Logarithmic Number System Is Optimal for AI Computations: Theoretical Explanation of an Empirical Success

Olga Kosheleva, Vladik Kreinovich,
Christoph Lauter, and Kristalys Ruiz-Rohena
University of Texas at El Paso
500 W. University, El Paso, TX 79968, USA
olgak@utep.edu, vladik@utep.edu
cqlauter@utep,edu, skruizrohena@miners.utep.edu

1. Introduction

- Everyone is familiar with recent spectacular successes of deep-learning-based AI systems such as ChatGPT.
- However, these systems are not perfect.
- To make them better, we need to train them more.
- Training already takes a long time – often years.
- So, to make further progress, we need to perform much more computations in the same period of time.
- In other words, we need to further speed up computations.
- The larger the range of possible values, the more bits we need to describe the exponent, and thus, the more bit operations we need.
- So, one way to speed up computations is to decrease this range by using a non-linear transformation.

2. Introduction (cont-d)

- For example:
  - if instead of the signal's power, we use amplitude – i.e., power's square root,
  - then the range from 1 to 100 reduces to 1 to 10.
- Empirical data shows that:
  - such nonlinear transformations do speed up training, and
  - logarithmic transformation works the best.
- Log transformation is the best of all that were tried.
- But is it really the best?
- Maybe some transformation that was not tried will be even better?

3. Introduction (cont-d)

- To answer this question, in this talk:
  - we formulate the question of selecting the best transformation in precise terms, and
  - we prove that under some reasonable assumptions, every optimal transformation should $x \mapsto \log(x)$, $x \mapsto \exp(x)$, or $x \mapsto x^\alpha$.
- Since empirically, log transformation is the best of these three, it is therefore optimal.
- This way:
  - not only we explain why log was better than all transformations that were tried,
  - we also provide an assurance that log is better than all others transformations – including those that no one tried.

4. Introduction (cont-d)

- To prove this result, we use the technique of invariances.
- This is one of the main techniques of modern physics, where invariances are known as *symmetries*.
- This technique has been also successfully used in AI to explain empirically successful selection:
  - of activation functions in neural networks and
  - of “and”- and “or”-operations in fuzzy logic.

5. Let Us Formulate This Problem in Precise Terms

- We want to find the optimal nonlinear transformation $x \mapsto y = f(x)$.
- All values that we deal with are approximate.
- So, it makes sense to require:
  - that small changes in x should lead to equivalently small changes in y, i.e., in precise terms,
  - that the function $f(x)$ be smooth (differentiable).
- In principle, there may be several equally good transformations, maybe the whole family of them.
- We want to make our formulation as general as possible.
- So, we need to consider looking for a *family* of transformations $x \mapsto F(x, c_1, \ldots, c_k)$ characterized by some parameters $c_1, \ldots, c_k$.
- Thus, we arrive at the following definition. Let k be fixed.
- By a *k-family*, we mean a smooth function $F(x, c_1, \ldots, c_k)$ whose dependence on x is nonlinear.

6. Formulating the Problem in Precise Terms (cont-d)

- We say that a function $f(x)$ *belongs* to the family if for some c_i, we have $f(x) = F(x, c_1, \ldots, c_k)$ for all x.
- In general, the fewer parameters the expression has, the faster it is to compute this expression.
- Since we want to speed up computations, we should be looking for families with the smallest number of parameters k.
- Informally, out of all k-families, we want to find the family that is, in some sense, optimal.
- How can we define "optimal"?
- In many applications, we have a well-defined objective function $J(a)$.
- Then, selecting an optimal alternative a simply means maximizing (or minimizing) this objective function.
- However, this formulation does not cover all possible optimization settings.

7. Formulating the Problem in Precise Terms (cont-d)

- For example, we may want to find the sorting algorithm with the smallest average computations time.
- There are several such algorithms.
- So, we can use this non-uniqueness to optimize something else.
- E.g., we may select, among all the fastest-on-average algorithms, the one with the best worst-case computation time.
- If we still have several equally good algorithms:
  - this means that our selection method is not final,
  - we can use the remaining non-uniqueness to optimize something else, etc.

8. Formulating the Problem in Precise Terms (cont-d)

- In all these cases, each optimality criterion allows us to compare pairs of alternatives (a, b) and decide:
  - whether a is better than b (we will denote it by $a \succ b$)
  - or b better than a ($b \succ a$),
  - or that they are equivalent with respect to this criterion (we will denote it $a \sim b$).
- Of course, if a is better than b and b is better than c, then a should be better than c.
- Thus, we arrive at the following definition.
- Let $\mathcal{A}$ be a set. Its elements will be called *alternatives*.

9. Formulating the Problem in Precise Terms (cont-d)

- By an *optimality criterion* on $\mathcal{A}$, we mean a pair $(\succ, \sim)$ of binary relations for which, for all a, b and c:
  - if $a \succ b$ and $b \succ c$, then $a \succ c$; if $a \succ b$ and $b \sim c$, then $a \succ c$;
  - if $a \sim b$ and $b \succ c$, then $a \succ c$; if $a \sim b$ and $b \sim c$, then $a \sim c$;
  - if $a \sim b$ then $b \sim a$; if $a \succ b$, then $a \not\sim b$.
- We say that the alternative a_{opt} is *optimal* if for each a, we have $a_{\text{opt}} \succ a$ or $a_{\text{opt}} \sim a$.
- We say that an optimality criterion is *final* if there is exactly one *optimal* alternative a_{opt}.
- In mathematics, such a pair of a strict order $<$ and an equivalence relation $\sim$ is called a *pre-order*.
- We want to find the optimal family of transformations $y = f(x)$.

10. Invariance

- To find the optimal family, let us take into account that:
  - for each physical quantity – be it power or amplitude or something else,
  - the numerical value is not absolute.
- If we change a measuring unit to a one that is a times smaller, all numerical values are multiplied by a, so that $x \mapsto a \cdot x$.
- Similarly, if we change the starting point to the one which is b units smaller, b is added to all numerical values.
- For example, 20° C becomes $20 + 273.16 = 293.16^\circ$ K.
- In general, we can have linear transformations $x \mapsto a \cdot x + b$.
- Similarly, we can have a linear transformation $y \mapsto A \cdot y + B$.
- In neural training, we can use data corresponding to different measuring units.

11. Invariance (cont-d)

- The relative quality of a machine learning algorithm should not change if we simply use, e.g., centimeters instead of meters.
- Use of different units means that instead of the original function $f(x)$, we consider a new function $T_{a,b,A,B}(f) \stackrel{\text{def}}{=} A \cdot f(a \cdot x + b) + B$.
- We say that functions $f(x)$ and $g(x)$ are *linearly equivalent* if there exist $a > 0$, b, $A < 0$, and B for which $g = T_{a,b,A,B}(f)$.
- If we apply the transformation $T_{a,b,A,B}$ to all the functions from the original family $\mathcal{F}$, we thus get a new family.
- Let us denote this new family by $T_{a,b,A,B}(\mathcal{F})$.
- We say that an optimality criterion is *linearly invariant* if for every two families $\mathcal{F}$ and $\mathcal{G}$ and for all $a > 0$, b, $A > 0$, and B:
  - $\mathcal{F} \succ \mathcal{G}$ implies $T_{a,b,A,B}(\mathcal{F}) \succ T_{a,b,A,B}(\mathcal{G})$, and
  - $\mathcal{F} \sim \mathcal{G}$ implies $T_{a,b,A,B}(\mathcal{F}) \sim T_{a,b,A,B}(\mathcal{G})$.

12. Main Result and Its Proof

- The smallest k for which there exists a final linearly invariant optimality criterion on the set of all k-families is $k = 3$.
- For this k, every function from the optimal family is linearly equivalent either to $\log(x)$, or to $\exp(x)$, or to x^α for some α.
- Let us prove this result.
- Since the criterion is final, there exists the unique optimal family $\mathcal{F}_{\rm opt}$.
- Let us first prove that this optimal family is itself linearly invariant, i.e., that $T_{a,b,A,B}(\mathcal{F}_{\rm opt}) = \mathcal{F}_{\rm opt}$ for all a, b, A, and B.
- Indeed, by definition of the optimal family, for each other family $\mathcal{F}$, we have either $\mathcal{F}_{\rm opt} \succ \mathcal{F}$ or $\mathcal{F}_{\rm opt} \sim \mathcal{F}$.
- In particular, for each family $\mathcal{F}$, we have either $\mathcal{F}_{\rm opt} \succ T^{-1}_{a,b,A,B}(\mathcal{F})$ or $\mathcal{F}_{\rm opt} \sim T^{-1}_{a,b,A,B}(\mathcal{F})$.
- Here $T^{-1}_{a,b,A,B}$ denotes the inverse linear transformation.

13. Proof (cont-d)

- Since the optimality criterion is linearly-invariant, we can apply the transformation $T_{a,b,A,B}$ to both sides of the corresponding relations.
- Thus, we conclude that for every $\mathcal{F}$, either $T_{a,b,A,B}(\mathcal{F}_{\rm opt}) \succ \mathcal{F}$ or $T_{a,b,A,B}(\mathcal{F}_{\rm opt}) \sim \mathcal{F}$.
- By definition of the optimal alternative, this means that the family $T_{a,b,A,B}(\mathcal{F}_{\rm opt})$ is also optimal.
- But since the optimality criterion is final, there can be only one optimal family.
- Thus, $T_{a,b,A,B}(\mathcal{F}_{\rm opt}) = \mathcal{F}_{\rm opt}$.
- This equality means, in particular, that:
  - for every function $f(x)$ from the optimal family $\mathcal{F}_{\rm opt}$ and for all possible values $a > 0$, b, $A > 0$, and B,
  - the function $A \cdot f(a \cdot x + b) + B$ also belongs to this family.
- This expression has 4 parameters a, b, A, and B.

14. Proof (cont-d)

- For $k \le 3$, we cannot have all these functions to be different – that would lead to a 4-parametric family.
- Thus:
  - at least some 1-parametric family of transformations – that starts with the identity for $t = 0$
  - should keep the function un-changed.
- So, we should have

$$A(t) \cdot f(a(t) \cdot x + b(t)) + B(t) = f(x) \text{ for all } t.$$

- Differentiating both sides of this formula with respect to t, we get

$$A'(t) \cdot f(a(t) \cdot x + b(t)) + A(t) \cdot f'(a(t) \cdot x + b(t)) \cdot (a'(t) \cdot x + b'(t)) = 0.$$

- Here, as usual, the dash – such as A' – means the derivative.
- In particular, for $t = 0$, we have the identity transformation for which $a(0) = A(0) = 1$ and $b(0) = B(0) = 0$.

15. Proof (cont-d)

- Thus, for $t = 0$, the above formula takes the form of the following (implicit) differential equation:

$$A_0 \cdot f(x) + f'(x) \cdot (a_0 \cdot x + b_0) + B_0 = 0.$$

- Here we denoted $A_0 \stackrel{\text{def}}{=} A'(0)$, $a_0 \stackrel{\text{def}}{=} a'(0)$, $b_0 \stackrel{\text{def}}{=} b'(0)$, and $B_0 \stackrel{\text{def}}{=} B'(0)$.
- Let us move the terms proportional to $f'(x)$ to the right-hand side and take into account that $f(x) = y$.
- Then, the above formula takes the form

$$A_0 \cdot y + B_0 = -\frac{dy}{dx} \cdot (a_0 \cdot x + b_0).$$

- We can separate the variables if we divide both sides of this equality by $A_0 \cdot y + B_0$ and by $a_0 \cdot x + b_0$ and multiply both sides by $-dx$:

$$-\frac{dx}{a_0 \cdot x + b_0} = \frac{dy}{A_0 \cdot y + B_0}.$$

16. Proof (cont-d)

- To solve this differential equation, let us consider four possible cases, depending on whether a_0 is equal to 0 and whether A_0 is equal to 0.
- If $a_0 = 0$ and $A_0 = 0$, then integrating both sides, we simply get $\text{const} \cdot x = \text{const} \cdot y + C$, where C is the integration constant.
- In this case, y is a linear function of x.
- This contradicts to our definition of a family as a class of nonlinear functions.
- Thus, the case $a_0 = A_0 = 0$ is not possible.

17. Proof: case when $a_0 = 0$ and $A_0 \neq 0$

- If $a_0 = 0$ and $A_0 \neq 0$, then $-\dfrac{dx}{b_0} = \dfrac{dy}{A_0 \cdot y + B_0}$.
- If we introduce a new variable $Y \stackrel{\text{def}}{=} A_0 \cdot y + B_0$, then $dY = A_0 \cdot dy$.
- So, if we multiply both sides by A_0, we get
$$-\frac{A_0}{b_0} \cdot dx = \frac{A_0 \cdot dy}{A_0 \cdot y + B_0} = \frac{dY}{Y}.$$
- If we now introduce a new variable $X \stackrel{\text{def}}{=} (-A_0/b_0) \cdot x$, then the equation takes the form $dX = \dfrac{dY}{Y}$.
- Integrating both sides, we get $X + C = \ln(Y)$, hence
$$Y = \exp(X + C) = \text{const} \cdot \exp(X).$$
- Since Y is a linear function of y and C is a linear function of x, it follows that the function $y(x)$ is linearly equivalent to $\exp(x)$.

18. Proof: case when $a_0 \neq 0$ and $A_0 = 0$

- If $a_0 \neq 0$ and $A_0 = 0$, then $-\dfrac{dx}{a_0 \cdot x + b_0} = \dfrac{dy}{B_0}$.
- If we introduce a new variable $X \stackrel{\text{def}}{=} a_0 \cdot x + b_0$, then $dX = a_0 \cdot dx$.
- So, if we multiply both sides of the above equation by $-a_0$, we get

$$\frac{a_0 \cdot dx}{a_0 \cdot x + b_0} = \frac{dX}{X} = -\frac{a_0}{B_0} \cdot dy.$$

- If we introduce a new variable $Y \stackrel{\text{def}}{=} -(a_0/B_0) \cdot y$, then the equation takes the form $dY = \dfrac{dX}{X}$.
- Integrating both sides, we get $Y = \ln(X) + C$.
- Since Y is a linear function of y and C is a linear function of x, it follows that the function $y(x)$ is linearly equivalent to $\ln(x)$.

19. Proof: case when $a_0 \neq 0$ and $A_0 \neq 0$

- If we multiply both sides of the equation by A_0, we get
$$-\frac{A_0}{a_0} \cdot \frac{a_0 \cdot dx}{a_0 \cdot x + b_0} = \frac{A_0 \cdot dy}{A_0 \cdot y + b_0}.$$
- If we introduce new variables $X \stackrel{\text{def}}{=} a_0 \cdot x + b_0$ and $Y \stackrel{\text{def}}{=} A_0 \cdot y + B_0$, then the equation takes the form
$$\alpha \cdot \frac{dX}{X} = \frac{dY}{Y}, \text{ where } \alpha \stackrel{\text{def}}{=} -A_0/a_0.$$
- Integrating both sides, we get $\ln(Y) = \alpha \cdot \ln(X) + C$, hence
$$Y = \exp(\alpha \cdot \ln(X) + C) = \exp(\alpha \cdot \ln(X)) \cdot \exp(C) =$$
$$\exp(C) \cdot (\exp(\ln(X))^\alpha = \text{const} \cdot X^\alpha.$$
- Since Y is a linear function of y and X is a linear function of x, it follows that the function $y(x)$ is linearly equivalent to x^α.

20. Conclusions

- Modern machine-learning-based AI systems have led to spectacular successes.
- However, their results are often not perfect, they need to be trained better.
- Already training takes a lot of computation time, up to years.
- So to train better, we need to speed up computations.
- It has been shown:
  - that we can speed up computations if we decrease the range of the values by applying an appropriate nonlinear transformation to all the values, and
  - that for this, logarithmic transformation leads to the best speedup.
- In this talk, we provide a theoretical explanation for this result.
- Namely, we prove that under reasonable conditions, log transformation is indeed one of the optimal ones.

21. Future Plans

- Can we do better?
- In our study, we considered families of transformations with the smallest possible number of parameters.
- Maybe families with more parameters will lead to further speedup?

22. Acknowledgments

This work was supported in part:

- by the US National Science Foundation grants:
  - 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science),
  - HRD-1834620 and HRD-2034030 (CAHSI Includes),
  - EAR-2225395 (Center for Collective Impact in Earthquake Science C-CIES),
- by the AT&T Fellowship in Information Technology, and
- by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) Focus Program SPP 100+ 2388, Grant Nr. 501624329,