The evolution of learning systems: to Bayes or not to be

Nestor Caticha[*] and Juan Pablo Neirotti[†]

[*]*Instituto de Física Universidade de São Paulo, São Paulo, Brazil*
[†]*NCRG Aston University ,Birmingham United Kingdom*

Abstract. Bayesian algorithms pose a limit to the performance learning algorithms can achieve. Natural selection should guide the evolution of information processing systems towards those limits. What can we learn from this evolution and what properties do the intermediate stages have? While this question is too general to permit any answer, progress can be made by restricting the class of information processing systems under study. We present analytical and numerical results for the evolution of on-line algorithms for learning from examples for neural network classifiers, which might include or not a hidden layer. The analytical results are obtained by solving a variational problem to determine the learning algorithm that leads to maximum generalization ability. Simulations using evolutionary programming, for programs that implement learning algorithms, confirm and expand the results.

The principal result is not just that the evolution is towards a Bayesian limit. Indeed it is essentially reached. In addition we find that evolution is driven by the discovery of useful structures or combinations of variables and operators. In different runs the temporal order of the discovery of such combinations is unique. The main result is that combinations that signal the surprise brought by an example arise always before combinations that serve to gauge the performance of the learning algorithm. This latter structures can be used to implement annealing schedules. The temporal ordering can be understood analytically as well by doing the functional optimization in restricted functional spaces. We also show that there is data suggesting that the appearance of these traits also follows the same temporal ordering in biological systems.

Keywords:
PACS:

INTRODUCTION

Evolutionary pressures arise from a wide variety of sources. We will look into the consequences that different capabilities of information processing may have in the fi tness, survival and evolution of information processing systems (IPS). By the latter we would like to mean natural organisms but will settle for the analysis of artifi cial IPS and then rather simple ones. Even in the restricted theater of computer simulations, evolution history cannot be retraced if there is the slightest of changes in initial conditions. The fi rst aim of statistical mechanics approaches to evolution is identifying reproducible features.

We will consider the evolution of certain classes of neural networks (NN) classifi ers that may learn from examples. The correct classifi cation of an example is determined by the environment represented itself by a classifi er. The NN may evolve by changes in its architecture, by changes of the learning algorithm it uses to learn, or both. Fitness will be given by some measure of the effi ciency such as the generalization ability, the probability of correctly classifying a new input. The input to the NN are N-dimensional vectors that can be thought of as representing sensorial data, the classification into one of two possible categories represent the action taken. We will discuss analytic and simulation results obtained from evolutionary programming. The main message is that in this simple scenario, Bayesian limits are essentially reached. Moreover, during evolution certain intermediate nonoptimal architectures and learning algorithms are visited in a quite systematic way. We have identified a temporal ordering in the appearance of features of the learning algorithm which may find a parallel in biological systems. We now review some results concerning optimal learning algorithms in a class of simple NN obtained from a variational method and optimization under restrictions. We then discuss their relation to Bayesian bounds and finally present results of a simulation of genetic programming [1] where the learning algorithms are represented by programs. By selection, offspring programs enventually evolve to algorithms that saturate Bayesian bounds.

OPTIMAL MODULATION

We consider a boolean perceptron learning from a set of examples $\{\mathbf{S}_\mu, \sigma_{B\mu}\}$, where the $\mathbf{S}_\mu$ are N-dimensional vectors drawn independently from a distribution $P(\mathbf{S})$ and the environment is represented by a function $\sigma_{B\mu} = T_E(\mathbf{S}_\mu)$. Here we only consider linearly separable boolean rules, so that $\sigma_{B\mu} = sign(\mathbf{B}.\mathbf{S}_\mu)$ for some unknown quenched vector $\mathbf{B}$ which represents the environment. The NN share the same architecture with $\sigma_\mu = sign(\mathbf{J}.\mathbf{S}_\mu)$. We can study both analytically and numerically more complex architectures. Elsewhere we will study evolving architectures and the evolution of the complexity of architecture. We consider as natural the on-line learning scenario where the NN receives the examples sequentially and errors are not fatal. Knowledge of $\{\mathbf{S}_\mu, \sigma_{B\mu}\}$ can be used in learning by updating the weights $\mathbf{J}_\mu \rightarrow \mathbf{J}_{\mu+1}$:

$$\mathbf{J}_{\mu+1} = \mathbf{J}_\mu + f(\mathbf{S}_\mu, \sigma_{B\mu}, ...) \tag{1}$$

The f function carries the information in the sensorial data, the correct classification and any other possible statistic that maybe available. Physicists have considered a version of Hebbian learning (e.g. [7]), inspired in biology, where f is given by $f_{Hebb} = \frac{1}{N}\mathbf{S}_\mu \sigma_{B\mu}$. We consider the simplest family of learning algorithms by extending to the modulated Hebbian learning, where

$$f = F\frac{1}{N}\mathbf{S}_\mu \sigma_{B\mu} \tag{2}$$

and F, the modulation function, is an unknown function of an, up to now, unknown set of variables. These variables are only restricted by the present and past available information and the capacity of the NN to remember it. For the simple perceptron case, such memory is quite limited, but in the general adaptive architecture case, modules dedicated to compute useful statistics can appear.

The final ingredient is the fitness function. We think natural to consider the generalization error. Although this is not available in constructing the NN, the environment can surely deem them fit or not according to it. The generalization error will be a functional of the modulation function:

$$e_g\{F\} = \int \Theta(-\sigma_B \sigma) dP(\mathbf{S}) \tag{3}$$

We now study the dynamics of e_g as a function of the number of examples. This can be done by considering the simplifying *thermodynamic limit* (TL) where the number of examples μ and $N \to \infty$, with $t = \mu/N$ finite.

We now make an unessential restriction to the case where the distribution of examples is uniform, so that eq. 3 is easy to calculate in the TL. Doing the integral of eq. 3 shows that the relevant *order parameter* is $\rho = \mathbf{B}.\mathbf{J}/J$, the overlap between the (normalized) environment weight function and that of the NN, of length J, then $e_g = \frac{1}{\pi}\cos^{-1}\rho$ Eq. 1 with a general modulation function (eq. 2) multiplied by $\mathbf{B}$ gives the variation of the overlap due to the addition of one example:

$$\rho_{\mu+1} = \rho_\mu + \frac{1}{NJ_\mu}\left[(b_\mu - \rho_\mu h_\mu)\sigma_{B\mu} F_\mu - \frac{\rho_\mu F_\mu^2}{2J_\mu}\right], \tag{4}$$

where $b_\mu = \mathbf{B}.\mathbf{S}_\mu$ and $h_\mu = \mathbf{J}_\mu.\mathbf{S}_\mu/J_\mu$.

It can be shown [4] that while ρ is a self-averaging quantity in the TL, the variation $\Delta\rho$ is not. Averaging over the μ^{th} example and taking the TL, with $dt = 1/N$, we get

$$\frac{d\rho}{dt} = \frac{1}{J}\int dhdbP(h,b)\left[(b-\rho h)\sigma_B(b)F - \frac{\rho F^2}{2J}\right]. \tag{5}$$

Here $\sigma_B(b) = sign(b)$ and $P(h,b) = \frac{1}{2\pi}exp(-(b^2+h^2-2bh\rho)/(2(1-\rho^2))$, h and b are gaussian variables with unit variance and correlation ρ.

We now ask the fundamental question: which modulation function F leads to the maximum gain of information per example? The answer obviously depends on the variables upon which F depends. Call $\mathbf{H}$ the hidden variables not available to F, which depends solely on the set $\mathbf{V}$ of visible variables. Then the optimal modulation function is obtained from

$$\frac{\delta}{\delta F}\frac{de_g}{dt} = 0, \tag{6}$$

This is the same as $\frac{\delta}{\delta F}\frac{d\rho}{dt} = 0$, which results in the posterior average over the nuisance variables $\mathbf{H}$

$$\bar{F}(\mathbf{V}) = \left\langle J\sigma_B(\frac{b}{\rho} - h)\right\rangle_{\mathbf{H}|\mathbf{V}}. \tag{7}$$

Fig. 1 (left) shows the modulation function as a function of the field $\sigma_B h$ for different stages of the learning process. Here the available information is $\mathbf{V} = \{\sigma_B, \mathbf{S}, \sigma_J, J, \rho\}$ while $\mathbf{H} = \{b\}$. In this particular setting we cannot do better than this and it even may seem too much to include ρ in the group, since in general the generalization error is not available. However, for the optimal modulation function the available J obeys a differential equation that is exactly the same as ρ. So starting from $\mathbf{J} = 0$ leads to $J = \rho$. This is not practical in an application but good online estimators of ρ can be found, [11]

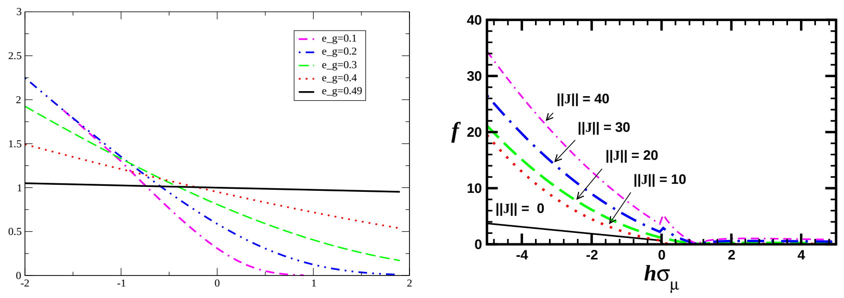

FIGURE 1. Modulation function (left) Variational, (right) Evolutionary program. Both show the same behavior. *Surprise*: Errors give rise to larger corrections than correct examples. *Performance*: This difference increases as the NN has been exposed to more examples and e_g decreases

leading to practically optimal algorithms for this particular learning scenario. In this case

$$\bar{F}(\mathbf{V}) = \frac{1}{\sqrt{2\pi}}\sqrt{1-\rho^2}\,\frac{e^{\frac{-h^2}{2\lambda^2}}}{erfc(-\frac{h\sigma_B}{\sqrt{2}\lambda})}. \tag{8}$$

where $\lambda = \sqrt{1-\rho^2}/\rho$ The most striking characteristics of the resulting algorithm are the following. The modulation function starts giving the same Hebbian weight to examples indifferent to whether they are correct or not and learning is driven purely by correlation between input and output. As the learning process goes on, the weight of errors is increasingly more important, with correct examples bringing about little if any weight change. At this latter stage the learning occurs by error correcting. So the modulation incorporates the correct annealing schedule.

Surprise, Performance and partial optimization

Correction of errors means that a change in the weights **J** occurs when $\sigma_J \neq \sigma_B$, that is when a **surprise** occurs, i.e the expectation of the answer is not met by the correct answer from the environment. At the beginning of the learning process learning is purely by correlations and the surprise is not important, an ignorant does not learn more by paying different attention to correct and wrong cases when it is wildly guessing. As learning makes errors less frequent, more importance should be given to them. That means the learning process has been changed by its improved *performance*.

We are interested in the evolution of learning systems. Fully optimized algorithms will not appear at the beginning of the evolutionary process, because the important variables have not been identifi ed yet. It is natural to ask: what are the optimal modulation functions that appear if the optimization is done with a restricted **V** set? It is remarkable that the answer points to a temporal order in how the set of variables should be augmented. Does the inclusion of a variable always lead to a fi tter algorithm? No. Consider two variables A and B and the fi tness $\mathscr{F}$. Call A =*surprise* and B =*performance* and call C collectively the rest of the available variables. By temporal ordering we mean the following result:

$$\mathscr{F}(C) = \mathscr{F}(B,C) < \mathscr{F}(A,C) < \mathscr{F}(A,B,C) \tag{9}$$

If the learning algorithm of an IPS that uses only C is to evolve into a better one, then eq. 9 shows that there will be no initial pressure to develop a structure to measure B, whereas the improvement due to the single addition of A may justify the cost of such a structure. This doesn't mean that B is not useful, for the last term shows that it contributes to the increase in fi tness. It means that B is only useful if A is present and so it had to appear before B, thus increase in complexity follows the time ordering
$(C) \rightarrow (AC) \rightarrow (ABC)$.

BAYESIAN LEARNING

A neural network can be thought of an IPS that is quick in the processing but might not be the best possible in performance. The analysis above prompted several workers to try understanding the variational results from a Bayesian perspective. The full Bayesian approach of ([5]) gave the off-line generalization error bounds but it was not constructive in the sense that it did not show if a NN could reach those bounds. In ref. ([8]) the variational method was used for the off-line problem and showed an off-line NN learning algorithm that saturates the Bayes limit by using replica techniques. Then an on-line Bayesian algorithm was proposed by Opper ([6]) and showing that the on-line optimal algorithm described above works as follows. The weight vector **J** is the expected value of a gaussian posterior distribution of weights and $\sqrt{1-\rho^2}/\rho$ is related to the variance of the posterior. The addition of a new example changes the posterior which is no longer a gaussian. The new posterior is projected into the manifold of gaussians by choosing the least information loss, or smallest Kullback-Liebler distance (MaxEnt projection). The change in mean and covariance of the posterior give a learning algorithm together with an annealing schedule which in the TL give the same behavior for non tensorial modulation functions as the variational method.

EVOLUTIONARY PROGRAMING: TO BAYES OR NOT TO BE

Methods

Genetic programming (GP) is not a genetic algorithm (GA). While both ideas belong to the general area of Evolutionary Programming, GA operate on vectors of parameters while GP deals with programs which have *a priori* no specifi c from nor size. Our programs here output a number and this will represent a modulation function and thus implement a learning dynamics as in eq. (1). To build the programs we defi ne a set of the variables described above. All variables plus numbers are available although they may not be used or if used, may not be useful. We of course exclude **B** and b since that is giving the answer away. A set of operators is also defi ned so that programs can be written. The operator set includes simple addition, subtraction of vectors and scalars, absolute value, multiplication of scalars, of scalar and vector, dot product. Division, log, sqrt, exp are protected so their arguments do not get out of bounds. Variables and operators are referred to in general as atoms. The programming language is LISP.

N_{pop} (typically 500) programs are created at random, taking care that syntactic rules are obeyed so programs compile and run without an error message in a given amount of time. Such programs are called *faithful*. If this maximum time is not enough to stop, then the program is killed. The population of perceptrons is presented sequentially with examples and the generalization error is measured. Since errors are not fatal and we want to obtain algorithms that are best possible at every stage of the learning curve, we measure an integrated fitness function $\mathscr{F}_k = -\sum_{\mu=1}^{P} \mu e_g(\mu)$, for each member k of the population. P is the age span of the NN. From this fitness function we select the programs that will serve to form the next generation which will also contain N_{pop} NNs. The new population will be constructed by applying the operators of GP. We use asexual reproduction, mutation and crossover. By asexual reproduction we mean that the top fraction, say the top ten percent programs are copied to the next generation. Mutation is implemented by making just a random change of an atom by another of the same type, to maintain faithfulness and then introducing the new individual into the next population. Crossover requires more explanation. Programs are to be thought about by their representation of their *parsing trees*. Two programs of the population are chosen by a tournament to be described below. The parsing trees are cut at compatible random points, and the cut branches are exchanged. The two new programs go to the next generation. For the tournament choose a ,$1 < a \leq P$, and ten programs. The one with the lowest $e_g(a)$ is chosen as a parent for crossover. Our crossover uses one parent obtained by tournament in the top fraction and another parent chosen by tournament form the whole population. Our learning bounds have been obtained in the TL but due to heavy numerical costs we have kept the vectors' dimension at $N = 11$.

RESULTS AND CONCLUSIONS

Again we stress that rerunning a simulation with a different random seed will give different results. Our aim is in identifying robust features that occur in a typical run. Not every run works in the sense that efficient learning rules appear. Some get trapped in evolutionary dead ends, where either programs get reduced to a single symbol or algorithms fail to learn, within the time allowed for a simulation [1]. Broader scope simulations with interactions with other species will probably lead these perceptron to extinction. But we have found that in a large fraction of runs (6 of 15), interesting things do happen. Although final programs are very different from run to run, they implement essentially the same algorithms with numerical results as good as the bounds. Fig. 1 shows the modulation function as a function of the field $\sigma_B h$ at left from the variational study and at right from a latter stage in a typical good run. Both figures show the main characteristics a good learning algorithm should have. At the early stages, learning occurs by correlation while at later times the surprise of an error gives rise to a large corrections, while correct results do not cause great change in the weights. In

[1] About one week on a 15 node Beowulf cluster

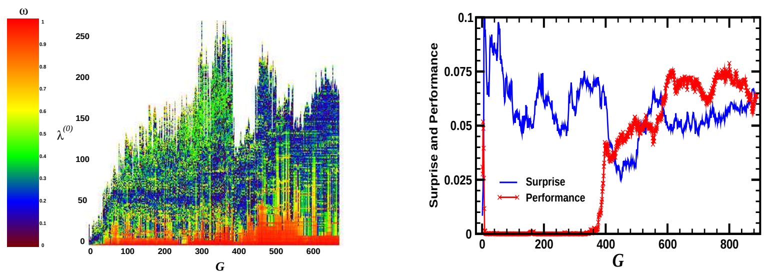

FIGURE 2.

fi g. 2 (left) we show information about the the best of generation (BOG). A program can be represented by a parsing tree, but is written as a list, a linear concatenation of symbols. Each symbol at a given position in the list appears in the population in an equivalent position with a certain frequency. Each vertical line in fi g. 2 represents the list of the BOG for a given generation and is color coded to show the frequency of that specifi c symbol at a given position. The higher positions are for the symbols nearest to the leaves of the parsing trees and the lowest are nearer to the bottom. The fact that symbols that appear at the bottom are very frequent in the population means that certain combinations of symbols, which represent good traits for the modulation functions, invade the population early on. In this particular example an absolute value operator ensures that the modulation function is positive. At the beginning this proves so much better than no program will survive with out that feature.

The dynamics shows sudden changes associated to the appearance of a new combination of variables in a favourable place in a program and the subsequent invasion of the population. Fig. 2 (right) shows two such invasions. We were able, with hindsight borrowed from the variational approach, to identify the structures that appeared and drove the dynamic transitions. The curve labeled *surprise* shows the frequency over the whole population of the combination of symbols $\sigma_B h$. This quantity is positive if the answer is correct ($sign(h) = \sigma_B$) and negative if wrong ($sign(h) \neq \sigma_B$). If it is large then the answer is easy, as small perturbations in **S** will not change the answer and diffi cult if it is near zero. For $\sigma_B h$ large positive there is no surprise, for $\sigma_B h \approx 0$ the answer is diffi cult, errors are likely but for $\sigma_B h$ large and negative the confi dent prediction turns out to be wrong. After about 8 generations there is a large fraction of programs that use this information and it is a very common combination. We cannot tell from the frequency that it is being used correctly always, but it shows that some correct use of the information must be occurring because of the invasion.

The second curve in fi g. 2 (right) shows the invasion of the population by the combination of symbols **J**.**J**. This gives in stationary environments, rise to a schedule annealing variable, negatively correlated to the covariance of the posterior distribution in the Bayesian approach and to ρ^2 of the variational method. It can appropriately be dubbed a *performance* variable.

In all runs where good results were obtained the *surprise* transition happened **before** the *performance* transition.

This temporal order is in total accord with the results of the variational program with restricted sets and in particular eq. 9. The fact that it happened should be analyzed in the light that these are different optimization methods.

We reported on part of a long program of studies. The variational program has been applied to more complex NN, with hidden units [9, 12], with additive or multiplicative noise [8],in drifting environments , adaptation to large sudden changes and more [11]. The offline variational approach for perceptrons gives exactly the Bayes limit and a NN learning algorithm that reaches that limit [8].

That the variational approach to determining learning algorithms, when possible to calculate, gives Bayes like results should not come as a surprise. The interesting surprise comes from looking at neuropsychological literature. Brain activity, as can be assessed from the study of lesions and from functional imaging is quite localized on a macroscopic scale. In particular, there are structures in the brain responsible for neuropsychological activities that are of direct interest to us. The amygdala has been pointed ([16]) as responsible for identifying the *mismatch* or the *surprise* element in the processing of new information. Also, work on prefrontal syndrome patients has clearly indicated the role of the prefrontal lobe in evaluating *performance* levels for on-line (working memory) procedures ([17]). Patients with bilateral damage on the ventrial prefrontal cortex have shown the perseverance effect, not changing strategies that once proved useful just as *lesioned* perceptrons that cannot evaluate performance ([11]). Most interestingly is the fact that the amygdala is an **older** structure than the prefrontal lobe in the phylogenetic evolution of the brain.

Studying Bayesian limits under constraints has pointed out important variables that have a role independent of the details of a particular realization of the evolution fi lm. A temporal order in the increase of complexity by the introduction of structures that can measure specifi c order parameters permits beginning to think that similar mechanisms have governed the evolution of our own brain.

Research supported partially by FAPESP. We thank Osame Kinouchi, Mauro Copelli and Renato Vicente for very useful discussions.

REFERENCES

1. J. R. Koza *Genetic Programming* (MIT Press, Cambridge, Ma, 1992)
2. G. L. Valiant *Comm ACM* **27** (1984), pp. 1134
3. O. Kinouchi and N. Caticha *J. Phys A: Math and Gen*, **25**, (1992), pp. 6243
4. G. Reents and R. Urbanczik *Phys. Rev. Lett*, **80** (1998), 5445
5. M. Opper and D. Haussler *Phys. Rev. Lett*, **66**, (1991), pp. 2677
6. M. Opper *Phys. Rev. Lett*, **77**, (1996), pp. 4671
7. A. Engels and C. Van den Broeck *Statistical Mechanics of Learning* , Cambridge UP (2000)
8. O. Kinouchi and N. Caticha *Phys Rev E*, **53**,(1996), pp 6341
9. M. Copelli and N. Caticha *J. Phys A: Math and Gen*, **28** (1995) , pp 1615
10. N. Caticha and O. Kinouchi *Phyl Mag B*, **77**,(1998), pp 1565
11. R. Vicente, O. Kinouchi and N. Caticha *Machine Learning*, **32**, (1998), pp. 179
12. R. Vicente and N. Caticha *J. Phys A: Math and Gen*, **30** (1997), pp L599
13. M.Biehl and N. Caticha in *Handbook Brain Theory and NN*, M. A. Arbib, Ed., MIT Press (2003)
14. N. Caticha and E. A. de Oliveira *Phys Rev E*, **63**,(2001), pp 061095
15. J. P. Neirotti and N. Caticha *Phys Rev E*, **67**,(2003), pp 041912
16. E. R. Kandel and J. H. Schwartz, eds *Principles of Neural Science* (Elsevier)
17. G. V. Williams and P. S. Goldman-Rakic *Naure*, **376**, pp 572