

Leslie Valiant – Innovator in Machine Learning

Over the past 30 years, Leslie Valiant has made fundamental contributions to many aspects of theoretical computer science. His work has opened new frontiers, introduced ingenious new concepts, and presented results of great originality, depth, and beauty. Time and again, Valiant's work has literally defined or transformed the computer science research landscape.

Computational learning theory. Valiant's greatest single contribution may be his paper "A theory of the learnable" (*CACM*, 1984), which laid the foundations of computational learning theory. The field of machine learning had originated in the 1950's. By the early 80's it was a thriving research area, but the field lacked a clear consensus as to its mathematical foundations. In this way, the state of learning theory before Valiant was closely analogous to that of computability before Turing.

The lack of widely agreed upon foundations was not merely a technical issue, but rather a major hindrance for the field as a whole. The importance of clear foundations is that they enable negative as well as positive results: if you can do something, you can simply demonstrate your abilities and avoid worrying about foundational issues, but if you can't, then how are you to know whether this is a personal failing or a genuine impossibility? Only a careful analysis based on clear, widely accepted foundations can answer this question and distinguish between the difficult and the impossible.

Valiant's "probably approximately correct" (PAC) model supplied beautiful foundations for the very concept of learning. Suppose we are faced with a classification problem, such as determining whether incoming e-mail should be filed under junk mail. The unpredictability of the e-mail is modeled by saying it is drawn from some unknown probability distribution. We assume there is a ground truth for the classification problem: in this case, whether the user would consider the e-mail to be spam. However, this ground truth may be quite subtle and difficult to capture via conventional algorithms.

A learning algorithm makes use of a training set of correctly classified e-mail received in the past, and it attempts to devise a hypothesis that can be used to classify future e-mail. The difficulty here is generalization error: even if the hypothesis correctly classifies all the old e-mail, it may still fail for future e-mail because of overfitting, the problem of matching the hypothesis too carefully to inconsequential features of the training set. (For an extreme example, consider the hypothesis that the only possible spam e-mails are the ones explicitly classified as spam in the training set. This explains the training set perfectly, albeit at the cost of violating Occam's razor, but it is useless for classifying future e-mail.)

It is typically impossible to avoid generalization error entirely, but we can try to minimize it, and this is exactly what the PAC model does. The phrase "probably approximately correct" means that with high probability ("probably"), the hypothesis

should have low generalization error (it should be “approximately correct”). The PAC model therefore allows two failure modes: the learning algorithm may on rare occasions be unable to learn from the training data, in which case it could output a terrible hypothesis, and the rest of the time its hypothesis should have good (but not necessarily perfect) accuracy on the data it will see in the future.

In this framework, it is very natural to ask about computational complexity: how large a training set and how much computation are required to produce a good hypothesis? By quantifying the problem in this way, we can seek graded examples that define the limits of learnability: some things simply cannot be learned, some can be learned but only at great cost, and others are easily learned. Valiant’s work thus brought together machine learning and computational complexity in a way that has since led to major advances in both fields. Indeed, PAC learning has developed into a vibrant research area which has had enormous influence on machine learning, artificial intelligence, and many areas of computing practice such as natural language processing, handwriting recognition, and computer vision.

In hindsight, PAC learning seems almost obvious, and part of the beauty of Valiant’s work is how perfectly it reflects our intuitions about the learning process. In that respect, it is very much like Turing’s definition of the Turing machine. Turing was certainly not the first person to think about computation, and he was not even the first person to try to formalize it mathematically. However, his model was so simple and compelling that it immediately captured the imagination of the community and led to widespread agreement that this was indeed the right approach. The same can be said of Valiant’s work on PAC learning: it has become an essential framework for the theory of learning.

Complexity of enumeration. In the early 1970’s, computational complexity generally dealt with the difficulty of decision problems, such as whether a graph has a perfect matching or whether a traveling salesman can find a route of at most a certain length. This difficulty was characterized by complexity classes, such as P (tractable problems) and NP (problems for which a solution can easily be checked once it has been produced, but perhaps not easily found in the first place). Of course, many important practical problems are not simply decision problems: instead of asking whether there is a route of at most 1000 miles, one could ask for the length of the shortest possible route. However, these more general problems can often be reduced to a sequence of decision problems, for example by using binary search to narrow in on the length of the shortest route.

One of Valiant’s most noteworthy discoveries is that counting problems are much more subtle than previous experience suggested. A counting problem asks for the number of some combinatorial objects: for example, how many perfect matchings are there in a graph? Now we are not just asking the decision problem of whether that number is positive, but also how large it is. If the decision problem is difficult, then the counting problem must be as well, but Valiant’s surprising realization was that the converse fails. In his paper “The complexity of computing the permanent”

(*Theoretical Computer Science*, 1979), he showed that although there is an efficient algorithm to tell whether a graph has a perfect matching, there is no efficient algorithm to count perfect matchings (unless $P = NP$), and in fact counting perfect matchings is as hard as any counting problem. This came as a shock to the computational complexity community, which had grown accustomed to the idea that decision problems would easily capture the key features of a problem. Instead, Valiant extended the theory of complexity classes to include new counting classes such as $\#P$.

If counting problems were limited to esoteric mathematical problems, Valiant's theory would still have been conceptually fascinating but it would have had limited impact. However, it turns out that these problems pervade much of computer science. For example, estimating any probability amounts to an approximate counting problem, and random sampling is closely related. Approximate counting and sampling are both important goals in their own right (for example, in statistics) and valuable tools for solving other problems; entire subfields of computer science, such as Markov Chain Monte Carlo algorithms, are by now devoted to these topics. The ideas Valiant introduced have grown into a thriving field, which has united computer science, probability and statistics, combinatorics, and even aspects of statistical physics.

It is noteworthy that so many of Valiant's papers have been singly authored, but he has also had several fruitful collaborations. For example, he and Vijay Vazirani wrote the influential paper "NP is as easy as detecting single solutions" (*Theoretical Computer Science*, 1986). Before this paper, many researchers believed that the hardest search problems were those with many solutions (sparsely embedded among far more non-solutions), because the multiplicity of solutions could confuse search algorithms and keep them from being able to narrow in on a single solution. Valiant and Vazirani gave a dramatic and beautiful demonstration that this idea was completely wrong, by showing how a good algorithm for finding unique solutions could be used to solve any search problem. This result was not only of great interest in its own right, but also a critical technical tool for many later discoveries.

Algebraic computation. Another key contribution to computational complexity was Valiant's theory of algebraic computation, in which he established a framework for understanding which algebraic formulas can be evaluated efficiently. This theory is in many ways analogous to the more traditional study of Boolean functions, but it has a rather different flavor as well as deep connections with algebraic geometry and other mathematical fields that have not often been applied to computer science.

In his paper "Completeness classes in algebra" (Proc. STOC, 1979), Valiant characterized the difficulty of algebraic computation in terms of two fundamental and closely related functions from linear algebra, namely the determinant and the permanent. The determinant can easily be computed, while the permanent is much harder, despite the superficial similarity between the two, and there is a technical sense in which the permanent is one of the hardest functions to compute. This work

also ties beautifully into the complexity of enumeration, because the permanent also plays a fundamental role in #P.

Valiant's papers played a pioneering role in bringing algebraic techniques into the toolbox of theoretical computer science. He thus set the stage for some of the most exciting subsequent developments in computational complexity, such as the development of interactive proofs for problems beyond NP (in which the permanent function was pivotal).

In addition to laying the foundations of the theory of algebraic computation, Valiant's work has also had striking consequences in other areas, such as the construction of networks with extremal properties. This problem arose in the study of the fast Fourier transform (FFT), a fundamental algebraic algorithm for data analysis that can perform a full spectral decomposition on n data points in about $n \log n$ operations. Ever since the introduction of the FFT algorithm by Cooley and Tukey, people have wondered whether the $n \log n$ running time could be reduced to n , which would be best possible (running in linear time). Given a linear-time FFT algorithm, one could extract from it a network known as a superconcentrator, which is a special type of highly connected network. Strassen, and Aho, Hopcroft, and Ullman posed the question whether no linear-sized superconcentrator could exist, which would have ruled out a linear-time FFT. However, in his breakthrough paper "Graph-theoretic properties in computational complexity" (*Journal of Computer and System Sciences*, 1976), Valiant showed how to construct linear-sized super-concentrators. Unfortunately, his methods did not actually lead to a linear-time FFT, but in fact they had arguably even more impact. They helped initiate the expander revolution in theoretical computer science, in which superconcentrators and more general expander graphs were used to build excellent error-correcting codes, derandomize randomized algorithms, and develop distributed algorithms for sorting, searching, and routing, among many other applications.

Parallel and distributed computing. In addition to computational learning theory and computational complexity, a third broad area in which Valiant has made important contributions is the theory of parallel and distributed computing. His results here range from simple, but powerful and elegant, insights to reexamining the very foundations. An example of a simple insight is his parallel routing scheme, described in the paper "A scheme for fast parallel communication" (*SIAM J. Computing*, 1982). Imagine that a variety of processors connected by some network are attempting to exchange data simultaneously. Using simple greedy schemes to route the data can often lead to congestion — too many data paths may end up using the same link in the network at once. On the other hand, complex schemes, even if conceived, posed the challenge that they would be hard to implement in hardware. Valiant discovered a brilliant and simple randomized solution to the problem. He suggested that every packet, instead of going directly to its destination, should simply pick *one* intermediate point, *randomly* among all the nodes in the network, and then works its way greedily along some shortest path from its origin to the intermediate point and then from there to its destination. This scheme, only minimally more complex than the obvious routing strategy, diffuses any potential

congestion (provably, with high probability).

Valiant's main contribution to parallel computing is the introduction of the "bulk synchronous parallel" (BSP) model of computing. One of the main articles introducing this model is his paper "A bridging model for parallel computation" (*CACM*, 1990). This paper is a must read for both technical and pedagogical reasons. It lays out the case that a model of parallel computing should attempt to bridge software and hardware. Specifically the model should capture parallel computing hardware by a small collection of numerical parameters. Similarly, parallel compilers should only need to know these few parameters from the model when compiling programs. Such a model should be judged by how well it can predict the actual running times of parallel algorithms.

In contrast to the case of sequential computing, where von Neumann's model easily satisfied these requirements (at least to a first approximation), coming up with a similar model in parallel computing has been hard. Following this articulation of the challenge, Valiant proposed the BSP model as a candidate solution. In this model, a computation is bulk synchronized in the sense that each processor may perform many steps independently in between global synchronization operations and exchanges of information. A parallel computer is further specified by parameters for the number of processors as well as for the latency and throughput of the interconnect.

Valiant showed how this model distinguishes the performance of different algorithms, recommending some over the others for certain choices of parameters, while switching to other algorithms when parameters change. Valiant argued that software designed with these parameters in mind is likely to run in the predicted time. Finally, he argued that the model remains rich enough to allow a variety of hardware implementations, in particular allowing the use of many of sophisticated hashing/routing/interconnection schemes for interprocessor communication. The debate on the right model for parallel computing remains unresolved to this day, with several competing suggestions. It is unlikely that the debate will ever be completely resolved by theoretical arguments, without building large multicore machines and software. However, Valiant's work in this setting shows us how far we can reason about the problem purely based on scientific principles.

Conclusion. Rarely does one see such a striking combination of depth and breadth as in Valiant's work. He is truly a heroic figure in theoretical computer science and a role model for his courage and creativity in addressing some of the deepest unsolved problems in science.

###

