University of Texas at El Paso
ScholarWorks@UTEP

Departmental Technical Reports (CS) | Computer Science

6-1-2022

Computational Paradox of Deep Learning: A Qualitative Explanation

Jonatan Contreras
The University of Texas at El Paso, jmcontreras2@utep.edu

Martine Ceberio
The University of Texas at El Paso, mceberio@utep.edu

Olga Kosheleva
The University of Texas at El Paso, olgak@utep.edu

Vladik Kreinovich
The University of Texas at El Paso, vladik@utep.edu

Nguyen Hoang Phuong
Thang Long University, nhphuong2008@gmail.com

Follow this and additional works at: https://scholarworks.utep.edu/cs_techrep

 Part of the Computer Sciences Commons, and the Mathematics Commons

Comments:
Technical Report: UTEP-CS-22-74

Recommended Citation

Contreras, Jonatan; Ceberio, Martine; Kosheleva, Olga; Kreinovich, Vladik; and Phuong, Nguyen Hoang, "Computational Paradox of Deep Learning: A Qualitative Explanation" (2022). *Departmental Technical Reports (CS)*. 1714.
https://scholarworks.utep.edu/cs_techrep/1714

This Article is brought to you for free and open access by the Computer Science at ScholarWorks@UTEP. It has been accepted for inclusion in Departmental Technical Reports (CS) by an authorized administrator of ScholarWorks@UTEP. For more information, please contact lweber@utep.edu.

Computational Paradox of Deep Learning: A Qualitative Explanation

Jonatan Contreras, Martine Ceberio, Olga Kosheleva, Vladik Kreinovich, and Nguyen Hoang Phuong

Abstract In general, the more unknowns in a problem, the more computational efforts is necessary to find all these unknowns. Interestingly, in state-of-the-art machine learning methods like deep learning, computations become easier when we increase the number of unknown parameters way beyond the number of equations. In this paper, we provide a qualitative explanation for this computational paradox.

1 Formulation of the Problem

Regression/machine learning: a brief reminder. In many practical situations, we need to find a dependence $y = f(x_1, \ldots, x_n)$ between different quantities – based on several cases $k = 1, 2, \ldots, K$ in which we know the values $x_1^{(k)}, \ldots, x_n^{(k)}$ and $y^{(k)}$ of all related quantities. In statistics, this problem is known as *regression*, in computer science, it is known as *machine learning*.

The usual approach to solving this problem is to select a family of functions $f(x_1, \ldots, x_n, c_1, \ldots, c_m)$ and select the values of the parameters $c_1, \ldots, c_m$ for which, for all k from 1 to K, we have

Jonatan Contreras, Martine Ceberio, and Vladik Kreinovich
Department of Computer Science, University of Texas at El Paso
500 W. University, El Paso, Texas 79968, USA
e-mail: jmcontreras2@miners.utep.edu, mceberio@utep.edu, vladik@utep.edu

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

Nguyen Hoang Phuong
Division Informatics, Math-Informatics Faculty, Thang Long University, Nghiem Xuan Yem Road
Hoang Mai District, Hanoi, Vietnam, e-mail: nhphuong2008@gmail.com

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

Sometimes, we select an explicit formulation of this family – e.g., in linear regression, we select linear functions:

$$f(x_1, \ldots, x_n, c_1, \ldots, c_{n+1}) = c_1 \cdot x_1 + \ldots + c_n \cdot x_n + c_{n+1}; \tag{2}$$

in other cases, we select the family of all quadratic functions, etc.

In many machine learning techniques, the corresponding family is described indirectly – e.g., in neural networks, the values c_i are weights of connections between different neurons.

It is known that the efficiency of regression/machine learning depends on the proper section of the corresponding family, in particular, on the number of parameters m.

In general, computational complexity increases with the number of unknowns. In general, the larger the number of unknowns, the more computational effort we need to solve the corresponding problem. For example, optimizing a function of one variable is easier than optimizing a function of two or three variables; solving a single equation is usually computationally easier than solving a system of several equations, etc.

Under-fitting and over-fitting: traditional statistical approach (see, e.g., [23]). As we have mentioned, the efficiency of regression/machine learning depends on the proper section of the corresponding family, in particular, on the number of parameters m.

If we select the value m to be too small, we may not get a good fit – e.g., if we only consider linear regression and the actual dependence is highly non-linear. This is known as *under-fitting*.

On the other hand, if we select the value m to be too high, e.g., if we choose $m \geq K$, then (1) becomes a system of K equations with $m \geq K$ unknowns. In general, in such cases, we can get exact equality in all K equations, i.e., we can have:

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

For $m > K$, we have more unknowns than equations, so we have several possible solutions for which there is exact equality (1a). The problem is that the measurement results $x_1^{(k)}, \ldots, x_n^{(k)}$ and $y^{(k)}$ come with random measurement errors, and the exact fit means that we follow this randomness – and thus, we get a not very good prediction accuracy.

For example, suppose that the actual value of y is constant $y = 1$, and we try to exactly fit the measurement results $y^{(1)} = 1.0$ and $y^{(2)} = 1.1$ corresponding to $x_1^{(1)} = 0$ and $x_1^{(2)} = 1$ with the linear dependence $y = c_1 \cdot x_1 + c_2$. Then the corresponding equations (1a), which in this case take the form

$$y^{(1)} = c_1 \cdot x^{(1)} + c_2 \text{ and } y^{(2)} = c_1 \cdot x^{(2)} + c_2$$

lead to $c_1 = 0.1$ and $c_2 = 1$, i.e., to the formula $y = 0.1 \cdot x + 1$. For large x, the predicted value can be very different from the desired value $y = 1$. This phenomenon is known as *over-fitting*.

First (accuracy-related) paradox of deep learning: going beyond supposed over-fitting increases prediction accuracy. At present, the most efficient machine learning techniques are the techniques of deep learning; see, e.g., [4]. These techniques require a large amount of data to be successful; in situations when we do not have that many available data points, other machine learning techniques work better.

Interestingly, deep learning and other state-of-the-art machine learning techniques often achieve their successes when the number of parameters m is much larger than the number K of data points – contrary to what the traditional statistics predicts. Researchers have studied this phenomenon in detail by gradually increasing m; see, e.g., [2, 3, 22]. They found out that as m increases beyond K, at first, the prediction accuracy decreases – exactly as predicted by traditional statistics. However, as the value m increases further, the prediction accuracy starts increasing again.

An explanation of the first (accuracy-related) paradox. A mathematical explanation of this paradox has been proposed in [2, 3].

Remaining problem: computational paradox of deep learning. While a solution to the accuracy-related paradox of deep learning is known, there is another deep-learning-related paradox – mentioned in [2, 3] – for which there is, at present, no satisfactory solution.

Specifically, it turned out that as the number m of parameters increases, solving the corresponding system of approximate equations – which usually means minimizing some functional (like least squares) $F(c)$, where $c \stackrel{\text{def}}{=} (c_1, \ldots, c_m)$ that describes the discrepancy between the left- and right-hand sides – becomes computationally easier; see, e.g., [21, 24].

What we do in this paper. In this paper, we provide a possible qualitative explanation for this computational paradox.

2 General Explanation

The proposed qualitative explanation is based on the following two known results.

First known result: uniqueness makes computations easier. It is known that, in general, uniqueness makes solutions easier; namely:

- for many general computational problems, there is an algorithm that solves all the cases in the which the solution is unique; see, e.g., [5, 6, 7, 9, 11, 15, 16, 17, 19]; this is true for all kinds of problems – optimization problems, solving systems of equations, etc.;

- on the other hand, for these same general classes of problems, it can be proven that no algorithm is possible that would solve all the cases in which the problem has exactly two solutions; see, e.g., [8, 9, 10, 11, 12, 13, 14, 15, 20].

There exist good arguments that a similar phenomenon occurs for problems for which a general algorithm is possible: namely, these arguments show that, in general, cases for which there exists a unique solution are easier to solve; see, e.g., [1].

Comment. Some of these results are technically difficult, so we will not provide the proofs, we only provide links to papers where these proofs are described.

Second known result: a random function attains its optimum at a single point. The second result – which is not that technically difficult – is related to the study of random functions, i.e., reasonable probability measures on reasonable sets of functions $F(c)$. An important feature of such probability measures is that for each reasonable numerical characteristic $v(F)$ of a function $F(c)$ (such as its integral or its maximum or minimum over a certain area), this characteristic should also be "random" in the intuitive sense of this word. In particular, this means:

- that for each real number r, the probability that we have $v(F) = r$ is equal to 0, and
- that for every two different characteristics $v \neq v'$, the probability that they have the same value, i.e., that $v(F) = v'(F)$, is also equal to 0.

Let us show how this applies to minima of random functions.

Suppose that a function $F(c)$ attains its minimum value m for each two different inputs $s \neq t$:

$$F(s) = F(t) = \min_c F(c). \tag{3}$$

Then, we can find a hyperplane H with rational coefficients that separates these inputs s and t: namely, s is in one of the half-spaces H_- corresponding to this hyperplane, while t is in another of these two half-spaces H_+. Since $s \in H_-$ and $F(s) = m$, we have

$$m = \min_c F(c) \le \min_{c \in H_-} F(c) \le F(s) = m,$$

thus

$$\min_{c \in H_-} F(x) = m.$$

Similarly, due to $t \in H_+$ and $F(t) = m$, we have

$$\min_{c \in H_+} F(c) = m.$$

Thus, we have:

$$\min_{c \in H_-} F(c) = \min_{c \in H_+} F(c). \tag{4}$$

In line with the above-described meaning of a reasonable probability measure, the probability that two characteristics

$$\max_{c \in H_-} F(c) \text{ and } \max_{c \in H_+} F(c)$$

have equal values – as in formula (4) – is 0.

There are countably many rational numbers, so there are countably many hyperplane with rational coefficients. For each of them, the probability of equality (4) is zero. Thus, the probability that the equality (4) happens for some hyperplane with rational coefficients is also 0 – since the union of countably many sets of probability measure 0 also has probability measure 0.

Thus, the probability that a function attains its minimum at two (or more) different inputs is 0. Therefore, a *random* function – i.e., a function that does not belong to any definable set of measure 0, or, alternative, that satisfies every law that is true for almost all functions (see, e.g., [18]) – cannot attain its minimum at two or more different points. So, a random function attains its minimum at a single point.

Resulting explanation. Now, we are ready to present our explanation.

In the usual statistical case, as we have mentioned earlier, when the number of parameters exceeds the number of unknowns, we have a non-unique solution to the corresponding fitting problem.

In deep learning, we do not simply make a fit, there is a lot of randomness involved: e.g., initial value of all the coefficients are random. This randomness is not just a feature of a specific training algorithm, it is inevitable: otherwise, if we start with the same deterministic values of the weights, and use some deterministic algorithm for training, all neurons in each layer will have the same weights, so their outputs will simply uselessly duplicate each other.

Because of this randomness, the actual objective function corresponding to several training steps of a neural network is random and thus, attains its minimum at only one point. So:

- in the traditional regression case, we have a problem with multiple solutions, while
- in the neural network case, we have a problem with a single solution.

Since, as we have mentioned, problems with a single solution are easier to solve than problems with multiple solutions, this explains, on the qualitative level, why optimization related to training a neural network turns out to be computationally easier than the optimization related to the usual multi-parameter regression.

3 Specific Explanation Related to Gradient-Based Training of Neural Networks

Gradient descent: a brief reminder. For neural networks, the above general explanation can be supplemented by specific details. These details are based on the fact that training of a neural network is based on gradient descent (see, e.g., [4]), when on each iteration, we replace the previous values c_i of all the parameters by a new

value

$$c_i' = c_i + \Delta c_i, \tag{5}$$

where

$$\Delta c_i = -\alpha \cdot \frac{\partial F}{\partial c_i} \tag{6}$$

for some constant α. After this change, the value of the objective function changes from the previous value $F(c_1, \ldots, c_m)$ to the new value

$$F(c_1', \ldots, c_m') = F(c_1, \ldots, c_m) + \Delta F, \tag{7}$$

where we denoted:

$$\Delta F \stackrel{\text{def}}{=} F(c_1', \ldots, c_m') - F(c_1, \ldots, c_m) =$$

$$F(c_1 + \Delta c_1, \ldots, c_m + \Delta c_m) - F(c_1, \ldots, c_m). \tag{8}$$

The changes Δc_i are usually small. So, to estimate ΔF, we can expand the right-hand side of the formula (8) in Taylor series in terms of Δc_i and keep only linear terms in this expansion. As a result, we get

$$\Delta F = \sum_{i=1}^{m} \frac{\partial F}{\partial c_i} \cdot \Delta c_i. \tag{9}$$

Substituting the expression (6) into this formula, we conclude that

$$\Delta F = -\alpha \cdot \sum_{i=1}^{m} \left(\frac{\partial F}{\partial c_i} \right)^2. \tag{10}$$

Resulting explanation. As we have mentioned, to fit K equalities (1a), it is sufficient to use K parameters $c_1, \ldots, c_K$. In this case, after each iteration of the gradient descent, the value of the objective function decreases by the value

$$\Delta F_K = -\alpha \cdot \sum_{i=1}^{K} \left(\frac{\partial F}{\partial c_i} \right)^2. \tag{11}$$

If we use all $m > K$ parameters, then the resulting decrease (10) in the value of the minimized objective function $F(c)$ is even larger – and, the larger m, the larger this decrease. So, indeed, if we increase the number of coefficients, gradient descent becomes much more efficient than for smaller number of coefficients – and this is exactly what was empirically observed in [21, 24].

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), 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. R. Beigel, H. Buhrman, and L. Fortnow, "NP might not be as easy as detecting unique solutions", In: *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (STOC)*, 1998, pp. 203–208.
2. M. Belkin, "Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation", *Acta Numerica*, 2021, Vol. 30, pp. 203–248.
3. M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-leaning practice and the classical bias-variance trade-off", *Proceedings of the National Academy of Science of USA*, 2019, Vol. 116, No. 32, pp. 15849–15854.
4. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, Massachusetts, 2016.
5. U. Kohlenbach. *Theorie der majorisierbaren und stetigen Funktionale und ihre Anwendung bei der Extraktion von Schranken aus inkonstruktiven Beweisen: Effektive Eindeutigkeitsmodule bei besten Approximationen aus ineffektiven Eindeutigkeitsbeweisen*, Ph.D. Dissertation, Frankfurt am Main, 1990.
6. U. Kohlenbach, Effective moduli from ineffective uniqueness proofs. An unwinding of de La Vallée Poussin's proof for Chebycheff approximation, *Annals for Pure and Applied Logic*, 1993, Vol. 64, No. 1, pp. 27–94.
7. U. Kohlenbach, *Applied Proof Theory: Proof Interpretations and their Use in Mathematics*, Springer Verlag, Berlin-Heidelberg, 2008.
8. V. Kreinovich, *Complexity measures: computability and applications*, Master Thesis, Leningrad University, Department of Mathematics, Division of Mathematical Logic and Constructive Mathematics, 1974 (in Russian).
9. V. Kreinovich, Uniqueness implies algorithmic computability, *Proceedings of the 4th Student Mathematical Conference*, Leningrad University, Leningrad, 1975, pp. 19–21 (in Russian).
10. V. Kreinovich, Reviewer's remarks in a review of D. S. Bridges, *Constrictive functional analysis*, Pitman, London, 1979; *Zentralblatt für Mathematik*, 1979, Vol. 401, pp. 22–24.
11. V. Kreinovich, *Categories of space-time models*, Ph.D. dissertation, Novosibirsk, Soviet Academy of Sciences, Siberian Branch, Institute of Mathematics, 1979 (in Russian).
12. V. Kreinovich, "Unsolvability of several algorithmically solvable analytical problems", *Abstracts Amer. Math. Soc.*, 1980, Vol. 1, No. 1, p. 174.
13. V. Kreinovich, *Philosophy of Optimism: Notes on the possibility of using algorithm theory when describing historical processes*, Leningrad Center for New Information Technology "Informatika", Technical Report, Leningrad, 1989 (in Russian).
14. V. Kreinovich and R. B. Kearfott, "Computational complexity of optimization and nonlinear equations with interval data", *Abstracts of the Sixteenth Symposium on Mathematical Programming with Data Perturbation*, The George Washington University, Washington, D.C., May 26–27, 1994.

15. V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1998.
16. V. Kreinovich and K. Villaverde, "Extracting computable bounds (and algorithms) from classical existence proofs: girard domains enable us to go beyond local compactness", *International Journal of Intelligent Technologies and Applied Statistics (IJITAS)*, 2019, Vol. 12, No. 2, pp. 99–134.
17. D. Lacombe, "Les ensembles récursivement ouvert ou fermés, et leurs applications à l'analyse récurslve", *Compt Rend.*, 1957, Vol. 245, No. 13, pp. 1040–1043.
18. M. Li and P. Vitanyi, *An Introduction to Kolmogorov Complexity and Its Applications*, Springer, Berlin, Heidelberg, New York, 2008.
19. V. A. Lifschitz, "Investigation of constructive functions by the method of fillings", *J. Soviet Math.*, 1973, Vol. 1, pp. 41–47.
20. L. Longpré, V. Kreinovich, W. Gasarch, and G. W. Walster, "m solutions good, $m-1$ solutions better", *Applied Math. Sciences*, 2008, Vol. 2, No. 5, pp. 223–239.
21. S. Ma, R. Bassily, and M. Belkin, "The power of interpolation: understanding the effectiveness of SGD in modern over-parameterized learning", in: J. Dy and A. Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, Proceedings of Machine Learning*, Stockholmsmässan, Stockholm, Sweden, 2018, Vol. 80, pp. 3325–3334.
22. D. Monroe, "A deeper understanding of deep learning: kernel methods clarify why neural networks generalize so well", *Communications of the ACM*, 2022, Vol. 65, No. 6, pp. 19–20.
23. D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman and Hall/CRC, Boca Raton, Florida, 2011.
24. M. Soltanolkotabi, A. Javanmard, and J. D. Lee, "Theoretical insight into the optimization landscape of over-parameterization shallow neural networks", *IEEE Transactions on Information Theory*, 2018, Vol. 65, pp. 742–769.