Attribute-Efficient Evolvability of Linear Functions

Extended Abstract

Elaine Angelino
Harvard University
elaine@eecs.harvard.edu

Varun Kanade
UC Berkeley
vkanade@eecs.berkeley.edu

Darwin's theory of evolution through natural selection has been a cornerstone of biology for over a century and a half. Natural selection acts on *phenotypes* that are a result of molecular activities at the level of cells; the molecular activities themselves are encoded in the DNA sequence, or *genotype*.[1] Every cell is a computing machine, sensing its environment and producing appropriate responses. Abstractly, cellular activity can be viewed as computing a function; the input is an "internal representation" of the environment using proteins and other molecules, and the output is the activation or repression of proteins expression. Yet, a quantitative theory of the complexity of such functions that could arise through Darwinian mechanisms has remained virtually unexplored. Here, complexity is viewed through the lens of theoretical computer science, *e.g.*, the size of the circuit required to compute the function (cf. Arora and Barak [2009], Papadimitriou [2003]).

To address this question, Valiant [2009] introduced a computational model of evolution.[2] In this model, an organism is an entity that computes a function of its environment. For simplicity, each organism computes only one function, though in reality there may be thousands if not more. For example, consider an enzyme that metabolizes sugars. The corresponding function computed by the cell could determine how much of this enzyme to produce, as function of the concentrations of various sugar molecules present in the environment. There is a (possibly hypothetical) *ideal function* indicating the best behavior in every possible environment, *e.g.*, the optimal amount of enzyme to produce, not necessarily to maximize the yield of its products but rather the overall performance of the organism, including tradeoffs between some action and its energetic costs. The performance of the organism is measured by how close the function it computes is to the ideal. An organism produces a set of offspring, that may have mutations that alter the function computed. The performance measure acting on a population of mutants forms the basis of natural selection.[3] The resources allowed are the most generous while remaining feasible; the mutation mechanism may be any efficient randomized Turing machine,[4] and the function represented by the organism may be arbitrary as long as it is computable by an efficient Turing machine.

Formulated this way, the question of evolvability can be asked in the language of computational learning theory. For what classes of ideal functions, C, can one expect to find an evolutionary mechanism that gets arbitrarily close to the ideal, within feasible computational resources? A function class captures a notion of complexity, say for example,

$$f(x_1, \ldots, x_n) = 2x_1 + 3.7x_4 - 0.7x_9 + 6x_{12} + 1.8,$$

is a linear function, and one could consider the class of all such linear functions. In the toy example of the enzyme that metabolizes sugars, the various x_i could be different molecules in the environment, where those appearing on the right hand side of the equation correspond to sugars. The different coefficients, if tuned by natural selection, could reflect differences among the sugars with respect to their benefits versus costs to the cell, and the positive additive constant corresponds to the baseline level of enzyme. A more complex class is that of quadratic functions, where a function takes the form $f(x_1, \ldots, x_n) = 3.2x_1^2 - 7.3x_1x_6 + 18x_3 + 8.1$. The point here is that the highest degree of any term appearing the expression is 2. One expects that the more *complex* the ideal function, the harder it is for evolution to succeed in approximating it. Here, the notion of approximation is the following: Suppose there is a distribution D over inputs $(x_1, \ldots, x_n)$, the ideal function is f and the organism computes some other function h. Then, the *loss* of the organism is

[1]There are epigenetic factors at play, but the general principle expounded in this document applies to those mechanisms as well.
[2]Also, see Valiant [2013] for a very accessible treatment of computational learning theory in general, and this model in particular.
[3]Recombination may increase the speed of evolution, but it is understood that in Valiant's model, it does not affect the complexity that can arise in functions.
[4]A Turing machine is a mathematical model on which all modern computers are based; the widely believed Church-Turing hypothesis states that *any* feasible computation in nature can be simulated by a Turing machine.

$\mathbb{E}_{x \sim D}[(f(x) - h(x))^2]$, the expected squared difference between the organism's function and the ideal.[5] An evolutionary mechanism is successful if the loss becomes *close* to 0 (which is the best possible) in a relatively small number of generations.

The reason for allowing mutations and representations of functions to be quite general in Valiant's model is primarily the lack of our current understanding of how mutations occur in nature and also the relationship betewen changes in genotype to changes in phenotype.[6] However, a consequence of this generality has been that *feasible* evolutionary mechanisms in this model can be encodings of very sophisticated algorithms. These computations have to be performed by "chemical computers" in cells, which could be severely restricted in their computational capabilities. Thus, we are interested in understanding what evolutionary mechanisms could succeed when they have limited computational power at their disposal. We illustrate our point with a particular kind of biological circuit: transcription networks.

The view presented here is an exaggerated simplification of the actual transcription process, the goal being to focus on the complexity of the circuit representation. A transcription network consists of interacting genes and proteins that are involved in the production of new protein. Genes are *transcribed* to produce mRNA, which is then *translated* into sequences of amino acids that ultimately fold into proteins.[7] In a transcription network, a gene's transcription may be regulated by a set of proteins called *transcription factors.* These factors may increase or decrease a gene's transcription by physically binding to regions of DNA that are typically close to the gene. In natural systems, only a small number of transcription factors regulate any single gene, and so transcription networks are sparsely connected.

The number of transcription factors varies from hundreds in a bacterium to thousands in a human cell. Some transcription factors are always present in the cell and can be thought of as representing a *snapshot* of the environment (cf. Alon [2006]). For example, the presence of sugar molecules in the environment may cause specific transcription factors to be *activated*, enabling them to regulate the production of other proteins. One of these proteins could be an *end-product*, such as an enzyme that catalyzes a metabolic reaction involving the sugar. Alternatively, the transcription factor could regulate another transcription factor that itself regulates other genes – we view this as intermediate computation – and may participate in further "computation" to produce the desired end-result. While transcription networks may include cycles (loops), here for simplicity we focus on systems that are directed acyclic graphs, and the resulting computation can be viewed as a circuit. These circuits are by necessity shallow due to a temporal constraint, that the time required for sufficient quantities of protein to be produced is of the same order of magnitude as cell-division time.[8]

In our work, we look at linear functions, *e.g.*, $f(x) = 3x_1 + 7x_3 - 5x_8$. A model to compute linear functions is an arithmetic circuit with one addition gate and several input wires. Each input wire can have a positive or negative weight associated with it. Thus, the linear function $f(x) = 3x_1 + 7x_3 - 5x_8$, can be computed by a simple circuit that has one addition gate and three input wires having weight 3, 7 and -5 respectively. We now want to know that if the *ideal function* is sparse, *i.e.*, most of the weights are zero, does there exist an evolutionary mechanism that would be successful, while having the property that the "cellular circuit" at each stage of evolution is itself a sparse one? Under rather mild assumptions on the distribution of inputs, we show that this is indeed the case. A further interesting property of this mechanism is that the main resource required, the number of generations, depends only on the number of relevant variables in the *ideal function*, and not on the total number of variables. The population required for evolution to succeed depends on the total number of variables, not just the relevant ones.

[5]The closer h is to f, the smaller the loss. Here, we are simply assuming that the performance depends on the approximation in terms of squared error. It is possible to use other loss functions and indeed a very interesting (and largely unresolved) question is how robust these evolutionary mechanisms are to changes in the loss function.

[6]It also has the added advantage that this allows one to talk about computationally feasible Darwinian evolution of any type, not just restricted to that observed on earth.

[7]In reality, this is a dynamical system where the rates of production are important. Note that this process need not be linear: a gene (mRNA transcript) can be transcribed (translated) multiple times, not only in series but also in parallel fashion.

[8]Other kinds of networks, such as signaling networks, operate by changing the shapes of proteins. The fact that these transformations are rapid may allow for much larger depth. Note that fast conformational changes govern how transcription factors directly process information from the environment in order to regulate gene expression. In our example, a sugar molecule binds to a transcription factor and changes its shape in a way that alters its ability to bind to DNA.

Of course, a linear function may be a very restrictive model of what actually goes on in the cell. As part of future work, we think it is worth investigating whether other functions such as the sigmoid operating on linear functions can also be evolved, assuming that the number of relevant variables is much smaller than the total number of variables.

Bibliographic Note

From the point of view of computer science, there have been very interesting developments in understanding the power of Valiant's model. We have largely left these out of this document for reasons of brevity and accessibility. Apart from Valiant's original paper [Valiant 2009], the interested reader is referred in particular to the work of Vitaly Feldman and Paul Valiant [Feldman 2008; 2009; 2011; Valiant 2012]. The second author's thesis contains an exposition of some of these results and other work [Kanade 2012].

In the case of biological literature, our omissions are even greater. The companion paper to this note [Angelino and Kanade 2014] contains a more complete bibliography. Our main purpose here is to introduce the computational framework in a language that is more widely understood.

Acknowledgments

We would like to thank Leslie Valiant for helpful discussions and comments on an early version of this paper. We are grateful to Frank Solomon for discussing biological aspects related to this work. EA was supported by a grant from the National Library of Medicine (4R01LM010213-04). VK is supported by a Simons Postdoctoral Fellowship.

REFERENCES

Alon, U. 2006. *An Introduction to Systems Biology: Design Principles of Biological Circuits.* Chapman and Hall/CRC, Boca Raton, FL.

Angelino, E. and Kanade, V. 2014. Attribute-efficient evolvability of linear functions. In *Innovations in Theoretical Computer Science*. To appear.

Arora, S. and Barak, B. 2009. *Computational complexity: a modern approach.* Cambridge University Press.

Feldman, V. 2008. Evolution from learning algorithms. In *Proceedings of the Symposium on the Theory of Computation (STOC).*

Feldman, V. 2009. Robustness of evolvability. In *Proceedings of the Conference on Learning Theory (COLT).*

Feldman, V. 2011. Distribution-independent evolution of linear threshold functions. In *Proceedings of the Conference on Learning Theory (COLT).*

Kanade, V. 2012. Computational questions in evolution. Ph.D. thesis, Harvard University.

Papadimitriou, C. H. 2003. *Computational complexity.* John Wiley and Sons Ltd.

Valiant, L. 2013. *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World.* Basic Books, New York, NY.

Valiant, L. G. 2009. Evolvability. *Journal of the ACM 56,* 1, 3:1–3:21.

Valiant, P. 2012. Evolvability of real-valued functions. In *Proceedings of Innovations in Theoretical Computer Science (ITCS).*