Why neural networks in the first place: a theoretical explanation

Jonatan Contreras [a], Martine Ceberio [a], Olga Kosheleva [b], and Vladik Kreinovich [a,*]

[a] *Department of Computer Science, University of Texas at El Paso, 500 W. University, El Paso, TX 79968, USA*
E-mail: {jmcontreras2,mceberio,vladik}@utep.edu

[b] *Department of Teacher Education, University of Texas at El Paso, 500 W. University, El Paso, TX 79968, USA*
E-mail: olgak@utep.edu

Abstract. Neural networks – specifically, deep neural networks – are, at present, the most effective machine learning techniques. There are reasonable explanations of why deep neural networks work better than traditional “shallow” ones, but the question remains: why neural networks in the first place? why not networks consisting of non-linear functions from some other family of functions? In this paper, we provide a possible theoretical answer to this question: namely, we show that of all families with the smallest possible number of parameters, families corresponding to neurons are indeed optimal – for all optimality criteria that satisfy some reasonable requirements.

Keywords: neural networks; invariance, function approximation, theoretical explanation

1. Formulation of the problem

A natural question. At present, the most successful machine learning tool is deep neural networks – a specific case of neural networks; see, e.g., [2]. This empirical success leads to a natural question: why are deep neural networks so successful? There are some theoretical explanations of why deep neural networks are more successful than traditional neural networks; see, e.g., [2,3,4]. There are some explanations of why neural networks are usually more effective than some other technique, e.g., than support vector machines [1].

However, a general question remains: why neural networks in general are so effective in the first place? This question is not only about computer applications: artificial neural networks started by simulating biological neurons – that are largely performing similar data processing. Biological neurons are a product of billions of years of improving evolution, so the fact that this type of data processing is used in biological neurons is a good indication that such data processing is effective – but why?

Let us formulate this question in more precise terms. A neural networks is composed of *neurons*, each of which transforms the inputs $x_1, \ldots, x_n$ into an output value

$$y = s(w_1 \cdot x_1 + \ldots + w_n \cdot x_n + w_0) \qquad (1)$$

for some coefficients w_i. In other words:

- first, we form a linear combination

  $$x \overset{\text{def}}{=} w_1 \cdot x_1 + \ldots + w_n \cdot x_n + w_0 \qquad (2)$$

  of the inputs, and
- then, we apply a non-linear function $s(x)$ of one variable – known as the *activation function* – to this linear combination x.

In these terms, the above question is: why is the family (1) of non-linear functions more effective than other possible families of non-linear functions?

[*] Corresponding author. Email: vladik@utp.edu

Let us simplify this question. In order to answer this question, let us perform the following two simplifications.

First, let us notice that in the linear expression (2), the last term w_0 is different from all the other terms. To make this formula more uniform, let us follow the usual arrangement and introduce an auxiliary variable $x_0 = 1$. Then, the formula (1) takes the form

$$s(w_0 \cdot x_0 + \ldots + w_n \cdot x_n). \tag{3}$$

Second, let us take into account that in many cases, the output signal y represents the value of some physical quantity. This happens, e.g., at the last layer of the neural network, when we generate the computations result – and in prediction problem, this result is about the future value of the quantity of interest (e.g., the next day's temperature). The numerical value of a quantity depends on the choice of a measuring unit. If we select a new measuring unit which is C times smaller than the original one, then all numerical values will multiply by C: e.g., if we replace meters by centimeters, all values are multiplied by 100. In the new units, the output of the neuron takes the form

$$C \cdot s(w_0 \cdot x_0 + \ldots + w_n \cdot x_n). \tag{4}$$

From this viewpoint, instead of considering a family of all the functions (3) corresponding to different values w_i, it makes sense to consider a more general family (4) corresponding to all possible values of C and w_i.

What we do in this paper. In this paper, we explain why the family (4) is better than other possible families of nonlinear functions that have the same (or smaller) number of parameters. This provides a possible theoretical explanation of why neural networks – in particular, deep neural networks – are so effective.

2. Analysis of the problem

Natural robustness requirement on transformation functions. Inputs to data processing comes from measurements, and measurements are never absolutely accurate, there is, in general, a non-zero different between the measurement result $\widetilde{x}_i$ and the actual (unknown) value x_i of the measured quantity. This different is known as the *measurement error*. This difference affects the result of data processing. We want to make sure that the corresponding effect is not amplified too much: we want to make sure that the difference in the results is proportional to the measurement errors, i.e., that for the corresponding transformation $y = f(x_1, \ldots, x_n)$ satisfies the following inequality:

$$|f(\widetilde{x}_0, \ldots, \widetilde{x}_n) - f(x_1, \ldots, x_n)| \le$$

$$L \cdot (|\widetilde{x}_0 - x_0| + \ldots + |\widetilde{x}_n - x_n|). \tag{5}$$

for some coefficient L. Such functions are known as *Lipschitz continuous*.

It is known that Lipschitz functions are almost everywhere differentiable, and many of their properties are similar to properties of smooth (everywhere differentiable) functions.

What do we mean by a family of functions. We are interested in functions of $n + 1$ variable $x_0, \ldots, x_n$. The expression (4) describes a family of such functions that depends, in addition to the multiplicative factor C, on $n + 1$ parameters $w_0, \ldots, w_n$, to the total of $n + 2$. Since we are interested in families with the same (or smaller) number of parameters, we need to consider families that also depend on no more than $n + 2$ parameters.

We also want to make sure that a family is uniquely determined by its functions, so if we simply change the parameters without changing the class of functions, we will end up with the same family.

Definition 1. *Let n and p be positive integers.*

- *We say that two Lipschitz continuous mapping $f(x_0, \ldots, x_n, c_0, \ldots, c_p)$ and*

  $$g(x_0, \ldots, x_n, c_0, \ldots, c_p)$$

  are equivalent *if the following two conditions are satisfied*

  $*$ *for each C and for each tuple $c = (c_0, \ldots, c_p)$, there exists a value C' and a tuple $c' = (c'_0, \ldots, c'_p)$ for which, for all x_i, we have:*

  $$C \cdot f(x_0, \ldots, x_n, c_0, \ldots, c_p) =$$

  $$C' \cdot g(x_0, \ldots, x_n, c'_0, \ldots, c'_p);$$

  * *for each C and for each tuple $c = (c_0, \ldots, c_p)$, there exists a value C' and a tuple $c' = (c'_0, \ldots, c'_p)$ for which, for all x_i, we have:*

$$C \cdot g(x_0, \ldots, x_n, c_0, \ldots, c_p) =$$

$$C' \cdot f(x_0, \ldots, x_n, c'_0, \ldots, c'_p).$$

- *By a* family*, we mean an equivalence class of functions – in terms of the above equivalence.*
- *We say that a function $t(x_1, \ldots, x_n)$* belongs *to the family $\mathcal{F}$ – as defined by its element*

$$f(x_0, \ldots, x_n, c_0, \ldots, c_p)$$

  if there exist values $C, c_0, \ldots, c_p$ for which, for all x_i, we have

$$t(x_0, \ldots, x_n) = C \cdot f(x_0, \ldots, x_n, c_0, \ldots, c_p).$$

What do we mean by "better"? We want to analyze why families corresponding to neural data processing perform better than other possible nonlinear families. Usually, "better" means that:

- we have some numerical criterion – e.g., mean square approximation error after a certain fixed computation time, and
- "better" means a smaller value of this numerical criterion.

However, we can have more complex cases: e.g., if we have several families with the same mean square approximation error, we can select, among them, the one with the smallest probability of approximation errors exceeding some given threshold. If this still leaves us with several possible families, we can minimize something else, etc.

So, to describe what is better in the most general way, let us go beyond simple numerical criteria and simply require that we have two relations on the set of all families:

- a relation $\mathcal{F} < \mathcal{G}$ meaning that a family $\mathcal{F}$ is better than the family $\mathcal{G}$; and
- a relation $\mathcal{F} \sim \mathcal{G}$ meaning that a family $\mathcal{F}$ has the same quality as the family $\mathcal{G}$ – with respect to the given criterion.

It is reasonable to require that these two relations satisfy transitivity: if $\mathcal{F}$ is better than $\mathcal{G}$, and $\mathcal{G}$ is better than $\mathcal{H}$, then $\mathcal{F}$ should be better than $\mathcal{H}$. Thus, we arrive at the following definition (see, e.g., [5]):

Definition 2. *By an* optimality criterion*, we mean a pair of relations $(<, \sim)$ on the set of all possible families for which the following properties are satisfied for all $\mathcal{F}$, $\mathcal{G}$, and $\mathcal{H}$:*

- *if $\mathcal{F} < \mathcal{G}$ and $\mathcal{F} < \mathcal{H}$, then $\mathcal{F} < \mathcal{H}$;*
- *if $\mathcal{F} < \mathcal{G}$ and $\mathcal{F} \sim \mathcal{H}$, then $\mathcal{F} < \mathcal{H}$;*
- *if $\mathcal{F} \sim \mathcal{G}$ and $\mathcal{F} < \mathcal{H}$, then $\mathcal{F} < \mathcal{H}$;*
- *if $\mathcal{F} \sim \mathcal{G}$ and $\mathcal{F} \sim \mathcal{H}$, then $\mathcal{F} \sim \mathcal{H}$;*
- *if $\mathcal{F} \sim \mathcal{G}$, then $\mathcal{G} \sim \mathcal{F}$;*
- *if $\mathcal{F} < \mathcal{G}$, then we cannot have $\mathcal{F} \sim \mathcal{G}$.*

In mathematical terms, this pair is known as *pre-order*. The difference from order is that we can have $\mathcal{F} \sim \mathcal{G}$ for $\mathcal{F} \neq \mathcal{G}$.

We have mentioned that if there are several families which are optimal with respect to a given criterion, this means that we can optimize something else – i.e., in effect, that the original criterion was not final. Thus, we arrive at the following definition.

Definition 3.

- *We say that a family $\mathcal{F}_{\rm opt}$ is* optimal *with respect to the optimality criterion $(<, \sim)$ if for every family $\mathcal{F}$, we have $\mathcal{F}_{\rm opt} < \mathcal{F}$ or $\mathcal{F}_{\rm opt} \sim \mathcal{F}$.*
- *We say that the optimality criterion $(<, \sim)$ is* final *if there is exactly one family which is optimal with respect to this criterion.*

Invariance. In many practical situations, it makes sense to consider not only the original values x_i, but also their linear combinations

$$x'_i = \sum_{j=0}^{n} a_{ij} \cdot x_j, \qquad (6)$$

where a_{ij} is a reversible matrix. For example, if x_i are coordinates, we can use a different coordinate system. We can also use different units for different inputs, which also – as we mentioned earlier – amounts of linear transformations $x_i \to C_1 \cdot x_i$.

Such a transformation does not change the problem, so it makes sense to require that the result of comparing two families should not change if we simply apply such a transformation. For example, it would be strange if one program worked better if all the data are in meters, but another one is better if all inputs are in inches. Thus, we arrive at the following definition.

Definition 4.

- *By an* affine transformation A*, we mean a reversible transformation of type (6).*
- *For each family $\mathcal{F}$ described by a function*

$$f(x_0, \ldots, x_n, c_0, \ldots, c_p)$$

and for each affine transformation A, by the result $A(\mathcal{F})$ *of applying this transformation to the family, we mean a family generated by the function*

$$Tf(x_0, \ldots, x_n, c_0, \ldots, c_p) \stackrel{\text{def}}{=}$$

$$f\left(\sum_{j=0}^{n} a_{0j} \cdot x_j, \ldots, \sum_{j=0}^{n} a_{nj} \cdot x_j, c_0, \ldots, c_p\right).$$

- *We say that the optimality criterion $(<, \sim)$ is* affine-invariant *if for every affine transformation A and for every two families $\mathcal{F}$ and $\mathcal{G}$, the following two conditions hold:*
  - ∗ *if $\mathcal{F} < \mathcal{G}$, then $A(\mathcal{F}) < A(\mathcal{G})$;*
  - ∗ *if $\mathcal{F} \sim \mathcal{G}$, then $A(\mathcal{F}) \sim A(\mathcal{G})$.*

Now, we are ready to formulate our main result.

3. Main result

Proposition.

- *The smallest p for which there exists an affine-invariant final optimality criterion on the set of all families is $p = n$.*
- *For $p = n$, for every affine-invariant final optimality criterion on the set of all families, the optimal family is of type (4) for some functions $s(x)$.*

In other words, neurons are indeed optimal non-linear transformation functions – optimal with respect to *any* optimality criterions that satisfies reasonable properties of being final and affine-invariant.

Proof. Let $(<, \sim)$ be a final affine-invariant optimality criterion, and let $\mathcal{F}_{\text{opt}}$ denote the family which is optimal with respect to this criterion. Let us prove that this function has the neural form (4).

1°. Let us first prove that the family $\mathcal{F}_{\text{opt}}$ is itself affine-invariant, i.e., that for each affine transformation A, we have $A(\mathcal{F}_{\text{opt}}) = \mathcal{F}_{\text{opt}}$.

Indeed, the fact that the family $\mathcal{F}_{\text{opt}}$ is optimal means that for every family $\mathcal{F}$, we have either $\mathcal{F}_{\text{opt}} < \mathcal{F}$ or $\mathcal{F}_{\text{opt}} \sim \mathcal{F}$. In particular, for every family $\mathcal{F}$, one of these two conditions is satisfied for the family $A^{-1}(\mathcal{F})$, where A^{-1} denote the inverse affine transformation. In other words, we have either $\mathcal{F}_{\text{opt}} < A^{-1}(\mathcal{F})$ or $\mathcal{F}_{\text{opt}} \sim A^{-1}(\mathcal{F})$.

Due to affine-invariance, taking into account that $A(A^{-1}(\mathcal{F})) = \mathcal{F}$, we conclude that $A(\mathcal{F}_{\text{opt}}) < \mathcal{F}$ or $A(\mathcal{F}_{\text{opt}}) \sim \mathcal{F}$. This is true for each family $\mathcal{F}$. By definition of optimality, this means that the family $A(\mathcal{F}_{\text{opt}})$ is optimal. But we know that $\mathcal{F}_{\text{opt}}$ is optimal, and we assumed that our optimality criterion is final – which means that there is only one optimal family. Thus, we indeed have $A(\mathcal{F}_{\text{opt}}) = \mathcal{F}_{\text{opt}}$.

2°. The property 1° means that for each function $t(x_1, \ldots, x_n)$ from the optimal family and for each affine transformation, the transformed function also belongs to the same optimal family.

The functions f forming the family $\mathcal{F}$ are Lipschitz and thus, almost everywhere differentiable. Let us pick one such function $t(x_0, \ldots, x_n)$ and a point $(X_0, \ldots, X_n)$ where its value is non-zero and its gradient is defined and is non-zero.

We can always perform an affine transformation of coordinates so that in the new coordinates the gradient vector will be parallel to the 0-th axis – e.g., we can rotate the axes so that one of the axes becomes parallel to the gradient vector. In the new coordinates $z_0, \ldots, z_n$, for the correspondingly transformed function $T(z_0, \ldots, z_n)$, we have

$$\nabla T = \left(\frac{\partial T}{\partial z_0}, \frac{\partial T}{\partial z_1}, \ldots, \frac{\partial T}{\partial z_n}\right) = (1, 0, \ldots, 0)$$

at the selected point – which in the new coordinates, has the form $(Z_0, \ldots, Z_n)$.

By multiplying this function $T \in \mathcal{F}_{\text{opt}}$ by an appropriate constant C, we can get, for each possible value $T_0 \neq 0$, a new function from the family $\mathcal{F}$ for which $C \cdot T(Z_0, \ldots, Z_n) = T_0$ and for which the gradient is still parallel to the 0-th axis. Similarly, for any given non-zero vector $v = (v_0, \ldots, v_n)$, by rotating the coordinates z_i (and, if needed, by re-scaling all of them), we can get a new function from the family $\mathcal{F}_{\rm opt}$ for which the gradient at the point $(Z_0, \ldots, Z_n)$ is equal to v. Thus, for each tuple $(T_0, v_0, \ldots, v_n)$, we have a function from the family $\mathcal{F}_{\rm opt}$ for which:

- the value at the point $(Z_0, \ldots, Z_n)$ is equal to T_0 and
- the gradient at this point is equal to $(v_0, \ldots, v_n)$.

Thus, if we assign, to each tuple $(T_0, v_0, \ldots, v_n)$, one of the corresponding functions from the family $\mathcal{F}_{\rm opt}$, we will get a $(n+2)$-parametric family of functions. Thus, the total number $p+2$ of parameters (one parameter C and $p+1$ parameters $c_0,]dots, c_p)$ cannot be smaller than $n+2$, thus $p \ge n$. This proves the first statement of our proposition. To be more precise, we also need to prove that such a criterion exists, but this is easy – e.g., a criterion according to which the neural family (4) is better than every other family – while all other families are equivalent to each other – is clearly final and affine-invariant.

Let us prove the second statement. For this, let us consider the case when $p = n$. In this case, the whole family $\mathcal{F}_{\rm opt}$ depends only on $n+2$ parameters. Thus, if we had, for each tuple, a whole at-least-1-parametric family of functions corresponding to this tuple, we would have a family determined by more than $n+2$ parameters – which would contradict to our assumption that $p = n$.

In particular, this means that the functions

$$T(z_0, \alpha \cdot z_1, z_2, \ldots, z_n)$$

which also belong to the family $\mathcal{F}_{\rm opt}$ and for which the tuple $(T_0, v_0, \ldots, v_n)$ is the same cannot form a 1-parametric family: which means that they should all be identical, i.e., that

$$T(z_0, \alpha \cdot z_1, z_2, \ldots, z_n) = T(z_0, \alpha' \cdot z_1, z_2, \ldots, z_n)$$

Acknowledgments

This work was supported in part by the National Science Foundation grants 1623190 (A Model of Change for all α and α'. This means that the function T cannot depend on z_1 at all. Similarly, we can prove that the function does not depend on any other variable, i.e., that it depends only on z_0: $T(z_0, \ldots, z_n) = s(z_0)$ for some function $s(x)$ of one variable.

By applying linear transformations to z_i and multiplying the expression y C, we get exactly the family (4). The proposition is proven.
for Preparing a New Generation for Professional Practice in Computer Science), and HRD-1834620 and HRD-2034030 (CAHSI Includes), and by the AT&T Fellowship in Information Technology.

It was also supported by the program of the development of the Scientific-Educational Mathematical Center of Volga Federal District No. 075-02-2020-1478.

References

[1] L. Bokati, O. Kosheleva, V. Kreinovich, and A. Sosa, *Why Deep Learning Is More Efficient than Support Vector Machines, and How It Is Related to Sparsity Techniques in Signal Processing*, Proceedings of the 2020 4th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence ISMSI'2020, Thimpu, Bhutan, April 18–19, 2020.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, Massachusetts, 2016.

[3] V. Kreinovich and O. Kosheleva, *Deep Learning (Partly) Demystified*, Proceedings of the 2020 4th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence ISMSI'2020, Thimpu, Bhutan, April 18–19, 2020.

[4] V. Kreinovich and O. Kosheleva, *Optimization under Uncertainty Explains Empirical Success of Deep Learning Heuristics*, In: P. Pardalos, V. Rasskazova, and M. N. Vrahatis (eds.), Black Box Optimization, Machine Learning and No-Free Lunch Theorems, Springer, Cham, Switzerland, 2021, 195–220.

[5] H. T. Nguyen and V. Kreinovich, *Applications of Continuous Mathematics to Computer Science*, Kluwer, Dordrecht, 1997.