Genetic Lander: An Experiment in Accurate Neuro-Genetic Control

Edmund Ronald - Marc Schoenauer

Centre de Mathematiques Appliquees, Ecole Polytechnique, Palaiseau.

Abstract. The control problem of soft-landing a toy lunar module simulation is investigated in the context of neural nets. While traditional supervised back-propagation training is inappropriate for lack of training exemplars, genetic algorithms allow a controller to be evolved without difficulty: Evolution is a form of unsupervised learning. A novelty introduced in this paper is the presentation of additional renormalized inputs to the net; experiments indicate that the presence of such inputs allows precision of control to be attained faster, when learning time is measured by the number of generations for which the GA must run to attain a certain mean performance.

1 Introduction to Neuro-Genetic Control

The research presented in this document is part of an ongoing investigation concerning heuristic methods of trajectory planning and control. The underlying methodology is to control a dynamical system by means of a neural net. As training exemplars are not available, classical backpropagation [PDP] cannot be employed to train the net; our solution is to view the matrix of weights as a vector of real numbers, and to optimise controller performance by means of a genetic algorithm. The reader will find an overview of evolutionary methods in neural-net training in [Yao 93].

For our purposes, the basic feasibility of neuro-genetic trajectory planning was established in [Schoenauer & Ronald 94], a study of genetic net training applied to the truck backer-upper benchmark problem ([Nguyen & Widrow 90]). Our experimentation with the truck problem indicated that achieving the convergence of the GA to find a controller net requires modest amounts of computer time,but the GA does not adequtely fine-tune the net for precision control.

Figure 1. shows a typical experiment with our truck-backer upper simulation. It can be seen that the truck makes a very good start at docking, from many starting positions, but is unable to follow up its good start with a precise end-manoeuvre. This problem with local fine tuning in neural net training by GA has been mentioned by Hiroaki Kitano [Kitano 90] who contemplating the biological implications of this inadequacy believes that "A need for a supplementary scheme for fine-tuning is clear...a cascaded scheme of natural selection, adaptation and learning is a necessary consequence for survival".

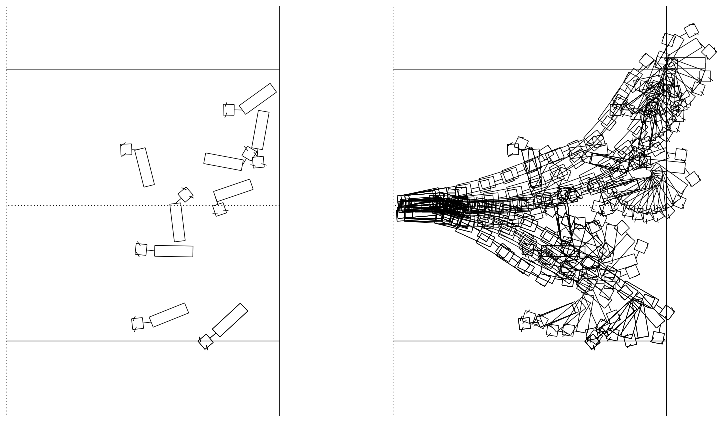


Figure 1: 8 random starting points for truck and ensuing docking traffic.

The methodology we developped for controlling the truck is very similar to that employed for the lunar lander, so we shall review this earlier work. We shall then outline our motives for switching to the simpler and better known lunar lander problem, and define the objective of the numerical experiments which form the core of this paper.

The truck backer-upper problem is fully specified in [Nguyen & Widrow 90]. We can give an abridged description by saying that a driver must back up a truck with trailer system, with the aim of positioning the back of the trailer accurately at a loading dock.

To solve this problem by means of neuro-genetic learning, a neural net controller that "drives" the net was evolved by means of a genetic algorithm. This neural net driver takes the position and orientation of the trailer, as well as the truck/trailer angle as its 4 inputs. The truck is assumed to move backwards by small uniform steps, and so the only output computed by the neural net is the angle of the steering wheels. A population of such "neural drivers" is evolved by means of a GA with the fitness function some combination of docking accuracy and trajectory length.

In the truck study we investigated two possible approaches to neuro-genetic control: On-line learning, and off-line learning.

- In on-line learning, neural net "drivers" are generated on the fly; they only solve the control problem for *one* starting configuration.
- In off-line learning one attempts to train a driver who is able to reach the goal from *any* starting configuration.

It can be seen that off-line learning is extremely time-consuming, as a representative subset of the input space must be trained for. In the case of the truck, on the fly training took a few minutes on a workstation rated at 100 SpecFp 92, while training times exploded to several hours for a driver able to start from 15 predetermined starting positions. Bear in mind that as the dimensionality of the space of staring configurations increases, the cardinality of test ensembles spanning that space increases exponentially. Thus off-line training may lose its appeal due to combinatorial explosion.

As can be seen in Figure 1, the convoluted trajectories of the truck backer-upper do not encourage intuitive interpretation of the control strategies adopted by the nets we evolve. In particular, a human operator will find the truck rather hard to dock by hand, and it thus becomes difficult to judge, other than by the numbers, whether a net has a "better" strategy than another net.

For the above reasons, the choice of a simpler problem was deemed preferable, to allow us to investigate ways in which accurate solutions could be syntehsized on the fly in a small number of generations. Ideally, such a speed-up of the learning process would demonstrate the feasibility of using a GA for real-time control.

In the hope of increasing control precision, and perhaps even of gaining a speedup sufficient to demonstrate the feasibility of real-time neuro-genetic control, we decided to tackle a simpler problem than the truck, to wit the lunar lander, which has also been studied in [Suddarth 88]. In our study of lunar lander we introduced a modification in the neural domain of the neuro-genetic method, namely larger nets with input data renormalisation.

Data renormalisation is the the main novelty studied in detail in this paper. This simply entails supplying the controller with redundant, rescaled, copies of its inputs. Although trivial to implement, this technique allowed us to achieve good accuracy with short learning times, for both the truck backer-upper and the lunar lander problems. The experiments reported below show a strong speedup of convergence, yielding much improved results: Convergence to accurate control occurs in fewer generations, mis-convergence occurs more rarely, and the mean accuracy of control improves dramatically.

2 Lunar Lander Dynamics

This paper deals with training a neural net to land a simulated rocket-driven lunar-lander module, under a gravity of 1.5 m/s^2. This simulation has given rise to numerous computer games since the advent of interactive computing, and will be familiar to most readers.

The lunar module is dropped with no initial velocity from a height of 1000 meters. The fuel tank is assumed to contain 100 units of fuel. Burning one unit of fuel yields one unit of thrust. Maximum thrust is limited to 10 units and the variation of the mass of the lunar module due to fuel consumption is not taken into account. The simulation time-slice was arbitrarily fixed at 0.5 seconds.

The input parameters relayed to the neural net once every time-step are the speed, and the altitude. The net then computes the desired fraction of maximal thrust, on a scale from 0 to 1, which is then linearly rescaled between 0 and 10 units.

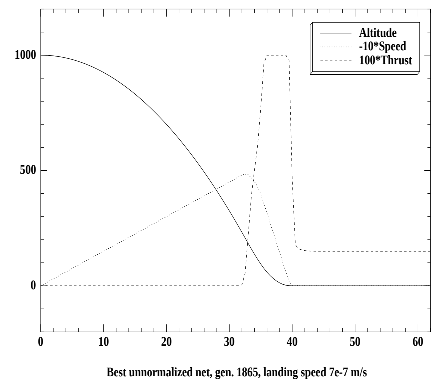


Figure 2: Best non-normalised control action in study.

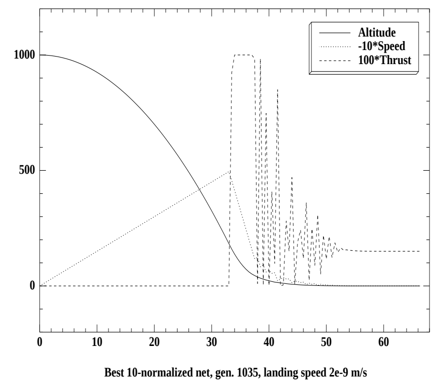


Figure 3: Best overall control action in study.

A very nice lunar landing effected in this study is shown in Figure 2. This achieved a landing speed of 7 $10^{-7} m/s$, ie. $0.7 mm/s$, hardly enough to mark the lunar surface! Such excellence would hardly be necessary in practice. This result was obtained by straightforward application of neuro-genetic control, with no data renormalization.

It was attained at the price of a long run, namely almost 2000 generations. The control action is also exemplary in the economy of fuel in the deceleration phase: For the main deceleration burst, thrust is pulsed in what amounts to almost a square wave to its maximal permissible value. The remarkable landing softness is attained by means of a long - and very smooth- hover phase which begins immediately after deceleration.

The best landing we achieved in this study is shown in Figure 3. The landing speed of 7 $10^{-9} m/s$, ie. $0.002 mm/s$ is a 2 orders of magnitude improvement over the previous result, and was attained in half as many (1035) generations. This is an achievement of data renormalisation: The net was presented with the two previously cited state inputs, namely speed and altitude, and another pair consisting of the same variables pre-multiplied by a factor of 10.

The interesting features of the best "Armstrong", as displayed in Figure 3, are its surprising precision, and the fact that this precision is attained either in spite of, or more probably, because of the displayed sawtooth shape and roughness of the thrust control. Of course, control by rocket to fit a speed tolerance of 20 Angstroms/s seems rather implausible in reality.

3 The Networks

A classical 3-layered net architecture was employed, with complete interconnection between layers 1 and 2, and 2 and 3. The neural transfer function (non-linear squashing) was chosen to be the usual logistic function $\mathcal{F}$, a sigmoid defined by

$$\mathcal{F}(x) = \frac{1}{1+e^{-\alpha x}}$$

In our work the parameter α was fixed, $\alpha = 3.0$. [1] Each individual neuron j computes the traditional [PDP] squashed sum-of-inputs

$$o_j = \mathcal{F}(\sum_i w_i^j x_i)$$

Regarding the sizing of the middle layer, we chose to apply the Kolmogorov model [Hecht-Nieslen 90] which for a net with n inputs and 1 output requires at least 2n+1 intermediate neurons.

Only two net architectures occur in this paper. Both types have only one output (controlling the lunar module's thrust). The canonical method for solving the control problem entails 2 inputs, namely speed and altitude, appropriately normalised between 0 and 1. However, the optimisation by input renormalisation which forms the core of this paper entails adding two inputs to the above cited net, therefore employing 4-input nets. As in both cases we have adhered to the Kolmogorov paradigm, we are studying 2-5-1 and 4-9-1 nets, which respectively have 21 and 55 weights/biases. With the topology fixed, training these nets for a given purpose is a search in a space of dimension 21 or 55. In the neuro-genetic approach, this search is effected by a genetic algorithm.

4 The Genetic Model

The experiments reported in this paper were done with a homebrew general-purpose genetic algorithm package embodying the principles described both in [Holland 75], and [Goldberg 89]. Our genetic algorithm software subjects a small population of nets to crude parody of natural evolution. This artificial evolutionary process aims to achieve nets which display a large *fitness*. The fitness is a positive real value which denotes how well a net does at its assigned task of landing the lunar module. Thus our fitness will be greater for slower landing speeds. The details of the calculation of the fitness are found below.

For the purpose of applying the genetic algorithm, a net is canonically represented by its weights, i.e. as a vector of real numbers. During the course of this work, two distinct homebrew GA software packages were employed. One package followed the first methods presented by John Holland in that it uses bit-strings to encode floating-point numbers. The second software package, described below, was a hybridized GA which directly exploits the native floating-point representation of the worktations which it was run on. The hybridization towards real numbers is described in [Radcliffe 91] and [Michalewicz 92]. The results obtained

[1] An experiment in searching for appropriate values of α is reported in [Schoenauer & Ronald 94]

with both programs were consistent, and only the experiments with the floating point package are detailed in this document.

The genetic algorithm progresses in discrete time steps called generations. At every generation the fitness values of all the nets in the population are computed. Then a new population is derived from the old by applying the stochastic selection, crossover and mutation operators to the members of the old population.

- Selection is an operator that discards the least fit members of the population, and allows fitter ones to reproduce. We use the roulette wheel selection procedure as described in [Goldberg 89], with fitness scaling and elitism, carrying the best individual over from one generation to the next.
- Crossover mixes the traits of two parents into two offspring. In our case, random choice is made between two crossover operators: Exchange of the weights beteween parents, at random positions. Or assigning to some of the weights of each offspring a random barycentric combination of its parents' weights.
- Mutation randomly changes some weights by adding some gaussian noise.

The experiments described in the next section used the following parameters in the GA: The net connection weights forming the object of the search were confined to the interval $[-10, +10]$ thereby avoiding overflow conditions in the computation of the exponential. Population size was held to 50 over the whole length of each run, fitness scaling was set to a constant selecetive pressure of 2.0, crossover rate was 0.6, mutation rate 0.2, the gaussian noise had its standard deviation set to half the weight space diameter, ie. 10.0, decreasing geometrically in time by a factor 0.999 at each generation.

5 Results

A spectrum of experiments was carried out with the bit-string and real-number GA's, to determine the effect of various parameter changes on the control accuracy and speed of convergence of the on-line neuro-genetic control method. A single starting point (speed=0m/s, altitude=1000m) was employed across the board.

For all tests, 2 state variables were taken as basis for the inputs to be presented to the net:

$$NormSpeed = Speed/100, NormAltitude = Altitude/1000$$

the fitness of a controller was computed to be

$$fitness = \frac{1}{(0.1 + {Speed_{crash}}^2)}$$

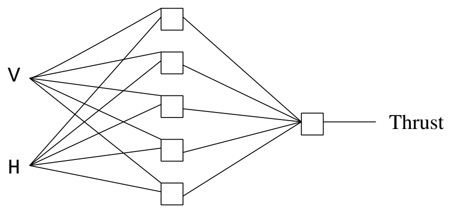
Thus the fitness for a net is some number between 0.0 and 10.0

The following controller architectures were investigated:

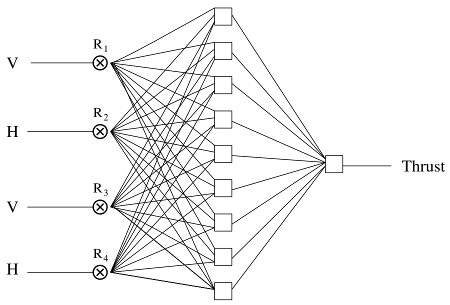
- 2-5-1 net. Inputs are $NormSpeed$ and $NormAltitude$
- 4-9-1 net. Inputs are $NormSpeed$ and $NormAltitude$, and the same renormalized by a factor of 100
- 4-9-1 net. Inputs are $NormSpeed$ and $NormAltitude$, and the same renormalized by a factor of 10
- 4-9-1 net. Inputs are $NormSpeed$ and $NormAltitude$, and the same renormalized by a factor of 1

The 1-renormalized net gets precisely the same inputs as the 2-input net, only of course, it gets them twice.

In the diagram below the two net architectures are diagrammed. We have denoted by r_i the renormalisation coefficients; the experiments consist of setting r_3 and r_4 to 1, 10, or 100, with r_1 and r_2 set to 1.



2-input Net



Generalised 4-input Net

A total of 60 GA runs was made for each of these architectures, with the population size set to 50 individuals. For each run, at each generation, the best individual's performance was saved to disk. At the end of each batch of 60 runs, the mean at each number of generations of each architecture's best individuals' performance was computed. The corresponding graph is plotted in Figs 4 and 5, in linear and logarithmic scales:

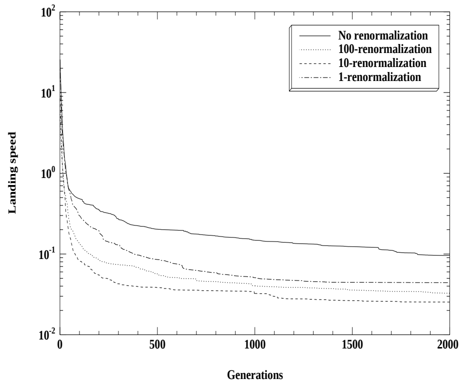


Figure 4: Mean performance, lin / log scales.

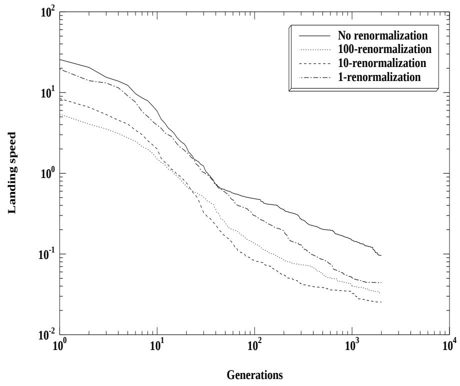


Figure 5: Mean Performance, log / log scales.

In our opinion, two facets of learning, as evidenced by these graphs, are of interest: Long-term learning, ie. asymptotic behaviour (best observed on the lin / log plot) and on the other hand fast learning ie. in few generations as best observed in the log / log plot.

– The asymptotic behaviour of the mean landing speeds demonstrates the superiority of the renormalised over the non-renormalized nets by the fact that the 4 presented curves are layered, in the order un-normalised / 1 / 100 / 10. In fact, in ultimate mean precision attained over 2000 generations the 10-normalised net wins by half an order of magnitude.
– The close-up, immediate behaviour of the GA demonstrates that this gain carries over to the fast-learning region. A new graph, in which we assess the ratio of the number of generations (normalised/ un-normalised) to reach some speed between 0.1 and 1.0 m/s. Figure 6 shows the speed-up to be truly useful, with computation proceeding faster by a factor of 25 at the speed of 0.1 m/s (4 inches/s).

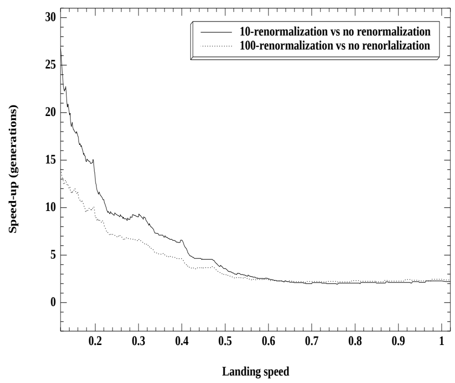


Figure 6: Computational speedup achieved through normalisation.

6 Discussion

The above comparison of computational costs employs the number of generations as a metric. However, in fact the normed nets are larger, (55 vs. 21 weights), and thus slow down the programs by a measured factor of about 2. This is more than offset by the gains in learning performance.

A rather surprising feature can be discerned in the log / log graph: at generation 1, the average best performance per run is already well under 10 m/s for the 1-normalised net. This led us to investtigate more closely the properties of pure random search.

The results are displayed in Figure 7. The "Generations" referred to in the legend correpond to packets of individuals of the same size as the population in the GA.

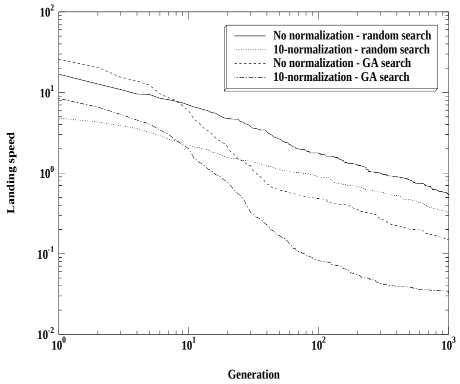


Figure 7: Random search outperforms genetic learning for first "generations".

Pure random search in the space of nets consistently outperforms genetic search for very fast learning, when very approximate precision of control is sufficient. This seemingly counter-intuitive result seems to indicate that random search may supply a basis for real-time control.

In conclusion, we believe that the fine-tuning problem cited by [Kitano 90] can be overcome in control applications by optimising the net architecture specifically for genetic learning. This optimisation can be aided by employing the GA itself to tune the normalisation coeficients [Schoenauer & Ronald 94]. In biological terms, this means we supply supplementary neural pre-adapted for optimal sensitivity at different levels of sensory stimulation.

However, speed remains an issue and as regards real-time control, cascaded random search may yet have its revenge over the sophisticated methods of evolutionary computation.

References

[Angeline, Saunders & Pollack 94] P. J. Angeline, G. M. Saunders and J. B. Pollack, An evolutionnary algorithm that construct recurrent neural networks. To appear in *IEEE Transactions on Neural Networks.*

[Goldberg 89] D. E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, Addison Wesley, 1989.

[Harp & Samad 91] S. A. Harp and T. Samad, Genetic synthesis of neural network architecture, in *Handbook of Genetic Algorithms*, L. Davis Ed., Van Nostrand Reinhold, New York, 1991.

[Hecht-Nieslen 90] R. Hecht-Nielsen, *Neurocomputing*, Addison Wesley, 1990.

[Holland 75] J. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Harbor, 1975.

[Kitano 90] Empirical studies on the speed of convergence of neural network training using genetic algorithms. In *Proceedings of Eight National Conference on Artificial Intelligence, AAAI-90*, Vol. 2, pp 789-795, Boston, MA, 29 July - 3 Aug 1990. MIT Press, Cambridge, MA.

[Michalewicz 92] Z. Michalewicz, Genetic Algorithms+Data Structures=Evolution Programs, Springer Verlag 1992.

[Nguyen & Widrow 90] D. Nguyen, B. Widrow, The truck Backer Upper: An example of self learning in neural networks, in *Neural networks for Control*, W. T. Miller III, R. S. Sutton, P. J. Werbos eds, The MIT Press, Cambridge MA, 1990.

[Radcliffe 91] N. J. Radcliffe, Equivalence Class Analysis of Genetic Algorithms, in *Complex Systems* **5**, pp 183-205, 1991.

[PDP] D. E. Rumelhart, J. L. McClelland, *Parallel Distributed Processing - Exploration in the micro structure of cognition*, MIT Press, Cambridge MA, 1986.

[Schaffer, Caruana & Eshelman 1990] J. D. Schaffer, R. A. Caruana and L. J. Eshelman, Using genetic search to exploit the emergent behaviour of neural networks, *Physica D* 42 (1990), pp244-248.

[Schoenauer & Ronald 94] M. Schoenauer, E.Ronald Neuro Genetic Truck Backer-Upper Controller, in *IEEE World Conference on Computational Intelligence* , Orlando 1994.

[Schoenauer & Ronald 94] M. Schoenauer, E.Ronald Genetic Extensions of Neural Net Learning: Transfer Functions and Renormalisation Coefficients, in *submitted to Artificial Evolution 94* , Toulouse 1994.

[Suddarth 88] Steve C. Suddarth, The symbolic-neural method for creating models and control behaviours from examples. Ph.D. dissertation. University of Washington. 1988.

[Yao 93] Xin Yao, A Review of Evolutionary Artificial Neural Networks, in *International Journal of Intelligent Systems* **8**, pp 539-567, 1993.