A Coevolutionary Approach to Learn Animal Behavior Through Controlled Interaction

Wei Li
Department of Automatic Control and Systems Engineering
The University of Sheffield, UK
wei.li11@sheffield.ac.uk

Melvin Gauci
Department of Automatic Control and Systems Engineering
The University of Sheffield, UK
m.gauci@sheffield.ac.uk

Roderich Groß
Department of Automatic Control and Systems Engineering
The University of Sheffield, UK
r.gross@sheffield.ac.uk

ABSTRACT

This paper proposes a method that allows a machine to infer the behavior of an animal in a fully automatic way. In principle, the machine does not need any prior information about the behavior. It is able to modify the environmental conditions and observe the animal; therefore it can learn about the animal through controlled interaction. Using a competitive coevolutionary approach, the machine concurrently evolves *animats*, that is, models to approximate the animal, as well as *classifiers* to discriminate between animal and animat. We present a proof-of-concept study conducted in computer simulation that shows the feasibility of the approach. Moreover, we show that the machine learns significantly better through interaction with the animal than through passive observation. We discuss the merits and limitations of the approach and outline potential future directions.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Knowledge acquisition*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; I.2.9 [**Artificial Intelligence**]: Robotics—*Autonomous vehicles*

Keywords

Science automation; animal behavior; coevolution; Turing test; evolutionary robotics; interaction; artificial life

1. INTRODUCTION

The scientific study of animal behavior—ethology—has been pursued for centuries [2]. The most widely used approach to infer the behavior of an animal is through observation. The animals under observation can be studied in either uncontrolled or controlled environmental conditions. If the conditions are not controlled, such as observing the animal in its natural habitat, it can be difficult to infer how the behavior is influenced by the stimuli provided by the environment. On the other hand, the problem of how to control the environmental conditions in a meaningful way is difficult, which is therefore solved by humans in the majority of cases.

Recent developments in science automation suggest that machines could autonomously conduct scientific investigation [7]. Inspired by this idea, we propose a new approach to learning about animal behavior. Our method allows a machine to infer the behavior of an animal in a fully autonomous way. In particular, it should be able to predict any observable action of the animal. Our assumptions are:

- The machine can observe the animal's actions. This paper focuses on the animal's displacement in space and assumes that it is possible to track the position of the animal at discrete steps in time.
- The machine is capable of simulating any relevant actions of the animal. In this paper, this constitutes being able to produce arbitrary sequences of numbers (coordinates in space).
- The machine must be able to control the environmental conditions throughout the experiment. In this paper, we focus on the intensity of the ambient light.

Our approach uses a coevolutionary algorithm comprised of two populations. The first population contains the animats, hereafter also referred to as *models*. The second population contains *classifiers*. The populations co-evolve competitively. The fitness of the classifiers depends solely on their ability to distinguish the behavior of the animats from the behavior of the animal under investigation. The fitness of the models depends solely on their ability to mislead the classifiers into making the wrong judgement, that is, classifying them as the animal.

In principle, our approach does not require any prior knowledge about the animal. It is not even necessary to identify how to gauge the differences between animals and animats. Interestingly, the approach could therefore be used to learn what constitutes human behavior in general, by letting the machine interact with a large number of humans. As a result, the models could potentially pass the Turing test [17]. Moreover, the classifiers could serve as Reverse Turing tests—enabling machines to distinguish between artificial and human behavior. This capability would be of wide interest. The "Completely Automated Public Turing test to tell Computers and Humans Apart" system (CAPTCHA) [9], for example, is widely used in Internet security.

This paper is organized as follows. Sec. 2 presents related work. Sec. 3 presents the methodology used, including the simulated animal behavior and the proposed coevolutionary algorithm. A proof-of-concept study, conducted in computer simulations, is presented in Sec. 4. Sec. 5 concludes the paper.

2. RELATED WORK

2.1 The Development of Science Automation

The field of science automation has been developed to a great extent because of the increasing demands of drug industry and relevant fields of biology and chemistry. Using high-throughput screening [14] technology, for example, one can automatically analyze a large number of potential drug candidates, which allows researchers to do thousands of pharmacological, biological and chemical experiments in a short time. According to Schmidhuber [15], by 2040, machines will be sufficiently 'intelligent' to make scientists redundant. King et al. [18, 10], for example, developed Adam, a robot scientist that could automatically generate functional genomics hypotheses about the yeast *Saccharomyces cerevisiae* and carry out experiments to test and refine hypotheses. Different from this work, our approach is not specific to any particular types of organisms.

In the system identification area, Gauld et al. [8] developed DAISY, a system to identify biological species automatically with high accuracy using advanced image processing techniques. MacLeod et al. [12] report that an imaging system that was originally designed for identifying marine zooplankton was used by the US government for monitoring horizon oil spill in deep waters. They argue that taxonomists and researchers in machine learning, pattern recognition as well as artificial intelligence should collaborate with each other in order to improve automation in taxonomic identification.

2.2 System Identification through Coevolutionary Algorithms

In the last decade, a number of works considered coevolutionary approaches to system identification (e.g. [11, 13]). In [4], Bongard and Lipson present a nonlinear system identification method called *estimation-exploration algorithm*, which co-evolves tests and models in a way that minimizes the amount of tests (in this study performed in simulation). In one of the scenarios, the models, composed of 17 parameters, have to approximate a mobile robot that gets damaged. One difficulty of the approach is that it requires a metric to estimate the differences between the models and the robot. In [3], "the observation was made that in many cases the simulated robot would exhibit wildly different behaviors even when it very closely approximated the damaged 'physical' robot. This result is not surprising due to the fact that the robot is a highly coupled, non-linear system: thus similar initial conditions [...] are expected to rapidly diverge in behavior over time". Bongard and Lipson address this problem using a more refined comparison metric reported in [3]. The novelty of our work is that it does not require any pre-defined metrics. Rather, the classifier learns to distinguish between the animal and model. Moreover, our approach allows the classifier to interact with the environment, by determining the experimental conditions on the fly in response to the observed behavior.

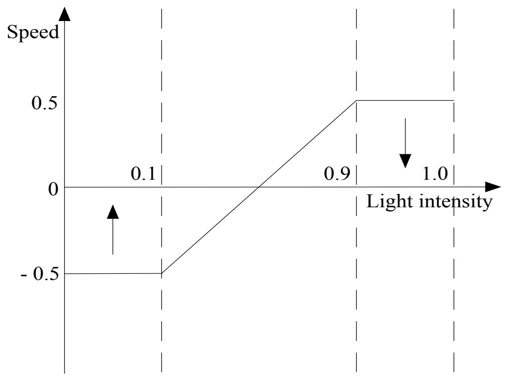


Figure 1: The speed of the animal as a function of the light intensity in its environment.

Table 1: This table shows the factor by which the animal's 'base' speed (shown in Fig. 1) is multiplied, for an example state sequence.

state	M	L	H	H	L	L	L	H	H
factor	1	1	1	α_1	1	α_1	α_1^2	1	α_2

L	L	H	H	L	L	L	L	M	H	H
1	α_2	1	α_3	1	α_3	α_3^2	α_3^3	1	1	1

Bongard et al. [5] study the problem how a robot can infer its own morphology through a process of continuous self-modeling. The approach generates test candidates as well as actions that lead to a maximum disagreement among the models. This greatly helps to minimize the number of tests performed on the real robot. The approach is not applicable to our scenario, as it is not in general possible to make the animal execute arbitrary instructions on demand. The approach described in [16], where the authors infer physical laws from observing mechanical systems, would also be applicable to learn about the behavior of an animal. Different from this approach our system learns about the behavior not through passive observation, but rather through an interactive process.

3. METHODOLOGY

This section describes the setup used in this paper to illustrate the ideas presented earlier. Sec. 3.1 presents the simulated animal behavior to be identified. Sec. 3.2 details the coevolutionary approach and its implementation.

3.1 Simulated Animal Behavior

The behavior to be identified was chosen to serve as a tractable test-bed for our approach; while it may loosely correspond to how some real animals react to the light intensity in their environment, it is not intended to mimic any specific animal. In this behavior, non-trivial interaction with the animal is critical for leading the animal to reveal a subsection of its behavioral repertoire.

We simulate a one-dimensional environment in continuous space. The simulation advances in discrete time steps $t \in \{0, 1, 2, \dots\}$. The (ambient) light intensity in the environment, I, can be varied continuously between 0 and 1.

The animal distinguishes between three levels of light intensity, low ($0 \leq I < I_L$), medium ($I_L \leq I \leq I_H$), and high ($I_H < I \leq 1$), with each level corresponding to a state of the animal. Hereafter, these states will be referred to as L, M, and H.

If the animal is in state M at time t, its speed, $s^{(t)} \in \mathbb{R}$, varies linearly with $I^{(t)}$ as:

$$s^{(t)} = k\left(I^{(t)} - 0.5\right), \quad (1)$$

where k is a constant.

If the animal is in state L, its behavior depends on the number of H to L transitions that had occurred since it has last been in state M (or since $t = 0$, if the animal has never been in state M). If no transitions had occurred, then the speed of the animal is $k\,(I_L - 0.5)$, and remains that way for as long as the animal remains in state L. If one H to L transition had occurred, then the speed of the animal decays exponentially with a rate α_1 for every time step that the animal remains in state L. Hence, in the first time step that the animal is in state L, the speed is $k\,(I_L - 0.5)$; then, it changes to $\alpha_1 k\,(I_L - 0.5)$, $\alpha_1^2 k\,(I_L - 0.5)$, and so on. If two H to L transitions had occurred since it has last been in state M (or since $t = 0$, if the animal has never been in state M), then the rate is α_2; if three or more transitions had occurred, then the rate is α_3.

The behavior of the animal in state H is analogous to that in state L. The possible exponential change rates are also α_1, α_2 and α_3; however, which one of these applies now depends on the number of L to H transitions that had occurred since the last M state.

Table 1 shows an example state sequence for the animal, along with the factor by which the animal's 'base' speed (i.e., the speed shown in Fig. 1) is multiplied in each time step.

Here, I_L and I_H are set to 0.1 and 0.9 respectively, k is set to 1.25; hence, the lower and the upper saturation values of the speed are $k\,(I_L - 0.5) = -0.5$ and $k\,(I_H - 0.5) = 0.5$. The exponential rates of change are set to: $\alpha_1 = 0.8$, $\alpha_2 = 0.4$, $\alpha_3 = 0.2$. Thus, in each case, the animal's speed decays exponentially towards zero.

3.2 Coevolutionary Approach and Implementation

The coevolutionary algorithm is comprised of two populations — one of models, and one of classifiers — that co-evolve with each other competitively.

The fitness of the classifiers depends solely on their ability to distinguish the behavior of the models from the behavior of the 'real' animal, described in Sec. 3.1. The fitness of the models depends solely on their ability to mislead the classifiers into making the wrong judgement, that is, classifying them as the 'real' animal.

3.2.1 Model Structure

In this paper, both animal and model share the same structure. The task is thus to estimate parameters k, α_1, α_2, and α_3, as described in Sec. 3.1. As a consequence of this choice, it is simple to gauge the quality of the models obtained (as discussed in the results section). It is worth mentioning that the fitness of the models solely depends on the performance of the classifiers, and that the latter do not have any knowledge about the model structure or parameters.

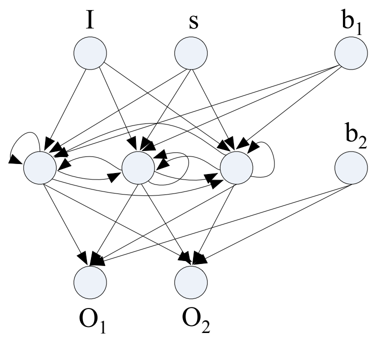


Figure 2: The structure of the classifiers is a recurrent Elman neural network with two inputs, three hidden neurons, and two output neurons. See the text for details.

3.2.2 Classifier Structure

The structure of the classifiers is a recurrent Elman neural network [6] with two inputs, three hidden neurons, and two output neurons (see Fig. 2). The network has a total of 26 parameters, which can all assume values in $\mathbb{R}$. The activation function used in the hidden and the output neurons is the logistic sigmoid, which has the range $(0, 1)$ and is defined as $\text{sig}\,(\cdot) = 1/\,(1 + \exp\,(-\,(\cdot)))$.

One of the inputs to the network is the light intensity in the environment at time step t, $I^{(t)} \in [0, 1]$, while the other input is the speed $s^{(t)}$.

In order to make a judgement between a model and the 'real' animal, the network observes the behavior of the animal (speed) over a period of time; here 10 s at 0.1 s intervals for a total of $T = 100$ time steps. In addition, the network is also in control of the light intensity in the animal's environment. At time $t = 0$, the value of the light intensity is chosen randomly with a uniform distribution in the range $[0, 1]$. The neural network is then updated, using $I^{(0)}$ and $s^{(t)}$. The value of the light intensity for the next time step is obtained from the neural network's output neuron O_1, and the process repeats. After having iterated through all the time steps, the final value of output neuron O_2 is used to make a judgement: the network decides on a model if $O_2 < 0.5$, and on the real animal if $O_2 \geq 0.5$.

3.2.3 Optimization Algorithm

The algorithm used here is based on a $(\mu + \lambda)$ evolution strategy with self-adaptive mutation strengths [1], and can be thought of as consisting of two sub-algorithms — one for the models, and another for the classifiers — which are identical, and do not interact with each other except for the fitness calculation step (described later in Sec. 3.2.4).

In this algorithm, an individual is a 2-tuple, $\mathbf{a} = (\mathbf{x}, \boldsymbol{\sigma})$, where $\mathbf{x} \in \mathbb{R}^{\mathbf{n}}$ represents objective parameters, and $\boldsymbol{\sigma} \in (0, \infty)^n$ represents mutation strengths. The i-th mutation strength in $\boldsymbol{\sigma}$ corresponds to the i-th element in $\mathbf{x}$. For the model sub-algorithm, $n = 4$, for the classifier sub-algorithm, $n = 26$.

Each generation g comprises a population of $\mu = 50$ individuals:

$$\mathcal{P}^{(g)} = \left\{ \mathbf{a}_1^{(g)}, \mathbf{a}_2^{(g)}, \ldots, \mathbf{a}_\mu^{(g)} \right\}.$$

In the population of the first generation, $\mathcal{P}^{(0)}$, all the objective parameters are initialized to 0.0 and all the mutation strengths are initialized to 1.0. Thereafter, in every generation g, the μ parent individuals are first used to create $\lambda = 50$ offspring individuals by recombination. For the generation of each recombined individual $\mathbf{a}_k^{\prime(g)}$, $k \in \{1, 2, \ldots, \lambda\}$, two individuals are chosen randomly, with replacement, from the parent population: $\mathbf{a}_\chi^{(g)}$ and $\mathbf{a}_\psi^{(g)}$, where $\chi, \psi \in \{1, 2, \ldots, \mu\}$. Discrete and intermediary recombination are then used to generate the objective parameters and the mutation strengths of the recombined individual, respectively:

$$x_{k,i}^{\prime(g)} = x_{\chi,i}^{(g)} \quad \text{OR} \quad x_{\psi,i}^{(g)}, \tag{2}$$

$$\sigma_{k,i}^{\prime(g)} = \left(\sigma_{\chi,i}^{(g)} + \sigma_{\psi,i}^{(g)} \right) / 2, \tag{3}$$

where $i \in \{1, 2, \ldots, n\}$ is indexing the elements within the vectors and, in Eq. 2, the selection is performed randomly and with equal probability.

Each of the λ recombined individuals is then mutated in order to obtain the final offspring population, $\mathcal{P}^{\prime\prime(g)}$. This is done according to:

$$\sigma_{k,i}^{\prime\prime(g)} = \sigma_{k,i}^{\prime(g)} \exp\left(\tau' N_k(0,1) + \tau N_{k,i}(0,1) \right), \tag{4}$$

$$x_{k,i}^{\prime\prime(g)} = x_{k,i}^{\prime(g)} + \sigma_{k,i}^{\prime\prime(g)} N_{k,i}(0,1), \tag{5}$$

for all $\{k, i\}$, where $k \in \{1, 2, \ldots, \lambda\}$ is indexing the individuals within the population and $i \in \{1, 2, \ldots, n\}$ is indexing the elements within the vectors. Eq. 4 generates the perturbed mutation strength from the original one according to a log-normal distribution. Eq. 5 mutates the objective parameter according to a normal distribution having the perturbed mutation strength as its variance. In Eq. 4, $N_k(0,1)$ and $N_{k,i}(0,1)$ are both random numbers generated from a standard normal distribution; however, the former is generated once for each individual (i.e. for each value of k), while the latter is generated separately for each element within each individual (i.e. for each combination of k and i). The parameters τ' and τ determine the learning rates of the mutation strengths, and are set as $\tau' = 1/2\sqrt{2n}$, $\tau = 1/2\sqrt{2\sqrt{n}}$ (similar to [19]).

Once the offspring population has been generated, the μ individuals with the highest fitness (see Sec. 3.2.4) from the combined population, $\mathcal{P}^{(g)} \cup \mathcal{P}^{\prime\prime(g)}$ (which contains $\mu + \lambda$ individuals), are selected as the parents to form the population of the next generation, $\mathcal{P}^{(g+1)}$. Individuals with an equal fitness have an equal chance of being selected.

3.2.4 Fitness Calculation

The fitness of each model is obtained by evaluating it with each of the classifiers in the competing population (100 in total). For every classifier that wrongly judges the model as being the real animal, the model's fitness increases by one. Therefore, in the end, the model obtains a fitness in the set $\{0, 1, 2, \ldots, 100\}$.

The fitness of each classifier is obtained by using it to evaluate (i) each model in the competing population (100 in total) once, and (ii) the real animal 100 times. For each correct judgement, the classifier's fitness increases by one, for a final fitness in the set $\{0, 1, 2, \ldots, 200\}$.

4. RESULTS

4.1 Coevolutionary Runs without and with Noise

We performed 100 coevolutionary runs using the setup described in Sec. 3. This setup, in which the classifier is in control of the light intensity in the animal's environment, is hereafter referred to as the "interactive" setup. In order to validate the advantages of the interactive approach, we compared it against the situation where the classifier only observes the animal passively. We considered two such setups: in the first setup (hereafter, "passive 1") the light intensity changes randomly, following a uniform distribution in $[0, 1]$, in every time step. In the second setup (hereafter, "passive 2"), the intensity changes randomly every ten time steps. All other aspects of these two setups are identical to the "interactive" setup. Another 100 coevolutions were performed for each of the two passive setups.

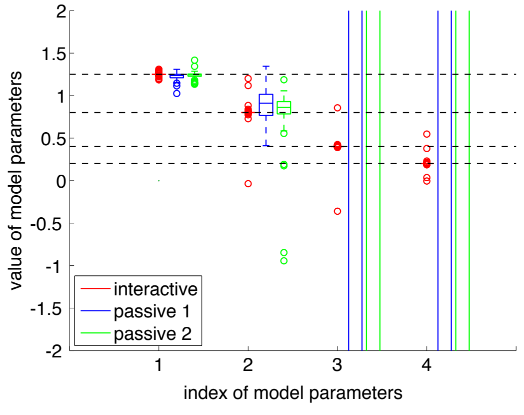
In addition, we considered the situation where both the animal and the learning system are affected by noise. Each of the three setups was tested by performing another 100 coevolutionary runs. The light intensity perceived by the animal is obtained by multiplying the real intensity by a random number generated uniformly in $[0.95, 1.05]$, and capping the perceived intensity to 1 if it exceeds this value. Noise is also applied to the speed of the animal by multiplying the original speed with a random number generated uniformly in $[0.95, 1.05]$. In order to make this setup more feasible to implement, it is assumed that the system cannot directly measure the speed of the animal, but rather its position. Noise on the position is applied by adding a random number generated from normal distribution $N(0, 0.005)$. The speed of the animal for the classifier's input is then calculated by subtracting the previous estimated position from the current estimated position.

4.2 Analysis of the Evolved Models and Classifiers

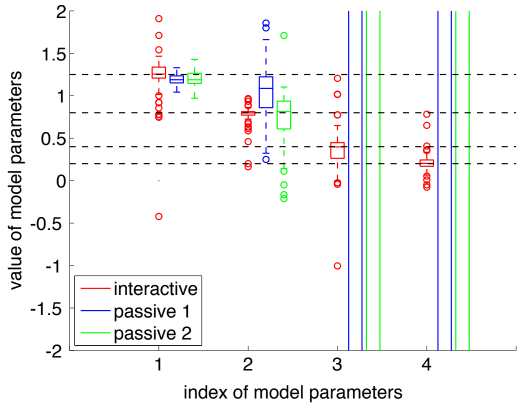
Fig. 3 shows a box plot[1] with the distributions of the best evolved parameters in the 1000th generation of the three coevolutionary setups, for the cases (a) without and (b) with noise.

The passive coevolutions are able to evolve the parameters k and α_1; however, they are not able to evolve α_2 and α_3. If the light intensity changes randomly, it is highly unlikely that the transitions L to H and/or H to L occur enough times, without an M state in between, such that the classifiers can observe the effects of α_2 and α_3. Therefore, the classifiers are not capable of distinguishing the behavior of models from the behavior of the real animal with respect to these two parameters, and in turn, these parameters do not converge to their true value in the model population.

[1]The box plots presented here are all as follows. The line inside the box represents the median of the data. The edges of the box represent the lower and the upper quartiles (25-th and 75-th percentiles) of the data, while the whiskers represent the lowest and the highest data points that are within 1.5 times the inter-quartile range from the lower and the upper quartiles, respectively. Circles represent outliers.



(a) without noise



(b) with noise

Figure 3: This plots shows the distributions of the best evolved parameters in the 1000$^{\text{th}}$ generation in the coevolutions. Parameters 1 through 4 correspond to k, α_1, α_2 and α_3, respectively. The true values are $k = 1.25$, $\alpha_1 = 0.8$, $\alpha_2 = 0.4$ and $\alpha_3 = 0.2$. Each box corresponds to 100 coevolutionary runs. Note that in order to zoom in on the relevant range, some boxes and outliers are omitted from the plot.

In contrast to the passive coevolutions, the interactive coevolution can evolve all four parameters reasonably accurately, in both the noiseless and the noisy cases. The effect of the noise is to widen the distribution of the evolved parameters across the 100 coevolutionary runs; however, the median values of the evolved parameters are still very close to the true values. Interestingly, the interactive coevolution does not seem to learn α_2 and α_3 significantly worse than it does k and α_1. This implies that, by the 1000$^{\text{th}}$ generation, the classifiers have learned how to control the pattern of the light intensity in such a way that they can distinguish models from the 'real' animal from the effect of any of the parameters.

To analyze the behavior of the evolved classifiers, the best evolved classifier from the last generation of a randomly-chosen interactive coevolutionary run without noise was post-evaluated. Fig. 4 shows, over 100 time steps (as used in the coevolution), the pattern of the light intensity set by the classifier and the corresponding speed of the animal. Initially, the classifier outputs a sequence alternating between L and H, which means that by the 20$^{\text{th}}$ time step, it has already observed all four parameters. The light intensity then remains in an L state until around the 70th time step. Interestingly, the classifier now once again makes the light intensity alternate between L and H, which means that it observes the effect of all four parameters for a second time. This repetition may make the classifier more robust in determining whether the behavior it is observing is that of a model or the 'real' animal.

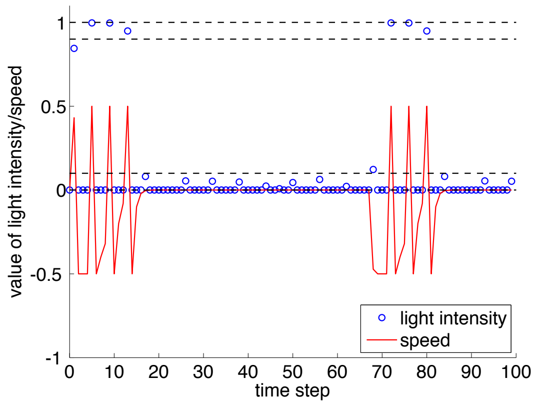


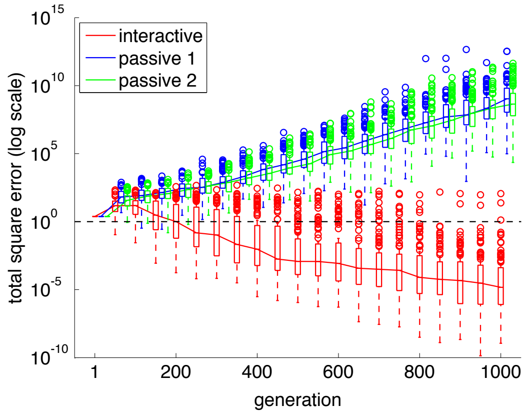
Figure 4: A post-evaluation with the best evolved classifier in the last generation of one of the interactive coevolutionary runs without noise.

4.3 Coevolutionary Dynamics

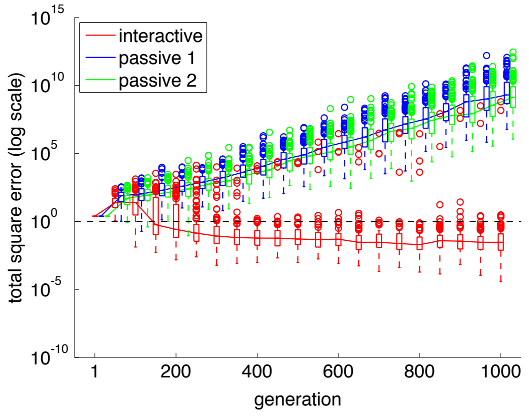
Fig. 5 shows the dynamics of the coevolutionary algorithms for the cases (a) without and (b) with noise. The horizontal axis shows the generation, while the vertical axis shows the total square error of the model parameters, i.e. the sum of the square differences of the four parameters (of the best individual in each generation) from their true values.

In the case of the interactive coevolution, the error quickly starts to reduce after around the 100$^{\text{th}}$ generation, and drops to below 1.0 by the 200$^{\text{th}}$ generation, for both the noiseless and the noisy cases. In the noiseless case, the error keeps improving until the last generation; in contrast, in the noisy case, the error stops improving substantially after around the 400$^{\text{th}}$ generation. In the case of the passive coevolutions, not only does the error not decrease, but it increases, such that the median error approaches 10^{10} by the 1000$^{\text{th}}$ generation.

In order to analyze why the interactive coevolution is successful while the passive ones are not, we can look at the dynamics of the subjective fitnesses of the classifiers and the models (as defined in Sec. 3.2.4) during the course of the coevolution. As the passive coevolutions fail to converge even in the noiseless case, it is sufficient to analyze the fitness dynamics only for this case, for the sake of simplicity. Fig. 6 shows the fitness dynamics of the interactive and the passive 1 coevolutions. In the case of the interactive coevolution, the average fitness of the classifiers starts off at around 0.5, which means that the classifiers make decisions that are no better than random decisions. However, the classifiers

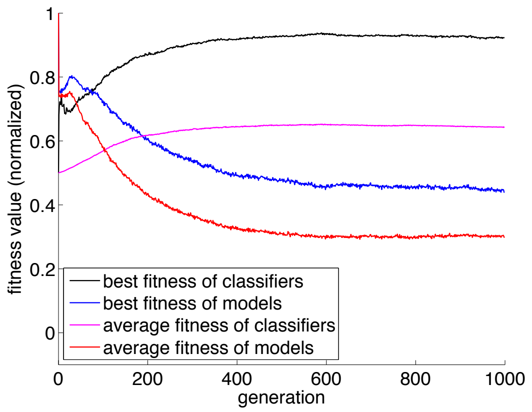


(a) without noise

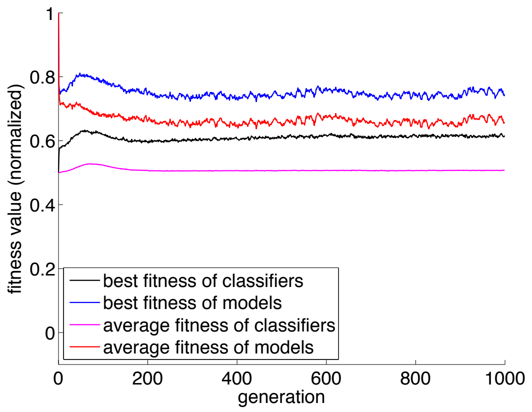


(b) with noise

Figure 5: This plot shows the convergence of the coevolutionary algorithms with and without noise. The horizontal axis represents the generation, while the vertical axis represents the total square error in the parameters of the best model in each generation. Each box corresponds to 100 coevolutionary runs, and the solid lines correspond to the median error.



(a) interactive coevolution (without noise)



(b) passive 1 coevolution (without noise)

Figure 6: This plot shows the normalized fitness of the classifiers and the models in (a) the interactive coevolution, and (b) the passive 1 coevolution. The curves show the average fitness across 100 coevolutionary runs.

quickly improve in fitness, which in turn causes the fitness of the models to decrease. This increases the selective pressure on the models. In the case of the passive 1 coevolution, the average fitness of the classifiers also starts off at around 0.5. In the first few generations, this increases slightly, because the classifiers learn how to distinguish models from the 'real' animal on the basis of parameters k and α_1. However, the models quickly adapt to this new ability of the classifiers. Now, as the classifiers are very unlikely to have the opportunity to observe the effects of α_3 and α_4, their average fitness returns to 0.5. This leads to a disengagement phenomenon, in which there is no more meaningful selection in the model population, leading their error to drift.

We also wish to analyze how the four individual parameters evolve during the course of the only successful coevolutionary setup, i.e. the interactive coevolution. For simplicity, we consider only the noiseless case, which is shown in Fig. 7. This plot reveals how learning proceeds in the coevolution. Parameter k is the first to be learnt, followed closely by α_1, while parameters α_2 and α_3 take a longer time to converge to the true values. This means that the classifiers first learn to distinguish models from the 'real' animal on the basis of k and α_1. This ability of the classifiers drives the model population to rid itself of models that are different to the 'real' animal in terms of k and/or α_1. Eventually, the classifiers also learn to exploit the effects of α_2 and α_3 in order to make the right decision; thereby driving the model population to evolve these two parameters correctly. After about the 500th generation, the learning of the four parameters proceeds with virtually identical rates.

4.4 Using a Simple Evolutionary Algorithm

In order to compare the coevolutionary method against a more traditional approach, we used a simple evolution where a single population of models evolves. As there are now no classifiers, an interactive approach is not possible, and thus we conducted 100 evolutionary runs for the passive 1 and passive 2 methods of changing the light intensity in the animal's environment. The structure of the evolution is identi-

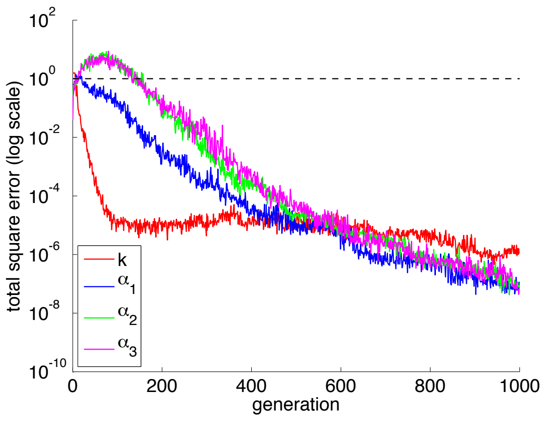


Figure 7: This plot shows how the square error in the individual parameters changes over the generations in the interactive coevolution in the case without noise. The curves correspond to median values from 100 **coevolutionary runs.**

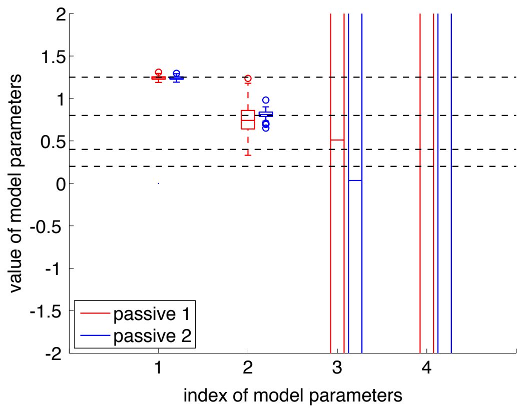


Figure 8: This plots shows the distributions of the best evolved parameters in the 1000th **generation in the simple evolutions with a single population of models (without noise). Parameters** 1 **through** 4 **correspond to** k**,** α_1**,** α_2 **and** α_3**, respectively. The true values are** $k = 1.25$**,** $\alpha_1 = 0.8$**,** $\alpha_2 = 0.4$ **and** $\alpha_3 = 0.2$**. Each box corresponds to 100 evolutionary runs. Note that in order to zoom in on the relevant range, some boxes and outliers are omitted from the plot.**

cal to the sub-algorithms used in the coevolution, including the parameter settings (see Sec. 3.2.3), except for the fitness calculation. Now, in each generation, 100 experiments are performed on the animal using 100 randomly generated intensity patterns. The 100 intensity patterns are used to evaluate a model 100 times, and the average square error between the model's and the animal's speed sequences is used as the model's fitness. Fig. 8 reveals that, like the passive coevolutions, the evolution is able to identify parameters k and α_1, but not α_2 and α_3.

5. CONCLUSIONS

This paper has presented a new platform that allows for the automatic learning of animal behavior without prior information. We have shown that, by allowing classifiers to control the stimuli in the animal's environment, the system is able to correctly identify the parameters of a relatively complex behavior. Classifiers that only passively observe the animal were unable to learn its behavior. Moreover, evolutions that attempted to evolve the behavior using a simple pre-determined comparison metric failed. This suggests that our approach of not relying on such metrics and of allowing the system to learn about the animal through interactions warrants further investigation. One of the limitations of the current setup is the large number of experiments that are performed on the 'real' animal. This can (i) cause the coevolutionary process to take a prohibitively long time to learn the animal's behavior, and (ii) raise ethical issues regarding the animal's well-being. In the proof-of-concept study, both model and 'animal' shared the same structure (yet the classifiers do not have any knowledge about the animal). In the future, we will address these limitations by reducing the number of experiments required, and by using other types of models (e.g. neural networks). Finally, we also intend to perform experiments using robotic animats and real animals.

6. ACKNOWLEDGMENTS

The research work disclosed in this publication is funded by the Marie Curie European Reintegration Grant within the 7th European Community Framework Programme (grant no. PERG07-GA-2010-267354).

M. Gauci acknowledges support by the Strategic Educational Pathways Scholarship (Malta). The scholarship is part-financed by the European Union—European Social Fund (ESF) under Operational Programme II—Cohesion Policy 2007–2013, "Empowering People for More Jobs and a Better Quality of Life".

7. REFERENCES

[1] H.-G. Beyer. *The theory of evolution strategies.* Springer, Berlin, 2001.

[2] J. J. Bolhuis and L.-A. Giraldeau. *The behavior of animals: mechanisms, function, and evolution.* Wiley-Blackwell, USA, 2004.

[3] J. C. Bongard and H. Lipson. Automated robot function recovery after unanticipated failure or environmental change using a minimum of hardware trials. In *Proceedings of the 2004 NASA/DoD Conference on Evolvable Hardware*, pages 169–176. IEEE Computer Society, 2004.

[4] J. C. Bongard and H. Lipson. Nonlinear system identification using coevolution of models and tests. *IEEE Transactions on Evolutionary Computation*, 9(4):361–384, 2005.

[5] J. C. Bongard, V. Zykov, and H. Lipson. Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121, 2006.

[6] J. L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[7] J. Evans and A. Rzhetsky. Machine science. *Science*, 329(5990):399–400, 2010.

[8] I. D. Gauld, M. A. O'Neill, and K. J. Gaston. Driving Miss Daisy: the performance of an automated insect identification system. In *Hymenoptera: evolution, biodiversity and biological control. The Fourth International Hymenoptera Conference*, pages 303–311. CSIRO PUBLISHING, 1999.

[9] L. Grossman. Computer literacy tests: Are you human? *Time Magazine*, June 2008.

[10] R. D. King, J. Rowland, S. G. Oliver, M. Young, et al. The automation of science. *Science*, 324(5923):85–89, 2009.

[11] B. Kouchmeshky, W. Aquino, J. C. Bongard, and H. Lipson. Co-evolutionary algorithm for structural damage identification using minimal physical testing. *International Journal for Numerical Methods in Engineering*, 69(5):1085–1107, 2007.

[12] N. MacLeod, M. Benfield, and P. Culverhouse. Time to automate identification. *Nature*, 467(7312):154–55, 2010.

[13] M. Mirmomeni and W. Punch. Co-evolving data driven models and test data sets with the application to forecast chaotic time series. In *2011 IEEE Congress on Evolutionary Computation*, pages 14–20. Auburn University, New Orleans, LA, 2011.

[14] A. Persidis. High-throughput screening. *Nature biotechnology*, 16(5):488–493, 1998.

[15] J. Schmidhuber. Der ideale Wissenschaftler (*in German*). `http://www.cio.de/karriere/personalfuehrung/803246`, June 2004.

[16] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[17] A. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[18] K. E. Whelan and R. D. King. Intelligent software for laboratory automation. *Trends in Biotechnology*, 22(9):440–445, 2004.

[19] X. Yao, Y. Liu, and G. Lin. Evolutionary programming made faster. *IEEE Transaction on Evolutionary Computation*, 3(2):82–102, 1999.