A Tale of Three Bio-inspired Computational Approaches

J. David Schaffer, Binghamton Univ.

ABSTRACT:

I will provide a high level walk-through for three computational approaches derived from Nature. First, evolutionary computation implements what we may call the "mother of all adaptive processes." Some variants on the basic algorithms will be sketched and some lessons I have gleaned from three decades of working with EC will be covered. Then neural networks, computational approaches that have long been studied as possible ways to make "thinking machines", an old dream of man's, and based upon the only known existing example of intelligence. Then, a little overview of attempts to combine these two approaches that some hope will allow us to evolve machines we could never hand-craft. Finally, I will touch on artificial immune systems, Nature's highly sophisticated defense mechanism, that has emerged in two major stages, the innate and the adaptive immune systems. This technology is finding applications in the cyber security world.

Introduction

Many of us have been impressed by the incredible performance of Nature's creations. This has led to many attempts to abstract the essences of how they work and apply these processes within our computing machines. We dream of performances out of reach for today's technologies. I will illustrate three such approaches: evolutionary algorithms, neural networks, and artificial immune systems. Space and time limitations necessarily restrict this exposition to a few simple illustrations.

Evolutionary Computation (EC)

I like to think of EC as the "mother of all adaptive processes" primarily because the basic paradigm is so simple. It's just generate-and-test. Perhaps a short story will help make the point. In 1969, Don Waterman built one of the very early poker-playing learning systems for his PhD at Stanford. It was a rule-based system and made bets in a simulated poker game. Whenever it made a wrong bet, another program, called the oracle and presumed to possess perfect knowledge of poker strategy, informed the learner: that it had made an error, what the right bet was, and what game features were needed to make that decision. With this set of information, the learner could craft an improved rule for this situation. The difficulty with this paradigm was that the learner could, at best, asymptotically approach the performance of the oracle. In 1980, Steve Smith at Pittsburgh, built another poker learning system based on EC. As a test, he played his system against (his approximation of) Waterman's oracle. The EC system, requiring nothing more than the results from games played, soon found that the oracle could be bluffed, and went on to large winnings by exploiting this weakness. The point is that simple trial-and-error learning imposes a minimum of restrictions on what can be achieved.

EC has two essential processes: survival of the fittest selection, and inheritance with variation. Its distinguishing characteristic with respect to other stochastic learning approaches is its use of a population of alternative solutions throughout its learning. One property deriving from the use of a population might be called soft selection: an offspring need not be strictly better than its progenerator in order to survive, it need only be better than some individual in the previous generation. Non-population algorithms that want this property must provide an explicit algorithm for it (e.g. the cooling schedule in simulated annealing). A within-EC distinction may also be made between algorithms that produce offspring by some mutation

Machine Intelligence and Bio-inspired Computation: Theory and Applications VIII,
edited by Misty Blowers, Jonathan Williams, Proc. of SPIE Vol. 9119, 911902
 · doi: 10.1117/12.2058210

process acting on a single parent (asexual reproduction) and algorithms that mate two parents, performing some form of genetic recombination (sexual reproduction). Among the ECs using genetic recombination, there are options that can arise with different chromosome representations, but an emerging insight seems to be that a property of a crossover operator that is very important is "respect": the preservation of alleles common to both parents [10]. This gives rise to a property we have called convergence constrained variation (CCV) [1], as a population converges at some chromosome loci, offspring will no longer vary on those dimensions. Figure 1 illustrates a simple binary string representation for numerical variables and how mutation and crossover can be applied to produce offspring. One may realize that virtually any information type can be mapped to and from bit strings, but also that the representations amenable to mutation and crossover operations are not limited to bit strings, and that for non-bit string representations, the options for creative mutation and crossover operations are also expanded. One example application of this approach is the evolution of digital power-of-two (multiplierless) filters [14].

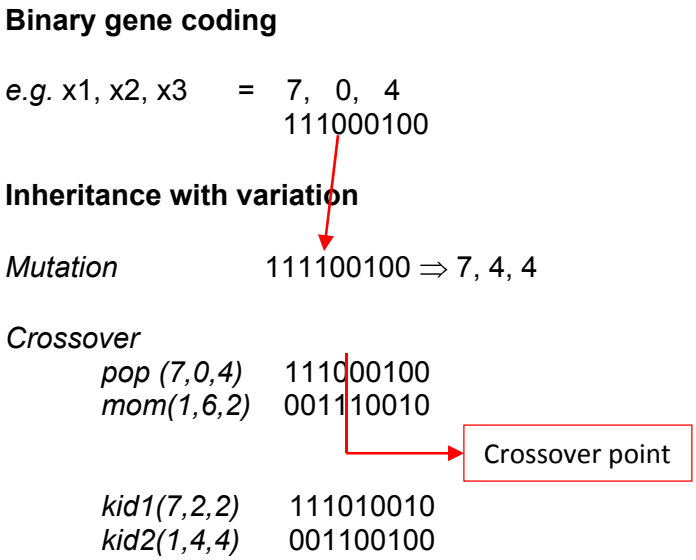

Figure 1. Example of bit encoding of numerical genes with mutation and one-point crossover

It is important to keep in mind, when considering the advantages and weaknesses of any algorithm, the No Free Lunch theorem of Wolpert and Macready [16]. The implication of this is that there are no universally superior search algorithms. Given this fact, theorists have become interested in defining classes of search problem complexities and what sorts of algorithm are best suited to them. While no general theory seems yet to have emerged, one class of problems for exploring these issues, proposed by Kaufmann [5] is NK Landscapes. These "black-box optimization tasks[1]" are characterized by two parameters, N the length of bit strings that define the search space, and K a measure of the epistasis, or complexity of the response function. By varying K, one can define search problems that are very simple (K=0) all the way to utterly structureless (K = N-1). Mathias et al. [7] have shown that the niche for EC may reside in the region of large N (> 70) and limited K (1-10). One should be aware that K=5 to 10 represents very considerable complexity.

[1] Given an unknown function in a black box, that takes bit strings of length N as inputs and reports a function value for each string, the challenge is to find the input that yields the minimum (or maximum) value.

Neural Networks (NNs)

The history of NNs may be considered to have started with the work of McCulloch and Pitts in 1943 [8]. They showed that any Boolean function could be implemented by a suitably designed network of simple switches. The excitement may be attributed to the linking of Boole's logic to "reasoning" and the apparently simple nature of binary switches. The telephone companies of that time were already building very large networks of such switches. A difficulty was that there were no design rules for building networks for any chosen Boolean function. Later came the perceptron [11] that also brought with it a procedure for learning from examples and a comforting proof that it was guaranteed to converge to the optimal network in a finite number of steps. This seemed truly exciting, but the excitement abated when Minsky and Papert [9] showed the perceptron was limited to linearly separable tasks. There followed a period sometimes called "NN winter," that finally thawed with the emergence of the multi-layered perceptron (MLP) and the improved learning procedure, backpropagation [13]. The renewed NN efforts soon revealed that the MLP has no dynamic memory for transient patterns, and so time series analyses remained a challenge. In the late 1990s there emerged a new paradigm, spiking NNs [3] (SNNs). These models implemented essentially asynchronous computing and also were shown to possess the main computational properties of the previous two generations of NNs.

The challenge of designing SNNs still remains. Little is known about how many neurons and how they should be connected to perform complex functions. However, efforts to apply EC to NN problems had been ongoing for some time [15]. In this vein, recently Roy et al. [12] have provided one approach to evolving a class of SNNs that can perform spatio-temporal signal discrimination. The result of this learning is a SNN that detects embedded temporal patterns in time series signals.

Artificial Immune Systems (AISs)

Stephanie Forrest has stated that "Biology IS the science of security." [2]. She was referring to the fact that since the very beginning of life, survival has involved adversarial interactions, leading to a rich arsenal of weapons, defenses, deception, and mimicry. With the emergence of more complex multi-cellular phenotypes, some defense against the rapid evolvability of simpler phenotypes was needed. This surely was the driver for the emergence of immune systems. Forrest was among the first [4] to explore the possibilities of artificial immune system algorithms and their application to cybersecurity.

Considering natural immune systems, we humans have two: the more ancient innate immune system (~500M years old) and the more recent (~300M years old) adaptive immune system. It is interesting that the human immune systems comprise a number of cells on the same order as the number of cells in the brain (~100B). Yet the immune system cells are mobile and therefore must coordinate their behaviors with chemical messages (cytokines). Unlike the brain, the immune system must respond to a pathogen attack that may occur at any place in the body. It is therefore not surprising that a system of this complexity can also exhibit failure modes such as auto-immune disorders and sepsis. Nevertheless, the immune system, like most of Nature's highly evolved systems, exhibits phenomenal performance, and so attempts to build algorithms that abstract some of its essential mechanisms have emerged.

We mention just two such algorithms. Negative selection algorithms aim to produce pattern detectors (themselves also being patterns) that can identify "foreign" patterns and at the same time avoid the many patterns that characterize the self (legitimate patterns). Such a process takes place in a protected environment (i.e. not on the job). Given a set of known self-patterns, randomly generate candidate detectors and eliminate those that match any of them. Another algorithm, called clonal selection, or affinity

maturation, happens “on the job.” Given a set of pathogen patterns, and some detectors with limited affinity for them, produce random variations of these detectors in such a way that the number of clones for each current detector is proportional to its affinity for the invaders, and the variations (mutations) are inversely proportional to the affinity. The result is patterns of increasing affinity for the targets. Naturally, among the approaches explored in the AIS domain, EC has also found a role to play [4,6].

Conclusion

This albeit brief overview of three bio-inspired paradigms, seems suggestive of an attractive research direction. Both NNs and AISs represent adaptive or learning systems that have been evolved. My own chosen research direction and one I recommend to others of like mind, is to explore creative ways to apply EC to the evolution of learning machines.

References

1. Eshelman, L.J., Mathias, K.E. and Schaffer, J.D., “Convergence Controlled Variation,” Foundations of Genetic Algorithms, pp. 203-224, 1996.

2. Forrest, S., Ulam Lectures, Santa Fe Institute, 2013. http://www.youtube.com/watch?v=dAAFKTtVMa8&list=PLZlVBTf7N6GqB3SVb5piIETIArIUgeLXF Last visited 2014-04-14.

3. Gerstner, W., and Kistler, W. M. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

4. Hightower, R., Forrest, S., and Perelson, A., “The Evolution of Secondary Organization in Immune System Gene Libraries,” Proc. Euro. Alife Conf., 1993.

5. Kaufman, S., *Origins of Order: Self-Organization and Selection in Evolution*, Oxford University Press, 1993.

6. Kim, J. and Bentley, P., “Immune Memory and Gene Library Evolution in the Dynamic Clonal Selection Algorithm,” Genetic Programming and Evolvable Machines, Vol. 5, Issue 4, pp 316-391, Dev 2004.

7. Mathias, K.E., Eshelman, L.J. and Schaffer, J.D., “Niches in the NK Landscapes,” Foundations of Genetic Algorithms 6, Morgan Kaufmann, San Francisco, CA, 2001.

8. McCulloch, W., and Pitts, W., “A logical calculus of the ideas immanent in nervous activity,” Bulletin of Mathematical Biophysics 7, pp. 115-133, 1943.

9. Minsky, M., and Papert, S., Perceptrons. MIT press, 1969.

10. Radcliffe, N. J., “Genetic set recombination,” Foundations of Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, pp 203-219, 1992.

11. Rosenblatt, F., “The perceptron: a probabilistic model for information storage and organization in the brain,” Psychological review 65, 6, 386, 1958.

12. Roy, A., Schaffer, J. D., and Laramee, C. B. Evolving spike neural network sensors to characterize the alcoholic brain using visually evoked response potential," Complex Adaptive Systems, Baltimore, MD, 2013.

13. Rumelhart, D. E., Hinton, G.E., and Williams, R.. "Learning representations by backpropagating errors," Nature 323, 533-536, 1986.

14. Schaffer, J.D., and Eshelman, L.J., "Designing Multiplierless Digital Filters Using Genetic Algorithms," Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, pp. 439-444, 1993.

15. Whitley, D., Schaffer, J.D. (editors), *COGANN-92 Combinations of Genetic Algorithms and Neural Networks*, IEEE Computer Society Press, Los Alamitos, CA, 1992.

16. Wolpert, D.H., Macready, W.G. (1997), "No Free Lunch Theorems for Optimization," *IEEE Transactions on Evolutionary Computation* **1**, 67.