Why Deep Learning Is Under-Determined? Why Usual Numerical Methods for Solving Partial Differential Equations Do Not Preserve Energy? The Answers May Be Related to Chevalley-Warning Theorem (and Thus to Fermat Last Theorem)

Julio C. Urenda, Olga Kosheleva, and Vladik Kreinovich

Abstract In this paper, we provide a possible explanation to two seemingly unrelated phenomena: (1) that in deep learning, under-determined systems of equations perform much better than the over-determined one – which are typical in data processing, and that (2) usual numerical methods for solving partial differential equations do not preserve energy. Our explanation is related to the intuition of Fermat behind his Last Theorem and of Euler about more general statements, intuition that led to the proof of Chevalley-Warning Theorem in number theory.

1 Two Challenges: Formulation of the Problem

1.1 First challenge: why deep learning is under-determined?

Need for machine learning. In many practical situations:

- we know that quantity y depends on the quantities $x_1, \ldots, x_n$ – i.e., that the values $x_1, \ldots, x_n$ uniquely determine the value y,
- but we do not know the exact expression for this dependence $y = f(x_1, \ldots, x_n)$.

To find this dependence, we can use our knowledge of a large number (K) situations $k = 1, \ldots, K$, in each of which we know both:

Julio Urenda
Dapertments of Mathematical Sciences and Computer Science, University of Texas at El Paso
500 W. University, El Paso, Texas 79968, USA, e-mail: jcurenda@utep.edu

Olga Kosheleva
Department of Teacher Educat, University of Texas at El Paso, 500 W. University
El Paso, Texas 79968, USA, e-mail: olgak@utep.edu

Vladik Kreinovich
Department of Computer Science, University of Texas at El Paso, 500 W. University
El Paso, Texas 79968, USA, e-mail: vladik@utep.edu

- the values $x_i^{(k)}$ of the inputs x_i and
- the value $y^{(k)}$ of the output y.

The problem of determining the dependence $y = f(x_1, \ldots, x_n)$ based on such *patterns* $\left(x_1^{(k)}, \ldots, x_n^{(k)}, y^{(k)}\right)$ is known as *machine learning*.

Usual way to solve machine learning problems. To find the desired dependence, we usually:

- select a finite-parametric family of functions

$$y = f(x_1, \ldots, x_n, c_1, \ldots, c_m)$$

  and then
- use the known patterns $\left(x_1^{(k)}, \ldots, x_n^{(k)}, y^{(k)}\right)$ to find the values of the parameters c_j that fit this data, i.e., for which

$$f\left(x_1^{(k)}, \ldots, x_n^{(k)}, c_1, \ldots, c_m\right) \approx y^{(k)}. \tag{1}$$

Regression vs. general machine learning. In traditional statistical data processing (see, e.g., [8]), this problem is called *regression*. In most of these methods, we have an explicit formula describing the dependence

$$f(x_1, \ldots, x_n, c_1, \ldots, c_m).$$

For example, we can have *linear regression*

$$f(x_1, \ldots, x_n) = c_1 \cdot x_1 + \ldots + c_n \cdot x_n + c_{n+1}.$$

Alternatively, we can have quadratic regression, etc.

In other machine learning techniques – e.g., in neural networks – the dependence on c_j is not given explicitly. For example, in neural networks, the values c_j correspond to the "weights" of the corresponding neurons. To be more precise:

- for each neuron, the dependence of its output signal Y on its inputs $X_1, \ldots, X_\ell$ is described by an explicit formula

$$Y = s(w_1 \cdot X_1 + \ldots + w_\ell \cdot X_\ell + w_{\ell+1})$$

  for some given non-linear function known as the *activation function*;
- however, the input-output relation of the whole neural network – in which some neurons process the inputs while the others process the neuron's outputs – is a composition of many neuron-related functions, a composition so complicated that it is not feasible to provide its explicit expression.

Our experience with regression: under- and over-determined systems of equations. In statistical regression, to determine m parameters $c_1, \ldots, c_m$, we have K equations (1) – corresponding to K available patters.

In the idealized situation, when all the values are known exactly, we need m equations to determine m unknowns, i.e., we need exactly m patterns. If we have fewer than m patterns – and thus, fewer equations – the corresponding system of equations (1) has, in general, many possible solutions. In this case, we cannot uniquely determine the values of the parameters c_j and thus, we cannot determine the desired function: the system of equations is *under-determined*.

In practice, the values x_i and y are always known with uncertainty – be it measurement error or inaccuracy of expert's estimates. As a result, if we have exactly as many equations as unknown, the coefficients c_i will also be determined with uncertainty. To decreases this uncertainty, we can use additional measurements – i.e., we can use more patterns than unknowns. In this case, the system of equations is *over-determined*, and this over-determination helps improve the accuracy.

Because of the measurement errors, if we use exactly as many unknowns as there are patterns, we fit exactly the values containing actual measurement errors. This over-fitting decreases the prediction accuracy,

For example, suppose that we consider a linear dependence $y = c_1 \cdot x_1 + c_2$, in which the actual dependence is $y = x$, corresponding to $c_1 = 1$ and $c_2 = 0$. Due to measurement errors, we may have:

- $y^{(1)} = -0.1$ for $x_1^{(1)} = 0$ and
- $y^{(2)} = 1.1$ for $x_1^{(1)} = 1$.

If we exactly fit a linear dependence to these two patterns, we get $y = 1.2 \cdot x_1 - 0.1$.

- For $x_1 = 10$, the predicted value is $y = 1.2 \cdot 10 - 0.1 = 11.9$ while the actual value is $y = 10$ – the difference is 1.9.
- For $x_1 = 100$, the difference between prediction and actual value will be 19.9, etc.

Paradoxical situation with deep learning. With deep learning, when we first increase the number of unknowns to be equal to the number of patterns, we observe the exact same over-fitting. However, interestingly, when we further increase the number of parameters c_j, the prediction error decreases. This is what leads to spectacular successes of deep learning – that when we use a network with billions of parameters to fit millions (or even thousands) of patterns, we get an almost perfect prediction. How to explain this is the first challenge that we consider in this paper.

1.2 Second challenge: why usual numerical methods for solving partial differential equations do not preserve energy

Physical equations and how we solve them: reminder. Many physical processes are described by partial differential equations (PDEs); see,e.g., [3, 9]. In some special cases, these equation have an explicit solution. In most cases, the only way to solve these equations it to use numerical methods.

Numerical methods provide only an approximate solution to the actual equation. The tighter the grid we get or the smaller the finite elements we take, the more accurate this approximation – but for any given grid size or finite element size, there is a non-zero approximation error.

Usual numerical methods for solving PDEs do not preserve energy: a challenge. In most physical situations, there are quantities such as energy that get preserved with time. However, for the approximate solutions, energy is usually *not* exactly preserved, it is preserved only approximately. This leads to non-physical solutions where the energy, instead of being constant, starts growing or decreasing.

At first glance, why cannot we have a numerical method in which energy is preserved? Energy conservation follows from the differential equations, so, in principle, if we replace one of the numerical equations with the energy conservation law, we should get the equivalent system of numerical equations in which energy is automatically preserved – but this is not happening. Why?

1.3 What we do in this paper

In this paper, we provide a possible explanation of these two challenges.

2 Possible Explanation

Commonsense intuition that makes the two above-described phenomena paradoxical. The reason why we consider both above-described phenomena counterintuitive is because we are using the intuitive idea that for the system of equations to have a unique solution, in general, the number of equations should be equal to the number of unknowns.

This intuition is based on – and confirmed by – our study of systems of linear equations, where indeed, this is, in general, true. For example, in general, if we have fewer equations than unknowns, then the system has multiple solutions – and, in particular, if the corresponding system has a zero solution, it must have a non-zero solution as well.

But is this intuition correct for nonlinear equations? In contrast to well-studied systems of linear equations, systems of non-linear equations are not so easy to analyze and are, therefore, not as well-studied, even when these systems consist of the next simplest – quadratic (or, more generally, polynomial) equations.

However, there is a similar problem for which a lot is known: namely, the solutions of system of polynomial equations, in which the unknowns are not real numbers, but elements of a *finite field*. A finite field is a finite set with addition and multiplication that satisfy the usual properties like commutativity, associativity, and distributivity, and in which each non-zero element has an inverse. The sim-

plest finite fields can be obtained if we take a prime number p, and consider the set $\{0,1,\ldots,p-1\}$ of all possible remainders when we divide by p. On this set, we can define addition and multiplication as addition and multiplication modulo p.

For such fields, there is also a result that – under certain condition – if a system of polynomial equations has a zero solution, then it must have a non-zero solution. However, the difference from the linear case is that this condition is, in general, *different* from a simple inequality $m > K$ between the number of unknowns m and the number K of equations. Instead, this condition takes a more complex form

$$m > \sum_{k=1}^{K} d_k, \tag{2}$$

where d_k is the degree of the polynomial in the k-th equation.

- For a system of linear equations, when $d_1 = \ldots = d_K = 1$, the condition (2) leads exactly to the usual condition $m > K$.
- However, when some of these equations are non-linear, with $d_k > 1$, the right-hand side of the inequality (2) becomes larger than K – so we need more unknowns to guarantee the existence of a non-zero solution.

In general, due to the formula (2), the smallest value m for which we have more than one solution is the value $\sum d_k + 1$. Thus, similarly to the linear case, it is reasonable to conclude that the value m for which we have exactly one solution is the next smallest value, i.e., $\sum d_k$.

Historical comment and how this is related to Fermat last theorem. The above result is known as *Chevalley-Warning Theorem*; see, e.g., [1, 10]. This result has been motivated by many specific example, one of which is Fermat Last Theorem, according to which:

- while we can have $x_1^2 + x_2^2 = x_3^2$ for many triples of positive integers – starting with $x_1 = 3$, $x_2 = 4$, and $x_3 = 5$,
- for any integer degree $d \geq 3$, a similar equation $x_1^d + x_2^d = x_3^d$ has no solution in which all x_i are positive integers.

Indeed, in all these cases, we have $m = 3$ unknown and a single equation of degree $d_1 = d$, so that $\sum d_j = d$. Here:

- for the degree $d = 2$, we have $m = 3 > d = 2$, in line with the fact that the equation $x_1^2 + x_2^2 = x_3^2$ has a positive integer solution – actually, it has infinitely many solutions;
- in contrast, for degrees $d \geq 3$, the inequality $m > d$ is no longer satisfied, in line with the fact that the corresponding equations hve no solutions.

(Of course, this is by no mean a proof of Fermat Last Theorem – Chevalley-Warning Theorem does not prevent the existence of non-zero solutions for *some* systems with $n \leq \sum d_k$ – but it was, as we have mentioned, one of the motivations behind this mathematical result.)

Similar ideas were discussed by Euler, who conjectured that for any degree d, it is only possible to have a more general equality $x_1^d + \ldots + x_k^d = x_{k+1}^d$ when $k > d$; see, e.g., [2, 5]. Clearly, Euler had an inequality like (2) in mind. By the way, this Euler's hypothesis turned out to be not true: for example, we have $24^5 + 84^5 + 110^5 + 133^5 = 144^5$; see, e.g., [6]. Later, other counterexamples to Euler's hypothesis were found. After the discovery of the first counterexample, a slightly modified – and generalized – version of Euler;s hypothesis was proposed: that if $x_1^d + \ldots + x_k^d = x_{k+}^d + \ldots + x_{k+\ell}^d$ for positive integers x_i, then $k + \ell \geq d$ [7]. This hypothesis is still an open problem: no counter-example has been found (and no proof either).

How this explain the non-conservation of energy. Let us show that the Chevalley-Warning Theorem explains both above-mentioned paradoxes. Let us start with explaining why energy is usually not conserved for numerical solutions of physical PDEs.

Indeed, usually, numerical methods of solving partial differential equations are based on the fact that for small changes in time, from moment t to moment $t + \Delta t$, the values $x_i(t + \Delta)$ are close to the corresponding values $x_i(t)$. Thus, we can ignore terms which are quadratic or of higher order in terms of the difference $x_i(t + \Delta) - x_i(t)$ and hence, keep all equations for determining the new values $x_i(t + \Delta)$ linear. To find N such values, we thus have a system of N linear equations – exactly as much as needed to uniquely determine all these values.

However, energy is usually a quadratic (or, sometimes, higher order) function of the corresponding physical quantities; a good example is kinetic energy

$$\frac{1}{2} \cdot mv^2$$

which is proportional to the square of the velocity v. Thus, the equation that describes the energy at the moment $t + \Delta t$ is equal to the energy at the moment t is also a quadratic equation. So, if we replace one of the original linear equations with this quadratic equation:

- the number of equations remains the same, but
- the quantity $\sum d_j$ increases by 1.

Thus, here, we still have $m = N$ unknown, but now the quantity in the right-hand side of (2) is equal to $N + 1$. So:

- while previously, we had $m = \sum d_k$, which – for linear systems – guarantees a solution,
- now we have $m < \sum d_k$ – so the condition (2) guaranteeing the existence of a non-zero solution is not satisfied.

This provides a possible explanation of why energy is not preserved for numerical solutions – since a system of equation containing energy conservation, in general, may not have a solution.

How this explains why in effective deep learning, the number of parameters is much larger than the number of equations. Neurons are non-linear. Thus, equations that related the input and output of a neural network are nonlinear. If all these equations have degree d, then, according to our argument, the smallest value m for which the corresponding system of equations has a solution – i.e., for which we can fit all the patterns – is equal to $\sum d_k = d \cdot K$.

Activation functions used in neural networks are not polynomial. While every function can be approximated, with any given accuracy, by a polynomial, we need polynomials of high degree d to accurately represent these functions. Thus, the number of parameters $m = d \cdot K$ that we need to accurately describe K patterns indeed has to be much larger than the number K of patterns K – which is exactly the surprising phenomenon that we observe in deep learning. So, we indeed have a possible explanation for this challenge as well.

3 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), HRD-1834620 and HRD-2034030 (CAHSI Includes), EAR-2225395, 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, and by a grant from the Hungarian National Research, Development and Innovation Office (NRDI).

References

1. C. Chevalley, "Démonstration d'une hypothése de M. Artin", *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 1935, Vol. 11, pp. 73–75, doi:10.1007/BF02940714.
2. L. E. Dickson, *History of the Theory of Numbers, Vol. 2: Diophantine Analysis*, American Mathematical Society, Providence, RDhode Island, 1999.
3. R. Feynman, R. Leighton, and M. Sands, *The Feynman Lectures on Physics*, Addison Wesley, Boston, Massachusetts, 2005.
4. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, Massachusetts, 2016.
5. R. Guy, *Unsolved Problems in Number Theory*, Springer Verlag, Berlin, Heidelberg, New York, 2010.
6. L. J. Lander and T. P. Parker, "Counterexample to Euler's conjecture on sums of like powers", *Bulletin of the American Mathematical Society*, 1966, Vol. 72, p. 1079.
7. L. J. Lander, T. R. Parkin, and J. L. Selfridge, "A survey of equal sums of like powers", *Mathematics of Computation*, 1967. Vol. 21, No. 99, pp. 446–459.
8. D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman and Hall/CRC, Boca Raton, Florida, 2011.

9. K. S. Thorne and R. D. Blandford, *Modern Classical Physics: Optics, Fluids, Plasmas, Elasticity, Relativity, and Statistical Physics*, Princeton University Press, Princeton, New Jersey, 2021.
10. E. Warning, "Bemerkung zur vorstehenden Arbeit von Herrn Chevalley", *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 1935, Vol. 11, pp. 76–83, doi:10.1007/BF02940715.