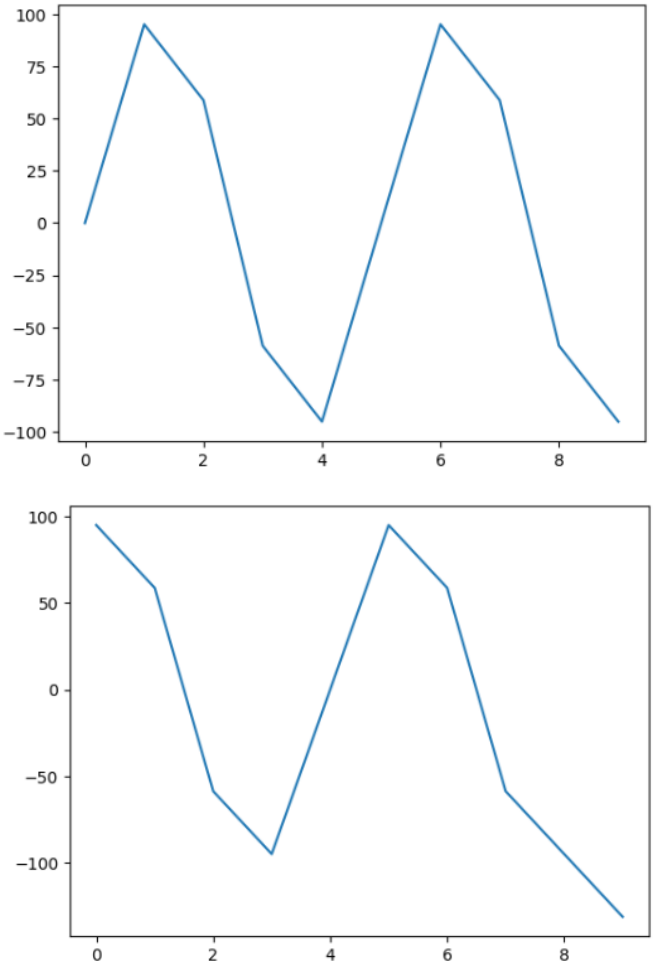

Fig. 3: A ten point sampling of data.

REFERENCES

[1] M. Hirn, Chapter 2 Artificial neural networks. (2020). *cmse890* [. Available at *https* : $//matthewhirn.files.wordpress.com/2020/08/cmse890_spring202$

[2] Cybenko, George V.. "Approximation by superpositions of a sigmoidal function." Mathematics of Control, Signals and Systems 2 (1989): 303-314.

[3] Andrey Kolmogorov, "On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables", Proceedings of the USSR Academy of Sciences, 108 (1956), pp. 179–182; English translation: Amer. Math. Soc. Transl., 17 (1961), pp. 369–373.

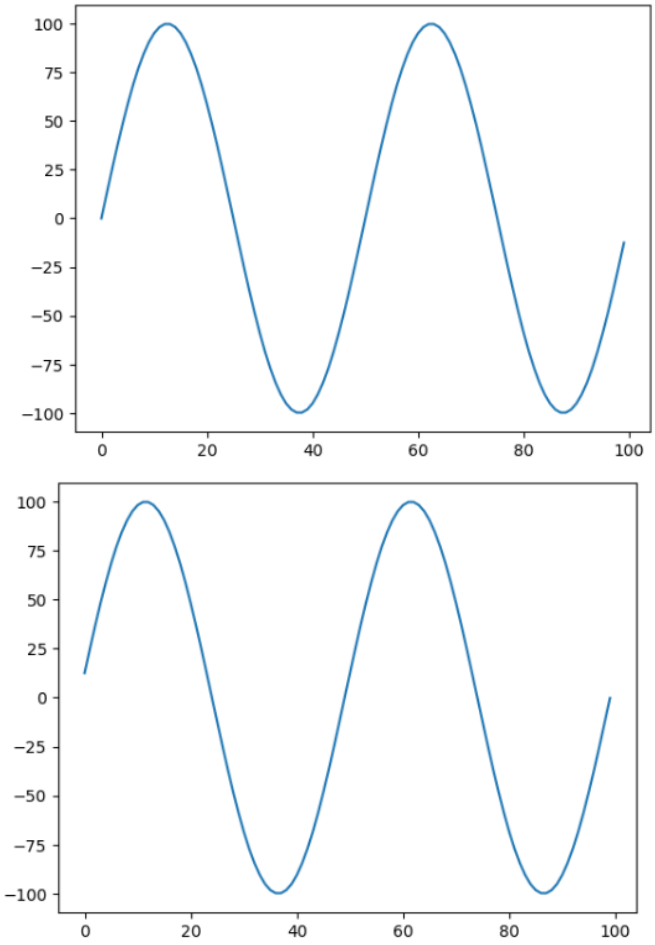

Fig. 4: A 100 points sampling of a sinusoid. The error decreases but the number of neurons is also large.