5-2016

Big Data: A Geometric Explanation of a Seemingly Counterintuitive Strategy

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

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

Comments:
Technical Report: UTEP-CS-16-27

Big Data: A Geometric Explanation of a Seemingly Counterintuitive Strategy

Olga Kosheleva and Vladik Kreinovich
University of Texas at El Paso
500 W. University
El Paso, TX 79968, USA
olgak@utep.edu, vladik@utep.edu

Abstract

Traditionally, the progress in science was usually achieved by gradually modifying known problem-solving techniques – so that the modified techniques can solve problems similar to the already-solved ones. Recently, however, a different – successful – paradigm of *big data* appeared. In the big data paradigm, we, in contrast, look for problems which *cannot* be solved by gradual modifications of the existing methods. In this paper, we propose a geometric explanation for the empirical success of this new paradigm.

1 Formulation of the Problem

Traditional approach to science. Traditional approach to science can be described by a metaphor of an increasing circle. Namely, at any given moment of time, there are problems that we can solve, and there are methods that enable us to solve these problems. A natural way to expand our knowledge is to look for not-yet-solved problems which are similar to the ones that we already know how to solve. The idea is that since the new problem is similar to the already solved ones, an appropriate small modification of the original methods will enable us to solve the new problem.

This is a usual way how science operates. For example, once it turned out that a simple system of differential equations describes Newton's mechanics, similar differential equations enabled us to describe other physical phenomena.

Big data approach: a new paradigm. In the new *big data* approach, instead of looking for unsolved problems which are *similar* to the previously solved ones – i.e., problems that could be solved by small modifications of the existing techniques, we instead look for problems which are *not solvable by such easy modification*. In other words, we explicitly look for problems that are very *dissimilar* to the already solved ones.

We then try to come up with methods for solving all such problems – or at least some of them.

Surprisingly, the big data approach has been successful, but why? Somewhat surprisingly, the big data approach has been successful (see, e.g., [2, 8, 8, 9, 10, 11, 13]): this way, researchers have indeed solved several important problems, and solutions to these problems helped solved other problems. But why?

- why searching for the most dissimilar problems is a good strategy? and
- why solving one or several of these dissimilar problems helps in solving other problems – problems very dissimilar to the ones that have previously solved?

In this paper, we provide a geometric explanation for this phenomenon.

2 Why Solution to One Hard Problem Helps to Solve Other Hard Problems: An Answer to the Second Question

Second question: reminder. Before we start answering the first of the above two questions – why searching for the most dissimilar problems help – let us discuss the second question: why solving one of the hard problems helps solving others.

Why we start with the second question. The reason why we start with the second question is that this question is not as counter-intuitive as the first one and is, actually, reasonably easy to answer. This answer comes from the theoretical computer science analysis of problem solvability.

Notions of NP, P, and NP-complete: a brief reminder. In most real-life problems, once we have a candidate for a solution, we can feasibly check whether this candidate is indeed a solution. Finding a solution may be difficult, but once we have found a solution, checking is correctness is straightforward. For example:

- In *mathematics*, it is usually difficult to come up with a proof or disproof of a mathematical hypothesis. However, once a detailed proof is presented, checking is correctness is straightforward.
- In *physics*, it may be difficult to come up with a formula that explains all observed phenomena in some area. However, once such a formula has been found, it is straightforward to check that this formula indeed covers all observations.
- In *engineering*, it may be difficult to come up with a design that satisfies all the specifications. However, usually, once the design is presented, it

is reasonably straightforward to check whether all the specifications are satisfied.

Such problems in which there is a feasible algorithm for checking whether a given candidate is a solution are called *problems from the class NP*; see, e.g., [12]. Some problems from this class can be solved by a feasible algorithm; the class of such problems is called P. Most computer scientists believe that P is different from NP, i.e., that there are problems from the class NP that cannot be solved by a feasible algorithm. However, as of now (2016), no one has yet succeeded in proving that P$\neq$NP; this is still an open problem. What has been proven is that within the class NP, some problems $\mathcal{P}_0$ are the hardest – in the sense that any other problem $\mathcal{P}$ from the class NP can be reduced to this problem $\mathcal{P}$. Such hardest problems are called *NP-complete*. This reduction means that for each instance of the problem $\mathcal{P}$, we can effectively find an instance of the problem $\mathcal{P}_0$ whose solution can feasibly lead to solving the original instance of the problem $\mathcal{P}$ [12].

Thus, if we have a heuristic algorithm that solves some instances of the problem $\mathcal{P}_0$, the above reduction enables us to automatically solve the corresponding instances of all other problems $\mathcal{P}$ from the class NP.

This is not just a theoretical possibility, this is a practically helpful phenomenon. For example, historically the first problem for which NP-completeness was proven is the *propositional satisfiability problem* SAT:

- *given* a propositional formula, i.e., an expression of the type

$$(v_1 \vee \neg v_2 \vee v_3) \,\&\, (\neg v_1 \vee v_2 \vee \neg v_3) \,\&\, \ldots$$

  obtained from propositional (true-false) variables v_i by using propositional connectives $\vee$ ("or'), $\&$ ("and"), and $\neg$ ("not"),

- *find* the truth values of the variables v_i that make the formula true.

The fact that SAT is NP-complete means that any problem from the class NP can be reduced to SAT. And indeed, SAT-solvers – heuristic algorithms for solving many instances of SAT – are widely used in solving many different problems from the class NP; see, e.g., [4].

Resulting answer to the second question. The above analysis shows that if we achieve progress is solving at least some instances of one hard problem, this automatically enables us – via the corresponding reduction – to gain some progress in solving all other hard problems as well.

This explains why in the big data approach, solving one of the hard problems helps solving others.

3 Why Searching for the Most Dissimilar Problems Is a Good Strategy: A Geometric Answer to the First Question

Main idea behind our explanation. Our ultimate goal is to solve all the problems in the world. The best strategy is the one that enables us to do it while minimizing our efforts.

Towards a formal description of the main idea. Knowledge acquisition processes can be divided into two stages:

- a more creative stage, when we come up with new ideas, and
- a somewhat less creative stage, when we simply develop the previous ideas.

Of course, this is a simplified description. Real-life knowledge acquisition process usually combines both types of processes.

Usually:

- if a problem is similar to one of the previously solved one, then we can solve this problem by a modification of the existing techniques, while
- problems which are very dissimilar with the previous ones requires a more creative approach.

Of course, in reality, there is a smooth transition between the two types of problems. However, for simplicity, let us assume that there exists a crisp threshold r_0 such that:

- if the distance $d(a, b)$ between problems a and b is smaller than or equal to r_0, then a solution to a can be transformed into a solution to b without the need to use radically new ideas, while
- if $d(a, b) > r_0$, then to transform a solution to a problem a into a solution to the problem b, we need to use a creative idea.

Between the two stages, the more creative stage clearly requires much more efforts. Thus, a natural way to minimize the overall effort is to minimize the overall number of creative ideas.

According to the above assumption, the sufficient number of creative ideas means that in the original metric space X of all the problems, we need to select a list of problems $a_1, \ldots, a_n$ for which every other problem is r_0-close to one of the problems a_i. This set is known as a r_0-*net*; see, e.g., [7].

Minimizing effort means finding the r_0-net with the smallest possible number of elements. The number of such elements is denoted by N_{r_0}; its logarithm $H_{r_0} = \ln(N_{r_0})$ is known as the r_0-*entropy* of the original metric space.

The resulting formal problem requires a heuristic approach. In general, the problem of computing N_{r_0} for a given metric space is known to be NP-hard

– i.e., as difficult (or maybe more difficult) that all NP-complete problems; see, e.g., [3, 5]. Thus, unless P=NP, no feasible algorithm is possible that would always find the desired r_0-net.

Since we cannot have a feasible algorithm that would *always* solve this problem, a natural idea is to have a heuristic algorithm that would *often* solve this problem.

Greedy approach and how it explains the big data paradigm. One of the known efficient heuristics for solving optimization problems is the *greedy approach* in which, instead of selecting all the points a_i at the same time, we select them one by one, each time selecting a_i so that the corresponding approximation to the desired objective function is as large as possible [1].

To come up with an appropriate greedy algorithm, let us take into account that, from the mathematical viewpoint, a problem of minimizing an objective function $f(x)$ under a constraint $g(x) \le c$ is equivalent to the Lagrange multiplier unconditional optimization problems $f(x)+\lambda\cdot g(x) \to \min$ and is, thus, also equivalent to the *dual* problem of minimizing $g(x)$ under the constraint $f(x) = c'$ for some c'. For our problem, this means that minimizing the number n of points under the constraint $\max\limits_{a\in X} d(a, \{a_1, \ldots, a_n\}) \le r_0$, where $d(a, B) \stackrel{\text{def}}{=} \min\limits_{b\in B} d(a, b)$, is equivalent to minimizing the value $s \stackrel{\text{def}}{=} \max\limits_{a\in X} d(a, \{a_1, \ldots, a_n\})$ under the condition that the number of points n is fixed.

In each stage of the greedy approach, we thus start with some values $a_1, \ldots, a_k$ for which we have some value $s_k = \max\limits_{a\in X} d(a, \{a_1, \ldots, a_k\})$. We want to select the next point a_{k+1} so that this value will be decreased. The maximum of s_k is attained at a point a' which is the farthest away from the points $a_1, \ldots, a_k$, i.e., for which the value $d(a, \{a_1, \ldots, a_k\})$ is the largest possible: $s_k = d(a', \{a_1, \ldots, a_k\})$. So, a reasonable way to minimize the objective function is to select the point a_{k+1} for which the value $v \stackrel{\text{def}}{=} d(a', \{a_1, \ldots, a_k, a_{k+1}\})$ is the smallest possible.

By definition of the distance between a point and a set, this value v is equal to the smallest of the following two numbers:

$$v = \min(d(a, \{a_1, \ldots, a_k\}), d(a', a_{k+1})).$$

The first of these two numbers does not depend on a_{k+1}. Thus, minimizing v is equivalent to minimizing the distance $d(a', a_{k+1})$. This distance is the smallest – and equal to 0 – when $a_{k+1} = a'$.

So:

- after we have selected the points $a_1, \ldots, a_k$,
- it is reasonable to select the point which is the farthest away from all the previous points as the next point a_{k+1}.

This is exactly the big data heuristics. Thus, the big data heuristic is indeed explained in geometric terms – as the use of greedy algorithm to solve the corresponding geometric optimization problem.

Comment. It is worth mentioning that the same geometric idea explains the use of benchmarks in numerical optimization [6]. Moreover, as we have shown in [6], this heuristic leads to an asymptotically optimal approach.

Acknowledgments

This work was supported in part by the National Science Foundation grants HRD-0734825 and HRD-1242122 (Cyber-ShARE Center of Excellence) and DUE-0926721, and by an award "UTEP and Prudential Actuarial Science Academy and Pipeline Initiative" from Prudential Foundation.

References

[1] Th. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts, 2009.

[2] A. Di Ciaccio, M. Coli, and J. M. Angulo Ibanez (eds.), *Advanced Statistical Methods for the Analysis of Large Data*, Springer Verlag, Berlin, Heidelberg, 2012.

[3] M. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Co., San Francisco, California, 1979.

[4] C. P. Gomes, H. Kautz, A. Sabharwal, and B. Sel, "Satisfiability Solvers", In: F. van Harmelen, V. Lifschitz, and B. Porter, *Handbook of Knowledge Representation*, Elsevier, Amsterdam, 2008, pp. 89–134.

[5] V. Kreinovich, "A review of 'Constructivization of the concepts of ε-entropy and ε-capacity' ", *Mathematical Reviews*, 1979, Vol. 57, No. 1, p. 16, review # 78.

[6] V. Kreinovich and S. A. Starks, "Why benchmarking is an (asymptotically) optimal approach to numerical methods: a geombinatoric proof", *Geombinatorics*, 2004, Vol. 13, No. 3, pp. 131–138.

[7] G. C. Lorentz, *Approximation of Functions*, Holt, Rinehart, and Winston, New York, 1966.

[8] B. Marr, *Big Data: Using SMART Big Data, Analytics and Metrics To Make Better Decisions and Improve Performance*, Wiley, Chichester, West Sussex, UK, 2015.

[9] N. Marz and J. Warren, *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*, Manning Publications, Shelter Island, New York, 2015.

[10] V. Mayer-Schönberger and K. Cukier, *Big Data: A Revolution That Will Transform How We Live, Work, and Think*, Eamon Dolan/Mariner Books, London, UK, 2014.

[11] F. J. Ohlhorst, *Big Data Analytics* John Wiley & Sons, New York, 2012.

[12] C. Papadimitriou, *Computational Complexity*, Addison Welsey, Reading, Massachusetts, 1994.

[13] S. Srinivasa and V. Bhatnagar (eds.), *Big Data Analytics*, Proceedings of the First International Conference on Big Data Analytics BDA'2012, New Delhi, India, December 24–26, 2012, Springer Lecture Notes in Computer Science, Vol. 7678, 2012.