Why neural networks in the first place: a theoretical explanation

Jonathan Contreras, Martine Ceberio
Olga Kosheleva, Vladik Kreinovich
University of Texas at El Paso
500 W. University
El Paso, Texas 79968, USA
jmcontreras2@utep.edu, mceberio@utep.edu,
olgak@utep.edu, vladik@utep.edu

1. A natural question

- At present, the most successful machine learning tool is deep neural networks – a specific case of neural networks.
- 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.
- There are some explanations of why neural networks are usually more effective than some other technique.
- E.g., there are explanations of why neural networks are more effective than support vector machines.
- However, a general question remains: why neural networks in general are so effective in the first place?

2. A natural question (cont-d)

- 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?

3. Let us formulate this question in more precise terms

- A neural network is composed of *neurons*.
- Each neuron transforms the inputs $x_1, \ldots, x_n$ into an output value $y = s(w_1 \cdot x_1 + \ldots + w_n \cdot x_n + w_0)$ for some coefficients w_i.
- In other words:
  - first, we form a linear combination $X \stackrel{\text{def}}{=} w_1 \cdot x_1 + \ldots + w_n \cdot x_n + w_0$ 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 above family of non-linear functions more effective than other possible families of non-linear functions?

4. 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, 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 above formula takes the form $s(w_0 \cdot x_0 + \ldots + w_n \cdot x_n)$.
- 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.
- In a prediction problem, this result is about the future value of the quantity of interest (e.g., the distance between a mobile robot and a wall).

5. Let us simplify this question (cont-d)

- 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

$$C \cdot s(w_0 \cdot x_0 + \ldots + w_n \cdot x_n).$$

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

6. What we do in this talk

- In this talk, we explain why:
  - the above family 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.

7. 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 difference between the measurement result $\widetilde{x}_i$ and the actual (unknown) value x_i of the measured quantity.
- This difference 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.

8. Natural robustness requirement (cont-d)

- So, the corresponding transformation $y = f(x_1, \ldots, x_n)$ should satisfy the following inequality for some L:

$$|f(\widetilde{x}_0, \ldots, \widetilde{x}_n) - f(x_1, \ldots, x_n)| \le L \cdot (|\widetilde{x}_0 - x_0| + \ldots + |\widetilde{x}_n - x_n|).$$

- Such functions are known as *Lipschitz continuous*.
- It is known that Lipschitz functions are almost everywhere differentiable.
- Many of their properties are similar to properties of smooth (everywhere differentiable) functions.

9. What do we mean by a family of functions

- We are interested in functions of $n+1$ variables $x_0, \ldots, x_n$.
- The above expression 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 $n+2$.
- We are interested in families with the same (or smaller) number of parameters.
- So, 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.

10. Resulting definition

- Let n and p be positive integers.
- We will denote $x = (x_0, \ldots, x_n)$ and $c = (c_0, \ldots, c_p)$.
- We say that two nonlinear Lipschitz continuous mapping $f(x, c)$ and $g(x, c')$ are *equivalent* if the following two conditions are satisfied
  - for each C and for each tuple c, there exists a value C' and a tuple c' for which, for all x, we have: $C \cdot f(x, c) = C' \cdot g(x, c')$;
  - for each C' and for each tuple c', there exists a value C and a tuple c for which, for all x, we have: $C' \cdot g(x, c') = C \cdot f(x, c)$.
- By a *family*, we mean an equivalence class of functions – in terms of the above equivalence.
- A family $\mathcal{F}$ is determined by the mapping $f(x, c)$.
- We say that a function $t(x)$ *belongs* to $\mathcal{F}$ if there exist values C, c for which, for all x, we have $t(x) = C \cdot f(x, c)$.

11. 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.

12. What do we mean by "better" (cont-d)

- So, to describe what is better in the most general way, let us go beyond simple numerical criteria.
- Let us 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.

13. Resulting definition

- By an *optimality criterion*, we mean a pair of relations $(<, \sim)$ for which the following properties hold for all $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$:
  - if $\mathcal{F} < \mathcal{G}$ and $\mathcal{G} < \mathcal{H}$, then $\mathcal{F} < \mathcal{H}$;
  - if $\mathcal{F} < \mathcal{G}$ and $\mathcal{G} \sim \mathcal{H}$, then $\mathcal{F} < \mathcal{H}$;
  - if $\mathcal{F} \sim \mathcal{G}$ and $\mathcal{G} < \mathcal{H}$, then $\mathcal{F} < \mathcal{H}$;
  - if $\mathcal{F} \sim \mathcal{G}$ and $\mathcal{G} \sim \mathcal{H}$, 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 $\mathcal{F} \neq \mathcal{G}$.

14. What is meant by "better" (cont-d)

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

15. Invariance

- In many practical situations, it makes sense to consider not only the original values x_i, but also their linear combinations $x'_i = \sum_{j=0}^{n} a_{ij} \cdot x_j$.
- Here 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.
- This also – as we mentioned earlier – amounts to linear transformations $x_i \to C_i \cdot x_i$.
- 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.

16. Invariance (cont-d)

- 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.

17. Resulting definition

- By an *affine transformation* A, we mean a reversible linear transformation.
- Let a family $\mathcal{F}$ be described by a function $f(x_0, \ldots, x_n, c_0, \ldots, c_p)$ and let A be an affine transformation.
- By the *result* $A(\mathcal{F})$, 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 $(<, \sim)$ is *affine-invariant* if for every affine transformation A and for every two families $\mathcal{F}$ and $\mathcal{G}$, we have:
  - if $\mathcal{F} < \mathcal{G}$, then $A(\mathcal{F}) < A(\mathcal{G})$; and
  - if $\mathcal{F} \sim \mathcal{G}$, then $A(\mathcal{F}) \sim A(\mathcal{G})$.
- Now, we are ready to formulate our main result.

18. Main result

- 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 neural form for some $s(x)$.
- In other words, neurons are indeed optimal non-linear transformation functions.
- They are optimal with respect to *any* optimality criterions that satisfies reasonable properties of being final and affine-invariant.

19. Proof: Part 1

- Let $(<, \sim)$ be a final affine-invariant optimality criterion.
- 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.
- 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} is the inverse.
- 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 $A(\mathcal{F}_{\text{opt}}) \sim \mathcal{F}$.

20. Proof: Part 1 (cont-d)

- 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.
- We assumed that our optimality criterion is final – which means that there is only one optimal family.
- Thus, we indeed have $A(\mathcal{F}_{\text{opt}}) = \mathcal{F}_{\text{opt}}$.

21. Proof: Part 2

- In Part 1, we proved 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 $t(x_0, \ldots, x_n)$; this function is nonlinear and almost everywhere differentiable.
- So, there exist points $(X_0, \ldots, X_n)$ where its value is non-zero and its gradient is defined and is non-zero. Let us select one such point.
- We can 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.

22. Proof: Part 2 (cont-d)

- In the new coordinates $z_0, \ldots, z_n$, for the correspondingly transformed function $T(z_0, \ldots, z_n)$, we have

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

- This is true 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.

23. Proof: Part 2 (cont-d)

- 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 $\mathcal{F}_{\text{opt}}$
  - 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 $\mathcal{F}_{\text{opt}}$ 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 $\mathcal{F}_{\text{opt}}$,
  - 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 $c_0, \ldots, c_p$) cannot be smaller than $n+2$, thus $p \geq n$.

24. Proof: Part 2 (cont-d)

- 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 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 $\mathcal{F}_{\text{opt}}$ 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.

25. Proof: Part 2 (cont-d)

- This would contradict to our assumption that $p = n$.
- In particular, this means that the functions $T(z_0, \alpha \cdot z_1, z_2, \ldots, z_n)$ corresponding to all $\alpha > 0$:
  - which also belong to the family $\mathcal{F}_{\text{opt}}$ and for which the tuple $(T_0, v_0, \ldots, v_n)$ is the same
  - cannot form a 1-parametric family.
- This means that all these functions should all be identical, i.e., that

$$T(z_0, \alpha \cdot z_1, z_2, \ldots, z_n) = T(z_0, \alpha' \cdot z_1, z_2, \ldots, z_n) \text{ for all } \alpha \text{ and } \alpha'.$$

- This means that the function T cannot depend on z_1 at all.

26. Proof: Part 2 (cont-d)

- Similarly, we can prove that the function does not depend on any other variable, i.e., that it depends only on z_0.
- So, $T(z_0, \ldots, z_n) = s(z_0)$ 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 neural family.
- The proposition is proven.

27. Acknowledgments

- This work was supported in part by the National Science Foundation grants:
  - 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science), and
  - HRD-1834620 and HRD-2034030 (CAHSI Includes).
- It was also supported 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.