5 Inferring Individual Behaviors Through Controlled Interaction

5.1 Introduction

In the previous chapters, we demonstrate how *Turing Learning (TL)* can be used for inferring swarm behaviors only through observation. In fact, this is based on an implicit assumption that the behavioral repertoire of agents in the swarm could be fully revealed through observation. From the perspective of system identification, the target system has high observability. However, when the target system has low observability, some hidden information may not be fully revealed only through observation. In this case, the machine needs to interact with the agent in an active way to explore its hidden information. Based on this idea, in this chapter we aim to infer such agent behaviors through *Turing Learning*. In particular, we extend *Turing Learning* in Chapter 3 with interactive ability.

Observation and interaction are widely adapted by ethologists when investigating the behavior of animals [167, 168, 169]. When investigating animals' behavior in their natural habitat, passive observation is preferable as it is difficult to change the environmental stimuli. In this case, inferring the causal relations between the animal's behavior and its environmental stimuli may become challenging, since the stimuli are not under the observer's control. However, when the experiments are carried out in a controlled laboratory (which is the primary case), it is possible to actively change the stimuli to interact with the animals under investigation in a meaningful way. In [169], in order to investigate causes of the dung beetle dance, biologists designed various experiments such as the appearance of disturbance or obstacles to interact with the dung beetle and learn how it adapts to the environmental changes.

In this chapter, we investigate whether a machine could infer the agent behaviors through actively interacting with the agents in an autonomous manner using the *Turing Learning* method. To validate this, two case studies are presented: deterministic agent behavior (Section 5.3.1) and stochastic behavior (Section 5.4.1). In these two case studies, the machine is able to control the agent's environmental conditions, which in this chapter corresponds to the intensity of the ambient light. At the same time, it is capable of simulating the actions of the agent. The learning result is a model of the agent that captures its behavior in relation to the environmental stimulus.

The advantages of our approach are twofold:

- Firstly, it does not rely on a pre-defined metric for gauging the resemblance of models to the agent. Rather, such metrics are implicitly defined by the classifiers, and hence incorporated into the evolutionary process.
- Secondly, the machine learns the agent's behavior by interacting with it, rather than simply observing its behavior in a passive manner. This interaction can help the machine to extract all of the agent's behavioral dynamics, as will be shown in the results section.

This chapter is organized as follows. Section 5.2 describes the methodology, illustrating how the *Turing Learning* method is extended to have interactive ability. The deterministic and stochastic behaviors under investigation are presented as two case studies (Sections 5.3 and 5.4). Section 5.3.1 describes the deterministic behavior. Section 5.3.2 presents the simulation setup. Section 5.3.3 presents the results of inferring the deterministic behavior, including analysis of the evolved models (Section 5.3.3.1), the coevolutionary fitness dynamics (Section 5.3.3.2), analysis of the evolved classifiers (Section 5.3.3.3), the noise study (Section 5.3.3.4), and a comparison of *Turing Learning* with a single-population evolutionary approach (Section 5.3.3.5) and an approach based on coevolution of inputs and models (Section 5.3.3.6). Section 5.4.1 describes the stochastic behavior for the general case using a state machine. Section 5.4.2 presents the simulation setup for inferring the stochastic behavior. Section 5.4.3 and 5.4.4 present the obtained results for the case of 2 states and 3 states, respectively. Section 5.5 summaries the chapter.

5.2 Methodology

In this chapter, we extend *Turing Learning* described in Chapter 3 with interactive ability. The basic idea is the same, that is, the method is comprised of two populations:

one of models, and one of classifiers, which coevolve 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 agent. The fitness of the models depends solely on their ability to mislead the classifiers into making the wrong judgment, that is, classifying them as the agent. In this following, we will describe the implementation that is related to the work in this chapter.

5.2.1 Models

The models are represented by a set of parameters that govern the rules of the agents. The details of these parameters will be described in Section 5.3.1 and Section 5.4.1. As we have argued in the previous chapters, explicit representation (i.e., evolving only the parameters) and knowing the ground truth (i.e., real parameters) makes it feasible for us to objectively gauge the quality of the models obtained.

5.2.2 Classifiers

The structure of the classifiers is similar to the one used in Chapter 3 (see Figure 3.1). The only difference is the classifiers take the environmental stimuli and the agent's response as inputs and the outputs are used to control the environmental stimuli and make judgment. Figure 5.1 shows the structure of the classifiers used in this chapter. In the following, we will explain how the classifiers work.

Suppose the agent responds to the level of light intensity in the environment, and we assume that the classifier could observe the agent's speed corresponding to the light intensity. One of the inputs to the classifier (network) is the light intensity in the environment at time step t, $I^{(t)} \in [0, 1]$, and the other input is the speed $v^{(t)}$. In order to make this setup more feasible to implement, it is assumed that the system cannot directly measure the speed of the individual, but rather its position. The speed of the individual for the classifier's input is then calculated by subtracting the previous estimated position from the current estimated position, and dividing the resulting number by the time interval between two measurements.

In order to make a judgment between a model and the agent, the classifier observes the behavior (speed) over a period of time. In addition, the classifier is also in control of

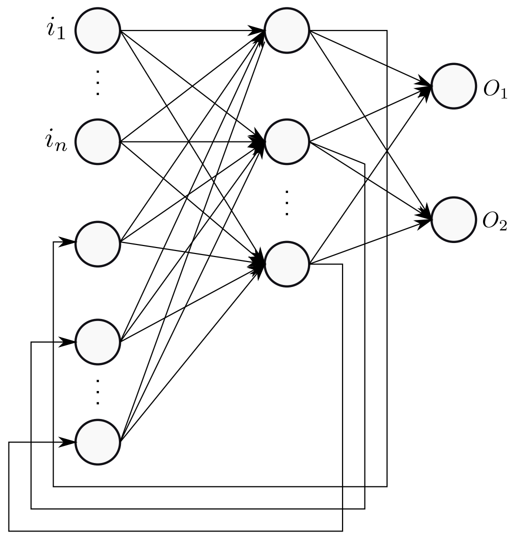


Figure 5.1: This diagram shows the structure of the classifiers used in this chapter. It is a recurrent Elman neural network [151] with i input neurons, h hidden neurons, and two output neurons (O_1 and O_2). O_1, which controls the stimulus, is fed back into the input; O_2 is used for making a judgment. Two bias neurons with a constant input of 1.0 are connected to each neuron of the hidden and output layers. See text for details.

the light intensity in the individual'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 $v^{(0)}$. The value of the light intensity for the next time step is obtained from the classifier's output neuron O_1, and the process repeats. After having iterated through all the time steps (a single trial), the final value of output neuron O_2 is used to make a judgment: the network decides on a model if $O_2 < 0.5$, and on the agent if $O_2 \geq 0.5$. The memory (value of hidden neurons) of the classifiers is reset at the end of every trial.

5.2.3 Optimization Algorithm

The algorithm used here is based on a $(\mu + \lambda)$ evolution strategy with self-adaptive mutation strengths [153, 154]. It is the same as the one used in Chapter 3. For the details of the implementation, see Section 3.1.1.3.

5.2.4 Fitness Calculation

Suppose the population sizes for the model and classifier are M and C, respectively. The fitness of each model is obtained by evaluating it with each of the classifiers in the competing population (C in total). For every classifier that wrongly judges the model as being the agent, the model's fitness increases by $\frac{1}{C}$. The final fitness is in $[0, 1]$.

The fitness of each classifier is obtained by using it to evaluate (i) each model in the competing population (M in total) once, and (ii) the agent L times with different initial light intensities. For each correct judgment of the model and the agent, the classifier's fitness increases by $\frac{1}{2 \cdot M}$ and $\frac{1}{2 \cdot L}$, respectively. The final fitness is in $[0, 1]$.

5.3 Case Study One

The behaviors to be identified in this chapter were chosen to serve as a tractable test-bed for proof-of-concept study. While it may loosely correspond to how some animals react to the stimuli in their environment, it is not intended to mimic any specific animal. In these behaviors, non-trivial interaction with the agent is critical for leading the agent to reveal all of its behavioral repertoire.

5.3.1 Deterministic Behavior

We simulate a one-dimensional environment in continuous space. The simulation advances in discrete time steps $t \in \{0, 1, 2, \ldots\}$. The (ambient) light intensity in the environment, I, can be varied continuously between 0 and 1. The agent distinguishes between three levels of light intensity, low ($0 \leq I < I_L$), medium ($I_L \leq I \leq I_H$), and high ($I_H < I \leq 1$). Hereafter, these levels will be referred to as L, M, and H.

If the light intensity is at level M at time t, the speed of the agent, $s^{(t)} \in \mathbb{R}$, varies linearly with $I^{(t)}$ as:

$$s^{(t)} = k\left(I^{(t)} - 0.5\right), \tag{5.1}$$

where k is a constant.

We define two constants: $c_1 = k\,(I_H - 0.5)$ and $c_2 = k\,(I_L - 0.5)$. The deterministic behavior under investigation is shown in Figure 5.2. The agent's behaviors for levels L

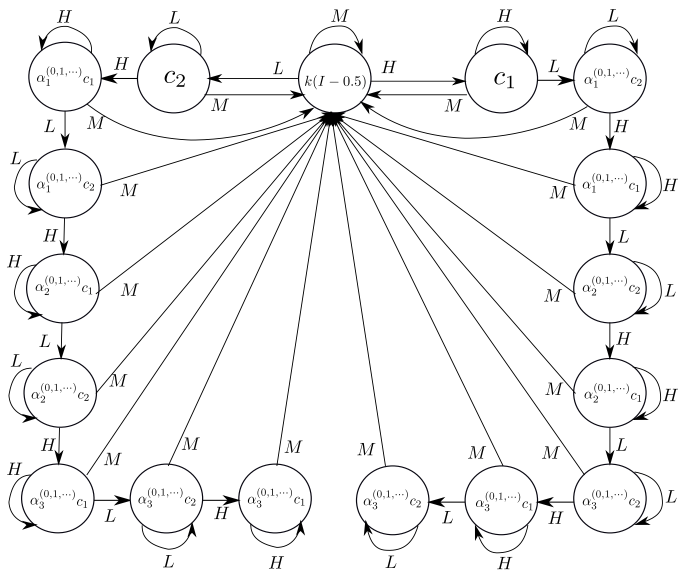

$k(I-0.5)$
c_1
c_2
$\alpha_1^{(0,1,\cdots)}c_1$
$\alpha_1^{(0,1,\cdots)}c_2$
$\alpha_2^{(0,1,\cdots)}c_1$
$\alpha_2^{(0,1,\cdots)}c_2$
$\alpha_3^{(0,1,\cdots)}c_1$
$\alpha_3^{(0,1,\cdots)}c_2$
L
M
H

Figure 5.2: The deterministic behavior under investigation. It shows how the agent responses to the level of light intensity (L, M and H) in its environment. Each state represents the agent's speed. See text for details.

Table 5.1: This table shows the change of the agent's speed (shown in Figure 5.2), for an example sequence of light levels.

level	M	H	L	H	L	L	L	H	H	L	L	L	H	H	L	L	H	H	H	M	H	L	L
speed	$k(I-0.5)$	c_1	c_2	c_1	c_2	$\alpha_1^1c_2$	$\alpha_1^2c_2$	c_1	$\alpha_1^1c_1$	c_2	$\alpha_2^1c_2$	$\alpha_2^2c_2$	c_1	$\alpha_2^1c_1$	c_2	$\alpha_3^1c_2$	c_1	$\alpha_3^1c_1$	$\alpha_3^2c_1$	$k(I-0.5)$	c_1	c_2	$\alpha_1^1c_2$

and H depend on the previous levels of light intensity (i.e. the agent has memory). The two behaviors are symmetrical to each other. Here, we will describe the behavior for level L; the behavior for level H is obtained by exchanging L with H and I_L with I_H in the following description.

When the light intensity is at level L, the agent's default speed is $k\,(I_L - 0.5)$ and remains at that value as long as the light intensity remains at level L. If the light intensity is at level H (for any number of time steps), and then immediately changes to level L, and remains at that level for at least one more time step, then the agent's speed decays exponentially with a rate of α_1: that is, in the first time step that the light intensity is at level L, the agent's speed is $\alpha_1^0 k\,(I_L - 0.5)$; it then changes to $\alpha_1^1 k\,(I_L - 0.5)$, $\alpha_1^2 k\,(I_L - 0.5)$, and so on as long as the light intensity remains at level L. The agent now registers that α_1 has been activated. If another $H \to L \to L$ sequence is observed, the agent's speed now decays exponentially with a rate of α_2. If further $H \to L \to L$ sequences are observed, the exponential decay rate becomes and remains at α_3. Note that at any time, if the agent observes a light intensity at level M, its speed is proportional to the light intensity, as shown in Eq. 5.1, and it forgets all its past observations of the light intensity (i.e., the memory of the agent is reset).

The behavior of the agent can thus be represented by five cases; one where the agent's response to the light intensity is *proportional* (which occurs whenever the light intensity is at level M); one where the agent's response is *constant* (i.e. the agent's speed reaches the lower and upper saturation values (c_1 or c_2) as shown in Figure 5.2); and three where the agent's response *decays exponentially* with the decay rates α_1, α_2 and α_3, respectively.

Table 5.1 shows an example sequence of light levels, along with the corresponding speed of the agent (i.e., the speed shown in Figure 5.2).

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 decay rates are set to: $\alpha_1 = 0.8$, $\alpha_2 = 0.4$, $\alpha_3 = 0.2$. Thus, in each case, the agent's speed decays exponentially towards zero. Note that these values (k, α_1, α_2 and α_3) are chosen arbitrarily and the coevolutionary algorithm is not sensitive to them.

The system identification task is to learn these four parameters (k, α_1, α_2 and α_3) of the agent. It is worth while to mention again that to learn α_3, the machine needs to at least output the sequence: $H \to L \to L \to H \to L \to L \to H \to L \to L$ or $L \to H \to H \to L \to H \to H \to L \to H \to H$. Since I_L and I_H are set to a relatively small and large value in $[0, 1]$ respectively, it is very unlikely to randomly generate such sequences.

5.3.2 Simulation Setup

We use three setups for the *Turing Learning* method. The setup, in which the classifier is in control of the light intensity in the agent's environment, is hereafter referred to as

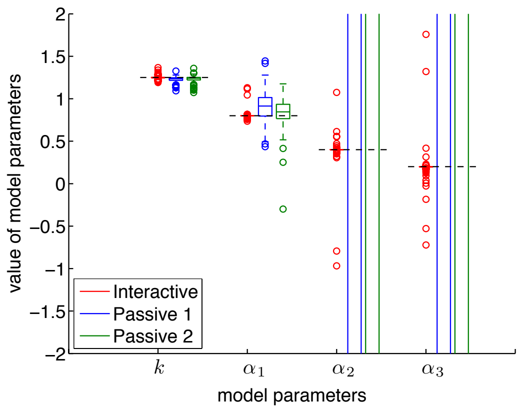


Figure 5.3: This plot shows the distributions of the evolved models with the highest subjective fitness in the 1000th generation in the coevolutions. Each box corresponds to 100 coevolution runs. The dotted lines correspond to the values of the four parameters that the system is expected to learn (i.e. those of the agent). From top to bottom, these are 1.25, 0.8, 0.4, 0.2, respectively. Note that in order to zoom in on the relevant range, some boxes and outliers are omitted from the plot.

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 agent in a passive manner; that is, it does not control the light intensity in the environment. We considered two such setups: in the first setup (hereafter, "Passive 1") the light intensity is randomly chosen from the uniform distribution in $[0, 1]$, in every time step. In the second setup (hereafter, "Passive 2"), the light intensity is randomly chosen only after certain number of time steps (in this setup the number is chosen to be 10). All other aspects of these two setups are identical to the "Interactive" setup.

The population sizes of the models and classifiers are chosen to be 100, respectively. We performed 100 coevolution runs for each setup. Each coevolution run lasts 1000 generation. In one generation, each classifier conducts 100 trials on the agent. In each trial, the classifier observes the agent for 10 s at 0.1 s intervals, that is, a total of 100 data points.

5.3.3 Results

5.3.3.1 Analysis of Evolved Models

Figure 5.3 shows a box plot with the distributions of the evolved models with the highest subjective fitness in the 1000th generation over 100 coevolution runs of the three setups. The passive coevolutions are able to evolve the parameters k and α_1 with a reasonable accuracy; however, they are not able to evolve α_2 and α_3. In the Passive 1 coevolution, the relative errors of the medians of the four evolved parameters $(k, \alpha_1, \alpha_2, \alpha_3)$ with respect to those of the agent are 1.2%, 14.3%, 7.8×10^4%, and 2.3×10^5%, respectively. The Passive 2 coevolution leads to similarly large relative errors in the evolved values of α_2 and α_3. This phenomenon can be explained as follows. If the light intensity changes randomly (either every time step, or every ten time steps), it is unlikely that the $H \rightarrow L \rightarrow L$ and/or $L \rightarrow H \rightarrow H$ sequences will occur enough times, without a level of M in between, such that the classifiers can observe the effects of α_2 and α_3. Therefore, the classifiers do not evolve the ability to distinguish the behavior of models from the behavior of the agent with respect to these two parameters, and in turn, these parameters do not converge to their true value in the model population.

In contrast to the passive coevolutions, the Interactive coevolution is able to evolve all the four parameters with a good accuracy. The relative median errors are 0.024%, 0%, 0.025% and 0.15% for k, α_1, α_2 and α_3 respectively. This implies that by the 1000th 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 agent based on the effect of any of the four parameters. Therefore, in order to compete for being selected in the population, the models are evolved to behave like the agent in every aspect.

5.3.3.2 Coevolutionary Dynamics

Figure 5.4 shows the dynamics of the coevolutionary algorithms. The horizontal axis shows the generation, whereas the vertical axis shows the total square error of the model parameters, that is, the sum of the square errors in the four parameters (of the model with the highest subjective fitness in each generation) with respect to their true values. In the case of the Interactive coevolution, the median error starts to reduce after around the 100th generation, and keeps decreasing until the last generation where it reaches a

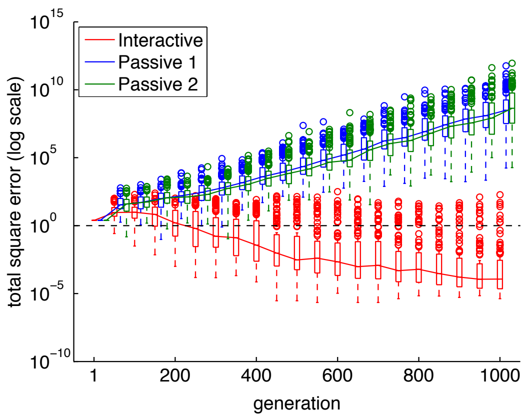


Figure 5.4: This plot shows the total square errors of the evolved model parameters compared to those of the agent over generations. The models with the highest subjective fitness in each generation are selected. Each box corresponds to 100 coevolution runs, and the solid lines correspond to the median error.

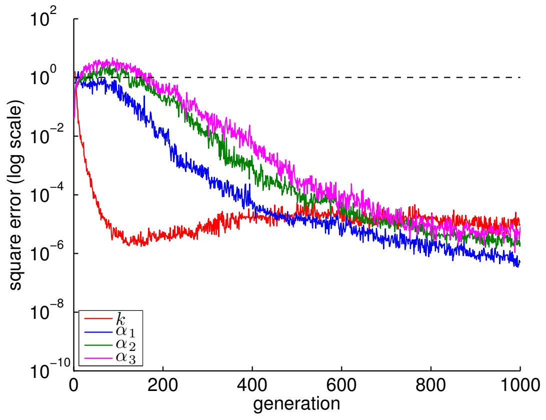
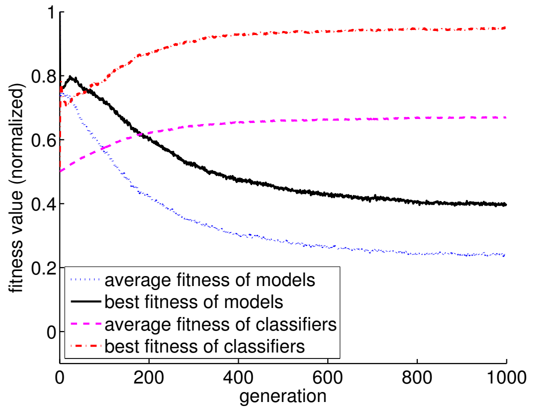


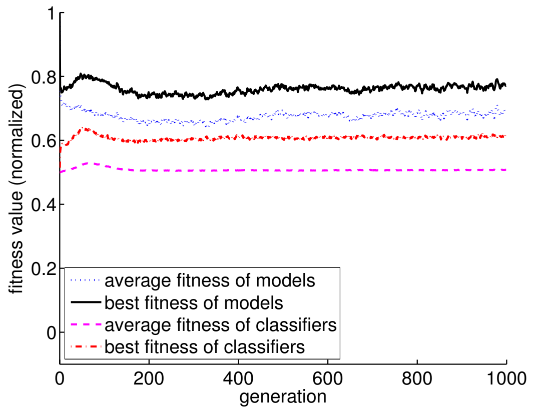
Figure 5.5: This plot shows how the square error in the individual model parameters changes over the generations in the Interactive coevolution. The curves correspond to median values from 100 coevolution runs.

value of 10^{-4}. In contrast, in the case of the passive coevolutions, not only does the median error not decrease, but it increases to a value of 10^{8} by the 1000th generation.

We now analyze how the four individual parameters evolve during the course of the Interactive coevolution, which is the only fully-successful setup. The plot shown in Figure 5.5 reveals how the learning proceeds in the coevolution. Parameter k is the first to be learnt, followed by α_1, while parameters α_2 and α_3 take a longer time to approximate



Interactive coevolution



Passive 1 coevolution

Figure 5.6: This plot shows the subjective fitness (normalized) 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 coevolution runs.

the true values. This means that the classifiers first learn to distinguish models from the agent on the basis of k and α_1. This ability of the classifiers drives the model population to evolve k and α_1, in order to mislead the classifiers. Eventually, the classifiers also learn to exploit the effects of α_2 and α_3 in order to make the right judgment; thereby driving the model population to evolve these two parameters accurately. After about the 600$^{\text{th}}$ generation, the learning of the four parameters proceeds with approximately identical rates.

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 Section 5.2.4) during the course of the coevolution. As both of the passive coevolutions fail to converge, we present the analysis of fitness dynamics only for Passive 1 coevolution (the other case was found to have similar dynamics). Figure 5.6 shows the fitness dynamics of the Interactive and the Passive 1 coevolutions. In the case of the Interactive coevolution (see Figure 5.6(a)), the average fitness of the classifiers starts off at 0.5, which means that the classifiers make judgments that are no better than random judgments. However, the classifiers quickly improve in fitness, which in turn causes the fitness of the models to decrease. This increases the selective pressure on the models. After about 600 generations both the fitness of classifiers and models reach a steady state, which according to Figure 5.5 corresponds to the region where the four parameters evolve with virtually identical rates. In the case of the Passive 1 coevolution (see Figure 5.6(b)), the average fitness of the classifiers also starts off at 0.5. In the first few generations, this increases slightly, because the classifiers learn how to distinguish models from the agent 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 unlikely to have the opportunity to observe the effects of α_2 and α_3, 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, therefore leading the parameters α_2 and α_3 to drift, the effect of which can be seen in Figure 5.4.

5.3.3.3 Analysis of Evolved Classifiers

In this section, we analyze the evolved classifiers in the Interactive coevolution. The model described in Section 5.2.1 is defined by four parameters: $\{k, \alpha_1, \alpha_2, \alpha_3\}$. In order to evaluate the quality of the evolved classifiers we performed a grid search over the space of the amount of disturbance (i.e. noise) injected into each of the four parameters. We used 11 noise magnitude settings per parameter, $M \in \{0, 0.1, \dots, 1\}$, with the noise being added to the parameters as follows:

$$p' = p(1 + \mathcal{U}(-M, M)), \tag{5.2}$$

where $p \in \{k, \alpha_1, \alpha_2, \alpha_3\}$ represents any of the four parameters, and $\mathcal{U}(-M, M)$ denotes a uniform distribution on the interval $(-M, M)$. Note that $M = 0$ corresponds to no noise, whereas $M = 1$ means that the noisy value of the parameter can be between 0 and twice its actual value.

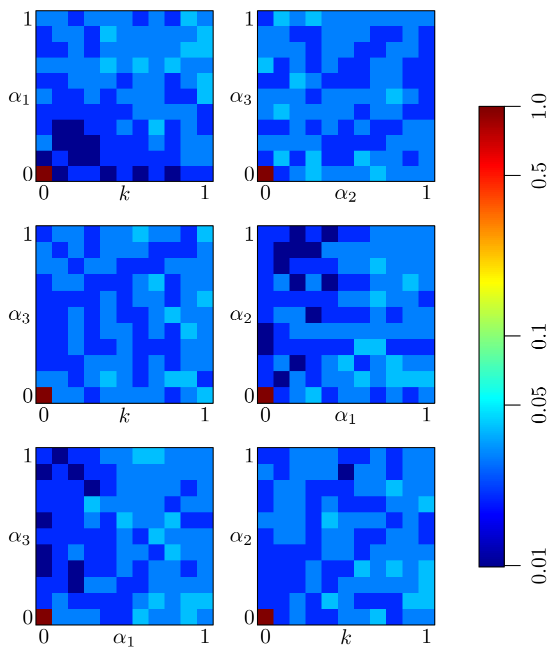


Figure 5.7: This figure shows the landscape of U^* (log scale) for the overall best classifier over the six sub-spaces with two parameters as degrees of freedom (from 100 coevolution runs). Each axis in each plot ranges between 0 and 1, corresponding to the minimum and maximum magnitude of noise added into each parameter respectively. See text for details.

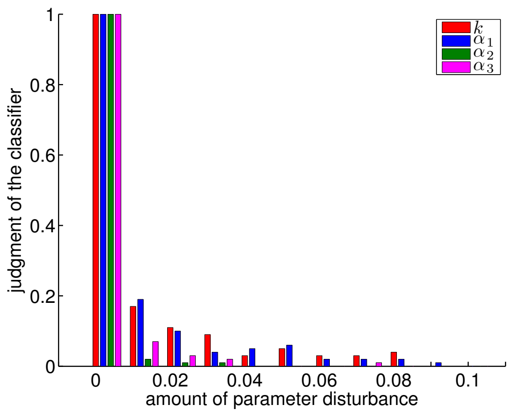


Figure 5.8: This plot shows the average judgment [U, see Eq. 5.3] of the best evolved classifier over 100 trials when the magnitude of noise added into each parameter of the agent is within 0.1.

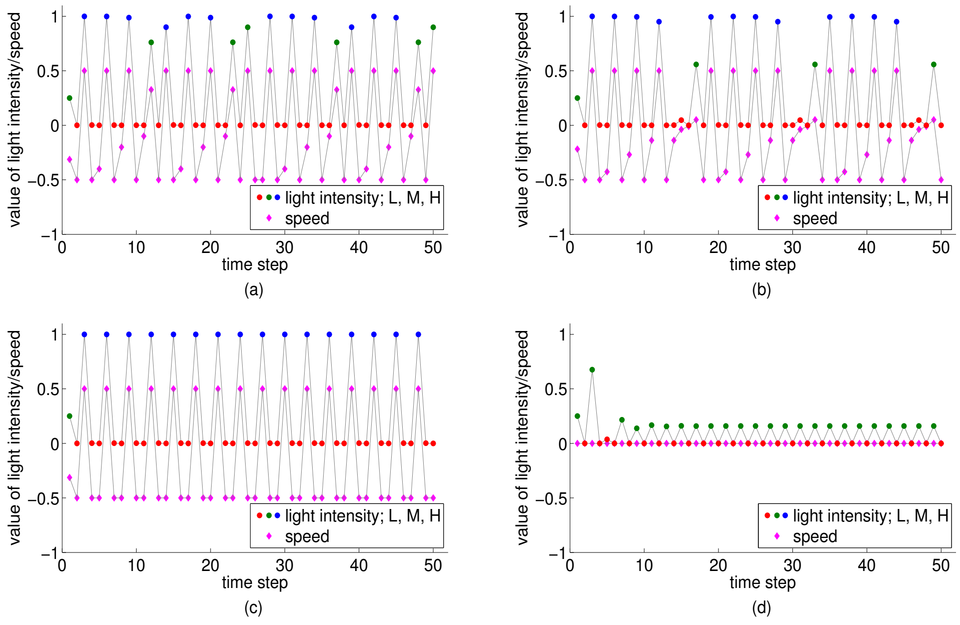


Figure 5.9: This plot shows the light intensity sequences as output by the overall best classifier (circular points), along with the corresponding speeds of the agents (diamond-shaped points) in four trials conducted on different agents: (a) the agent; (b) a model similar to the agent, generated randomly according to Eq. 5.2 with $M = 0.5$; (c) a model whose speed is linear to the light intensity; and (d) a static model, whose speed is always 0 (i.e. $k = 0$). The colors of the circular points (red, green, blue) correspond to light intensities at levels L, M and H, respectively.

For the sake of simplicity, we only analyzed the classifiers with the highest subjective fitness in the last generation of the 100 coevolutions. For each of the 100 classifiers, and for each combination of noise magnitudes, we conducted 100 trials. In other words, we conducted $100 \cdot 11^4 \cdot 100 = 146,410,000$ trials in total. In each trial, we modulated the agent's parameters according to Eq. 5.2. Each trial was run for T = 100 time steps (the same setting used within the coevolutions). For each combination of magnitudes employed, the overall performance U was computed as the sum of the final judgments of the classifier in each trial, divided by the total number of trials, N:

$$U = \sum_{i=1}^{N} J_i/N, \tag{5.3}$$

where, $J_i \in \{0,1\}$ is the final judgment of the classifier in trial i, and N is the total number of trials conducted. Note that $J_i = 0$ and $J_i = 1$ imply that the classifier has judged the behavior as being that of a model and the agent, respectively.

In order to find the overall best classifier from the 100 classifiers analyzed, we defined the following metric:

$$\begin{aligned} W = {} & \left[1 - U\left(0,0,0,0\right)\right] \\ & + \frac{1}{\Omega} \underbrace{\sum\sum\sum\sum}_{\substack{i,j,k,\ell \in \{0,0.1,\ldots,1\} \\ i^2+j^2+k^2+\ell^2 \neq 0}} \left(i+j+k+\ell\right) U\left(i,j,k,\ell\right), \end{aligned} \tag{5.4}$$

where $\Omega = 29282$ is the maximum value that the quadruple sum can achieve (i.e. if all the U's are equal to 1). The first term in Eq. 5.4 penalizes the classifier if $U < 1$ when there is no noise on the parameters; they are thus undisturbed and identical to the ones of the agent. The second term penalizes the classifier if $U > 0$ for any non-zero noise combination, with increasing penalties being applied to higher noise magnitudes. The normalization of the second term by Ω serves to make the two terms contribute equally to W, which can take values in $[0,2]$. Note that the minimum value of $W = 0$ can only be achieved by the perfect classifier, that is, one that outputs 1 if $i = j = k = \ell = 0$ and 0 otherwise. In our case, the best classifier achieved a value of $W = 3.7 \cdot 10^{-3}$.

The performance landscape of the models is 5-dimensional (4 model parameters plus performance measure), and cannot be visualized directly. Therefore, we considered each combination of two model parameters ($\binom{4}{2} = 6$ combinations) as a sub-space, and for each point in this sub-space, we calculated the performance measure as the maximum value over the sub-space spanned by the remaining two parameters. For instance, on the sub-space (k, α_1), the performance measure $U^*(k, \alpha_1)$ was calculated as:

$$U^*\left(k,\alpha_1\right) = \max_{\alpha_2,\alpha_3} U\left(k,\alpha_1,\alpha_2,\alpha_3\right). \tag{5.5}$$

Note that for all the points except $(0,0,0,0)$, Eq. 5.5 corresponds to the worst-case scenario for the classifiers, because the value of U for the ideal classifier at these points is 0. For the point $(0,0,0,0)$, the value of U for the ideal classifier is 1.

Figure 5.7 shows the landscape of U^* (log scale) for the overall best classifier over the six sub-spaces. When interacting with the agent (i.e. $i = j = k = \ell = 0$), the output of the classifier is always 1. This corresponds to the point $U^*(0,0)$ in the six sub-spaces of Figure 5.7 (here only the maximum is shown). When interacting with the models (i.e. when the parameters of the agent are perturbed), for any combination of noise magnitudes, the average output of the classifier is below 0.05 for the six sub-spaces.

In order to further analyze how sensitive the overall best classifier is when the parameters of the agent are only slightly perturbed, we set the maximum magnitude of noise added to each parameter to 0.1, and used a resolution of 0.01. In this evaluation, we only injected noise into one parameter at a time. For each noise magnitude, the classifier was evaluated in 100 trials (with different initial light intensities, and randomly-generated noise), and the average performance measure was computed according to Eq. 5.3.

Figure 5.8 shows the average judgment [U, see Eq. 5.3] of the best classifier, which was evaluated in 100 trials. With increasing noise magnitudes, the average judgment of the classifier approaches zero, which means that the classifier can identify the agents as models more consistently. The classifier has different sensitivities to the four parameters; the effects of noise on α_2 and α_3 on its judgment are higher than those of k and α_1. For instance, in the case of α_2, even with $M = 0.01$, the classifier correctly judges the behavior as being that of a model in 98% of the trials. As we have seen, k and α_1 are the first two parameters to be identified in the coevolution. α_2 and α_3 are addressed in the later generations of the coevolutionary process, and it seems that the classifier is more sensitive to these two parameters.

We now investigate how the overall best classifier interacts with the agents. We conducted four trials with four different agents: (a) the agent; (b) a model similar to the agent, generated randomly according to Eq. 5.2 with $M = 0.5$; (c) a model whose speed is linear to the light intensity without exponential decay (i.e. $k = 1.25$, $\alpha_1 = \alpha_2 = \alpha_3 = 1$); and (d) a static model, whose speed is always 0 (i.e. $k = 0$). In each trial, the initial light intensity was set to 0.25. The trials lasted for 100 time steps.

Figure 5.9 shows the sequences of light intensity output by the classifier, along with the speed of the agents in the four trials, for the first 50 time steps (we observed that the last 50 time steps constitute a repetition of the first 50). As we can see, the classifier outputs different sequences of light intensity in order to interact with the different agents. The greater the difference between a model and the agent, the more varied is the sequence of light intensities that the classifier outputs when interacting with it as compared to the one it outputs when interacting with the agent. For the agent, the classifier repeatedly outputs a $H \rightarrow L \rightarrow L$ sequence, in order to fully reveal the behavior (this is analyzed in detail in the following paragraph). For the model that is similar to the agent (see Figure 5.9(b)), the sequence of light intensities is similar to the one produced for the agent, but it is not identical. Interestingly, for the model without exponential decay (see Figure 5.9(c)), the classifier produces the $H \rightarrow L \rightarrow L$ sequence even more often than it does for the agent, but it does not set the light intensity to level M. For the static model (see Figure 5.9(d)), the classifier never outputs a $H \rightarrow L \rightarrow L$ sequence; instead, it outputs a sequence that alternates between M and L.

We now focus on analyzing the behavior of the agent in Figure 5.9(a). Initially, the classifier outputs the sequence $L \rightarrow H \rightarrow L \rightarrow L \rightarrow H \rightarrow L \rightarrow L \rightarrow H \rightarrow L \rightarrow L \rightarrow M$, which means that by the 12th time step, it has already observed the effects of all four parameters. Interestingly, the classifier then (essentially) repeats this sequence and thus observes the effect of all the parameters for another seven times (four repetitions occur in the last 50 time steps, which are not shown in Figure 5.9(a)). We conjecture that the repetitions make the classifier more robust in determining whether the behavior under observation is that of a model or the agent. These results suggest that the classifier succeeds in controlling the level of ambient light in a way that helps distinguish between the agent and the models, revealing all aspects of the underlying behavior. Note that the classifiers are not provided with any prior information about the agent, including its structure. Thus, all knowledge is gained through an evolutionary process that is driven by the accuracy of their judgments.

5.3.3.4 Noise Study

We now consider the situation where, during the coevolutionary process, noise is injected into the agent's behavior, and the agent's positions as measured by the system. This makes the overall setup more realistic as the locomotion of agents and any tracking system will be affected by noise and measurement error. Since the passive coevolutions fail even in the noiseless case, we consider only the Interactive coevolution for the sake of simplicity. We performed 100 coevolutionary runs with the following settings. The light intensity perceived by the agent at time t is obtained by multiplying the actual 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 agent by multiplying the original speed with a random number generated uniformly in $(0.95, 1.05)$. Noise on the estimated position on the agent is applied by adding a random number generated from a normal distribution: $\mathcal{N}(0, 0.005)$.

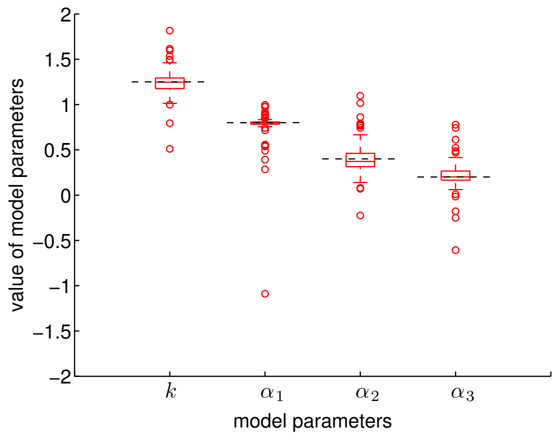


Figure 5.10: This plot shows the distributions of the evolved models with the highest subjective fitness in the 1000th generation of the Interactive coevolution with noise (for a comparison to the case without noise, see Figure 5.3). The dotted lines correspond to the values of the four parameters that the system is expected to learn (i.e. those of the agent). From top to bottom, these are 1.25, 0.8, 0.4, 0.2, respectively.

Figure 5.10 shows a box plot with the distributions of the evolved models with the highest subjective fitness in the 1000th generation of the Interactive coevolution with noise. 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 learn k and α_1.

5.3.3.5 Using a Single-Population Evolutionary Algorithm

In order to compare *Turing Learning* against a more traditional approach, we used a simple evolution where a single population of models evolves. We call this method SPEA. 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 agent's environment. The structure of the evolution is identical to the sub-algorithms used in the coevolution, except for the fitness evaluation step. Now, in each generation, 100 experiments are performed on the agent using 100 randomly generated intensity patterns. The 100 intensity patterns are used to evaluate a model 100 times. The average square error between the model's and the agent's speed

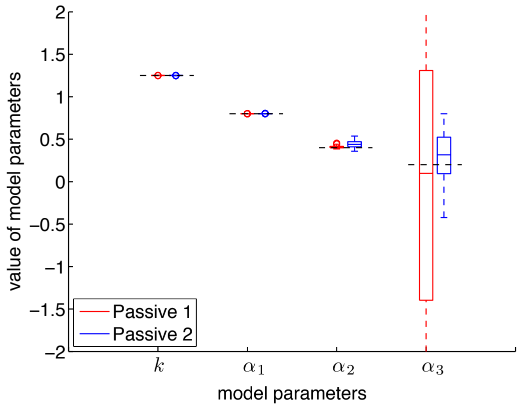


Figure 5.11: This plots shows the distributions of the evolved models with the highest fitness in the 100000th generation in the simple evolutions with a single population. Each box corresponds to 100 evolution runs. The dotted lines correspond to the values of the four parameters that the system is expected to learn (i.e. those of the agent). From top to bottom, these are 1.25, 0.8, 0.4, 0.2, respectively. Note that in order to zoom in on the relevant range, some boxes and outliers are omitted from the plot.

Table 5.2: This table shows a comparison of all approaches. The numbers show the relative errors of the evolved parameters (median values over 100 runs) with respect to the parameters of the agent (in absolute percentage).

	k	α_1	α_2	α_3
Interactive (TL)	0.024	0	0.025	0.15
Passive 1 (TL)	1.2	14.3	7.8×10^4	2.3×10^5
Passive 2 (TL)	0.7	5.74	1.3×10^4	3.3×10^5
Passive 1 (SPEA)	0	0	1.2	48.7
Passive 2 (SPEA)	0	0	9.8	48.5
CoEAIM	4.0×10^{-5}	4.9×10^{-5}	1.1×10^{-3}	2.5×10^{-4}

sequences is used as the model's fitness. Each evolutionary run lasts $100,000$ generations. In other words, the number and duration of experiments on the agent is kept the same as that in the coevolutionary approach, as outlined in Section **??**.

Figure 5.11 reveals that the evolution is able to identify parameters k, α_1, α_2, but not α_3. Note that, apart from the single-population evolution not being able to consistently identify the parameter α_3, they also rely on a pre-defined metric for their operation; in this case, computed as the square error between the model's and the agent's speed

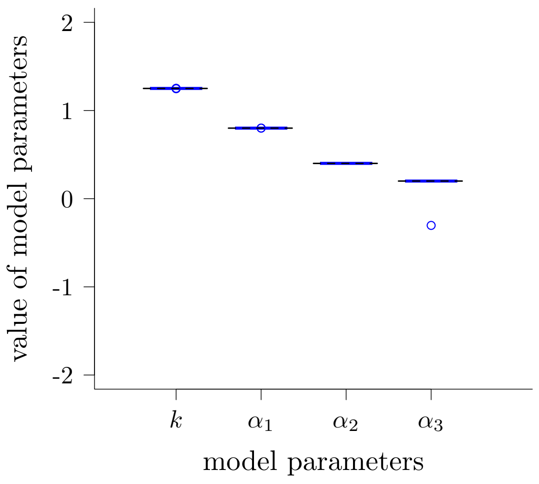


Figure 5.12: This plots shows the distributions of the evolved models with the highest fitness in the 100000th generation in the simple evolutions with a single population. Each box corresponds to 100 coevolution runs. The dotted lines correspond to the values of the four parameters that the system is expected to learn (i.e. those of the agent). From top to bottom, these are 1.25, 0.8, 0.4, 0.2, respectively.

sequences.

5.3.3.6 Coevolution of Inputs and Models

In this section, we compare *Turing Learning* with another coevolutionary system identification method, in which inputs and models competitively coevolve. In particular, we use the classifiers to generate sequences of inputs (in this case, light intensity). The models are optimized through minimizing the square error of speed between the agent and models, given the same sequence of inputs generated by the classifiers. The classifiers compete with the models through generating inputs that cause the maximum disagreement between the model's prediction and the agent's output, which is similar to the concept in [14]. Note that in this case the classifiers are only used for generating the inputs and they do not make judgments. We call this method CoEAIM. Figure 5.12 shows the results of CoEAIM. As we can see, the model parameters are also identified with a high accuracy. In other words, there is no 'true' interaction between the agent

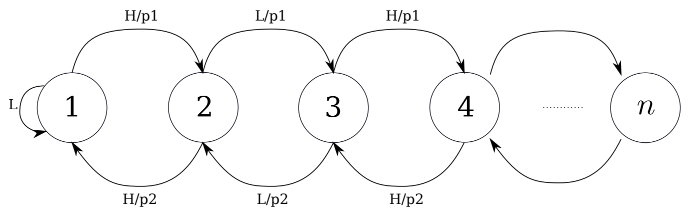


Figure 5.13: A state machine which represents the stochastic behavior of the agent under investigation. The initial state is 1. p_1, p_2 are probabilities. The ambient light intensity that the agent responds to has two levels: H and L. See text for details.

and classifiers during the experiments in *Turing Learning* when investigating the deterministic behavior. The classifiers only need to learn how to generate a fixed sequence of inputs to extract all the information from the agent. For a comparison of all approaches, see Table 5.2.

In order to further validate and highlight the benefit of *Turing Learning*, we investigate the stochastic behaviors in the following section.

5.4 Case Study Two

5.4.1 Stochastic Behavior

Stochastic behaviors are widely observed in the animal kingdom. Given the same stimuli, the animal may behave differently. In order to make the animal under investigation reveal all its behavioral repertoire, ethologists sometimes need to interact with the animals in real time and change the stimuli dynamically according to the animal's response. Based on this motivation, in this section we apply *Turing Learning* to infer stochastic behaviors and investigate whether a machine could interact with the agent through dynamically changing the stimulus rather than generating a fixed sequence of inputs.

We will describe the stochastic behavior for the general case. It is represented as a state machine shown in Figure 5.13. Suppose the agent has n states. The agent's behavior depends on the level of the light intensity in the environment and its current state (i.e., the agent has short-term memory). For the initial state (1), if the light intensity is in low level (L), the agent keeps staying in state 1; if the light intensity is in high level

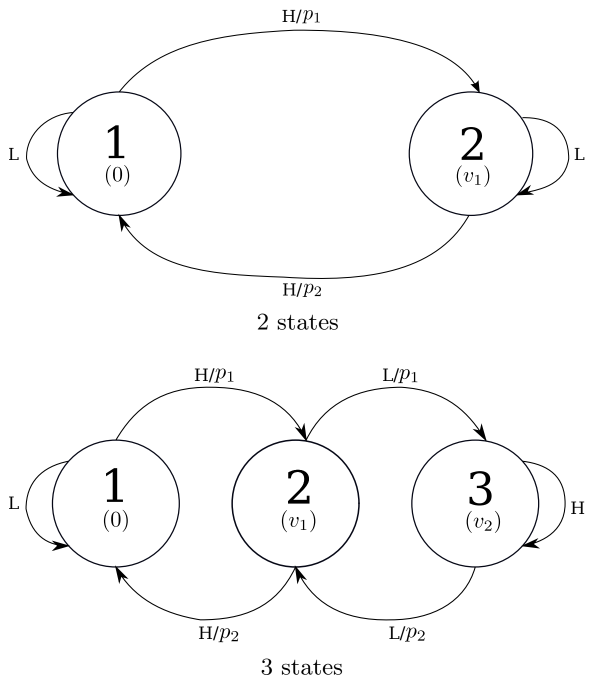


Figure 5.14: The stochastic behaviors with (a) 2 states and (b) 3 states under invesitgation. p_1 and p_2 are probabilities. For all the other cases (e.g., $H/(1-p_1)$), the agent keeps staying in the same state.

(H), the agent has the probability of p_1 to move forward to state 2. For the middle state with even (odd) number, if the level of light intensity is L (H), the agent has the probability of p_1 to move forward to another state with higher number; otherwise if the light intensity is H (L), it has the probability of p_2 to move back to a previous state with lower number. When the agent is in the final state, n, its behavior is similar to that in the middle state. The only difference is the agent never moves forward. Figure 5.14(a) and Figure 5.14(b) show the agent behavior with 2 and 3 states, respectively. These are the two behaviors to be investigated in the thesis.

In the agent's behavior, p_1 is selected to a low value and p_2 is set to a high value. As a consequence of this choice, the agent has a low chance of moving forward to a state with a higher number and thus the higher state has lower observability. The classifiers need to learn how to interact with the agent to infer all its behavioral repertoire.

For a proof-of-concept study, we assume that the agent moves in one-dimensional space and moves at a consistent speed in each state. Suppose the agent stays static and its initial state is known. The system identification task is to identify the parameters of the agent's other states (i.e., v_1 and v_2 shown in Figure 5.14) and the two probabilities, p_1 and p_2. The speed of the agent in each state is chosen arbitrarily. p_1 and p_2 are chosen to have a value of 0.1 and 1.0. This makes the agent's state with higher number has lower observability.

In order to make the agent stay in the state with higher number as long as possible, the classifiers need to capture the point when the agent starts to move forward from a state with a lower number to a state with a higher number and switches the level of light intensity immediately. This makes it easier for *Turing Learning* to learn the behavior as the state can be observed for a sufficient time. If the classifiers fail to do that, the agent would immediately move to the previous state with lower number.

For the 2-state machine shown in Figure 5.14(a), v_1 is selected to be 0.5; for the 3-state machine shown in Figure 5.14(b), v_1 and v_2 are selected to be 0.5 and 1.0, respectively. Therefore, the parameters to be identified for the 2-state and 3-state agent behavior are:

$$\mathbf{q}_1 = (v, p_1, p_2) = (0.5, 0.1, 1.0). \tag{5.6}$$

$$\mathbf{q}_2 = (v_1, v_2, p_1, p_2) = (0.5, 1.0, 0.1, 1.0). \tag{5.7}$$

5.4.2 Simulation Setup

For *Turing Learning*, we still used three setups: "Interactive", "Passive 1" and "Passive 2", as discussed in Section 5.3.2. We also compared *Turing Learning* with the two metric-based methods (SPEA and CoEAIM) described in Section 5.3.3.5 and Section 5.3.3.6, respectively. Note that for each setup, we added a certain amount of noise into the measurement of speed. This is realized by multiplying the agent's real speed with a random value in $[0.95, 1.05]$ in each time step.

5.4.3 Results: Two States

In this section, we compare and analyze the results obtained using the methods discussed in the previous section (5.4.2) for inferring the 2-state agent behavior shown in Figure 5.14(a).

Table 5.3: This table shows a comparison of all approaches for learning the 2-state stochastic behavior shown in Figure 5.14(a). The values show means of AEs (defined in Equation 3.9) of each evolved model parameter with respect to that of the agent.

	v_1	p_1	p_2
Interactive (TL)	0.003	0.03	1.6×10^{-5}
Passive 1 (TL)	0.01	0.03	1.0×10^{-5}
Passive 2 (TL)	0.02	0.02	1.0×10^{-5}
Passive 1 (SPEA)	0.47	0.9	1.0
Passive 2 (SPEA)	0.47	0.9	1.0
CoEAIM	0.47	0.9	1.0

5.4.3.1 Analysis of Evolved Models

Figure 5.15 shows a box plot with the parameters of the evolved models with the highest subjective fitness in the 1000th generation for (a) *Turing Learning*, (b) SPEA, and (c) CoEAIM. Using *Turing Learning*, the system identified all parameters of the agent with good accuracy. For the other two metric-based methods, all the three parameters are not learned well. Instead, the three evolved parameters converge into three different values: $v_1 \rightarrow 0.0$, $p_1 \rightarrow 1.0$, $p_2 \rightarrow 0.0$. The failure of SPEA and CoEAIM can be explained as follows. As the behavior is stochastic, given the same sequence of inputs the agent would probably exhibit different behaviors. Therefore, quantitatively measuring the difference between the models and agent (e.g., using square error) would not lead the model parameters to converge into their true values. For all methods, the means (standard deviations) of the AEs of each parameter of the evolved model with the highest fitness in the final generation over 30 coevolution runs are shown in Table 5.3. Clearly, *Turing Learning* infers the stochastic behavior significantly better than the two metric-based methods. There is no significant difference among “Interactive”, “passive 1” and “passive 2” setups of *Turing Learning* in terms of AEs of the evolved parameters.

In order to show the advantage of “Interactive” setup of *Turing Learning*, we investigate the convergence of model parameters during the evolutionary process. Figure 5.16 shows the convergence of the model parameters over generations for the three setups of *Turing Learning*. As we can see, the evolved model parameters in the “Interactive” setup converge much faster than those in the two passive setups. For the “Interactive” setup, after about 100 generations, all the three parameters converge into their true values. For the “Passive 1”, all the parameters converge after about 200 generations, while for the

"Passive 2" it takes longer time. In terms of v_1, there is much smaller disturbance in the "Interactive" setup than that in the other two setups.

5.4.3.2 Analysis of Evolved Classifiers

In order to investigate why the "Interactive" setup of *Turing Learning* learns the agent behavior much faster than the two passive setups. We post-evaluated how the classifiers interact with the agent during a trial. Figure 5.17 shows how the classifier with the highest subjective fitness in different generations (of a particular coevolution run) interact with the agent in a trial. Similar phenomenon could be observed in other coevolution runs. As shown in the top left of Figure 5.17, the classifier outputs H level and then waits until the agent 'jumps' from state 1 to state 2 (and this process is stochastic), and it immediately switches the light intensity from H to L level in order to make the agent stay in state 2 as long as possible. Therefore, the agent's behavior in state 2 (hidden information) can be observed longer and inferred efficiently. Note that the classifier has learned the good strategy and exhibited such 'intelligent' behavior at the very beginning of the coevolution run (before 50 generations in this case). After 500 generations, the classifier changed the strategy a little bit. Instead of always outputting H level, it keeps switching between H and L level until it observes the agent 'jumps' from state 1 to state 2, and after that it immediately switches the light intensity from H to L level. This is unlikely to happen when generating random sequence of input.

5.4.4 Results: Three States

In order to further demonstrate the advantage of "Interactive" setup of *Turing Learning*, this section discusses the results of inferring the 3-state stochastic agent behavior shown in Figure 5.14(b).

5.4.4.1 Analysis of Evolved Models

Figure 5.18 shows the distribution of the evolved models in the 1000th generation for all setups. The "Interactive" setup of *Turing Learning* is the only one that infers all the parameters of the agent with good accuracy. For the two setups using pre-defined metrics (SPEA and CoEAIM), all the parameters are not learned well. Table 5.4 compares the

Table 5.4: This table shows a comparison of all approaches for inferring the 3-state agent behavior shown in Figure 5.14(b). The values show means of AEs (defined in Equation 3.9) of each evolved model parameter with respect to that of the agent.

	v	v_2	p_1	p_2
Interactive (TL)	0.005	0.03	0.02	2.0×10^{-6}
Passive 1 (TL)	0.02	0.23	0.05	0.03
Passive 2 (TL)	0.02	0.47	0.08	0.13
Passive 1 (SPEA)	0.47	0.97	0.9	1.0
Passive 2 (SPEA)	0.47	0.98	0.9	1.0
CoEAIM	0.46	0.96	0.67	0.89

accuracy of each parameter of the evolved model with the highest fitness in the final generation over 30 coevolution runs using all approaches.

Figure 5.19 shows the evolutionary process of the models for the three setups of *Turing Learning*. As we can see, all the model parameters in the“Interactive” setup converged to their true value smoothly within about 200 generations. For the two passive setups, v_1, p_1 and p_2 are learned well, but they still take longer time to converge to their true values than those of the “Interactive” setup. There is dramatic disturbance during the evolutionary process of v_2 for the passive setups, and this parameter is not learned well.

5.4.4.2 Analysis of Evolved Classifiers

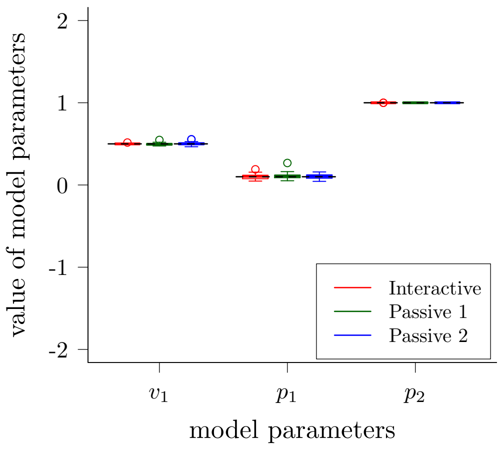
Figure 5.20 shows an example of how the classifier with the highest subjective fitness in different generations (of a particular coevolution run) evolved to interact with the agent. In other coevolution runs, we still observed similar phenomenon. As shown in Figure 5.20, the strategy learned by the classifiers shown in 3 of the 4 sub-figures (corresponding to 100th, 200th, 1000th generation) is as follows. The classifier outputs H first, and once the agent moves forward from state 1 to state 2, the classifier switches the light intensity into level L and keeps it in that level. As long as the agent moves forward from state 2 to state 3, the classifier switches the light intensity from L to H and keeps the light intensity in that level. Note that the ‘best’ classifier sometimes lost its ability to interact with the agent in the 500th generation shown in bottom left of Figure 5.20. However, this does not influence the learning process, as long as there are still some other classifiers in the population obtain this interactive ability.

5.5 Summary

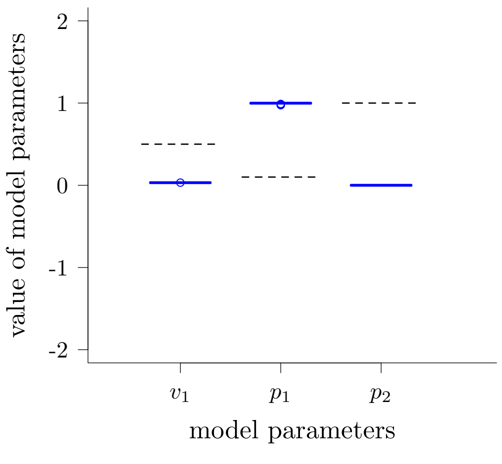
In this chapter, we extend *Turing Learning* with interactive ability to autonomously learn the behavior of an agent. We have shown that, by allowing classifiers to control the stimulus in the agent's environment, the system is able to correctly identify the parameters of relatively complex behaviors. The advantage of *Turing Learning* with interaction is validated using two case studies: stochastic and deterministic behaviors of an agent. In both case studies, the results show that learning through interaction can infer the agent behaviors better or faster than only through passive observation.

When inferring the deterministic behavior, the classifiers have learned to generate a complex sequence of inputs to extract all the hidden information from the agent, which facilitates the learning process. However, generating such sequences in random is unlikely. This makes the SPEA (in which the inputs are generated randomly) fail to infer all the parameters of the agent. However, by coevolving inputs and models (CoEAIM), the system still obtained very good model accuracy in terms of all parameters.

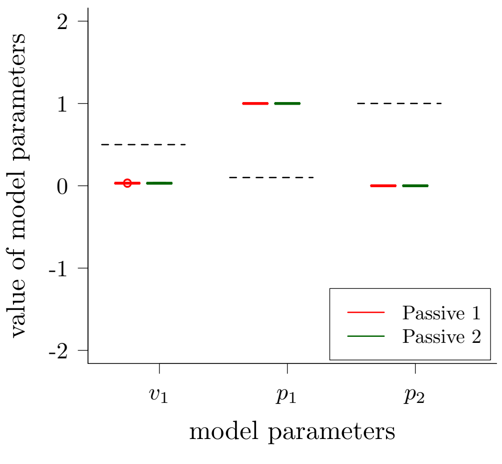
When inferring the stochastic behavior, the advantage of *Turing Learning* with interaction is highlighted. It performs significantly better than the two metric-based methods (SPEA and CoEAIM) in inferring stochastic behaviors. In the "Interactive" setup of *Turing Learning*, the classifiers learned to dynamically interact with the agent, which leading to infer all the agent's parameters in a very efficient way. The results suggest that a machine could also exhibit such intelligent behavior as observed in human behavior, and this could pave the way to further development of machine intelligence.



Turing Learning

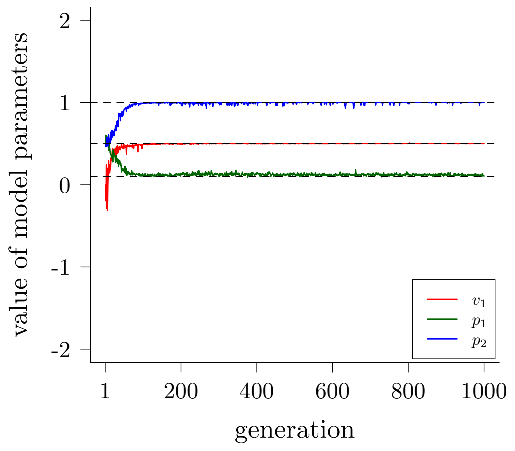


SPEA

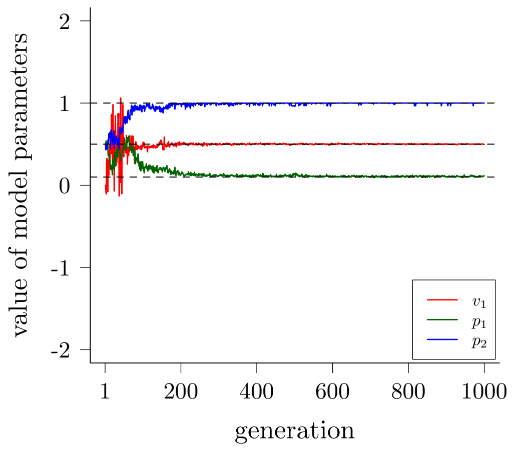


CoEAIM

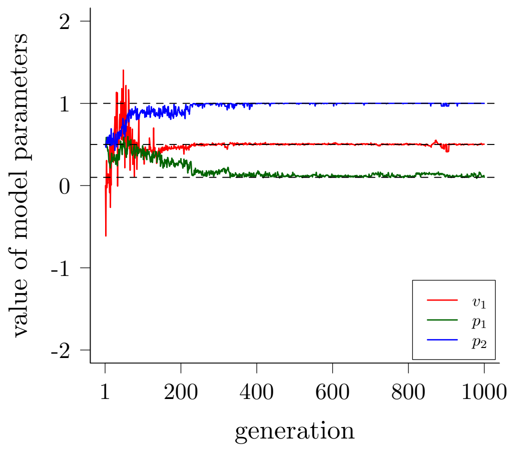
Figure 5.15: This plot shows the distributions of the evolved models with the highest subjective fitness in the 1000th generation in the coevolutions. Each box corresponds to 30 coevolution runs. The dotted lines correspond to the values of the three parameters that the system is expected to learn (i.e. those of the agent). See text for details.



Interactive



Passive 1



Passive 2

Figure 5.16: Evolutionary process of the evolved model parameters for (a) “Interactive”, (b) “Passive 1” and (c) “Passive 2” setups of the metric-free method when learning the 2-state agent behavior. Curves represent mean values across 30 coevolution runs. Dotted black lines indicate true values.

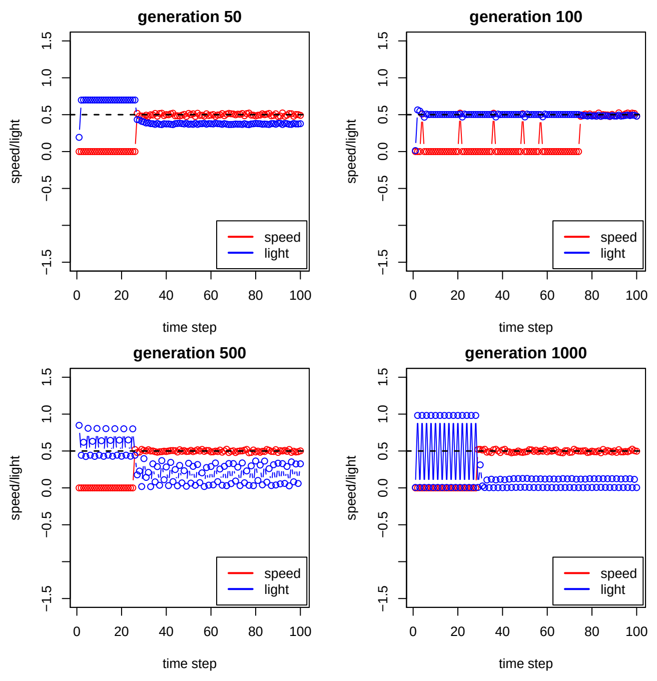
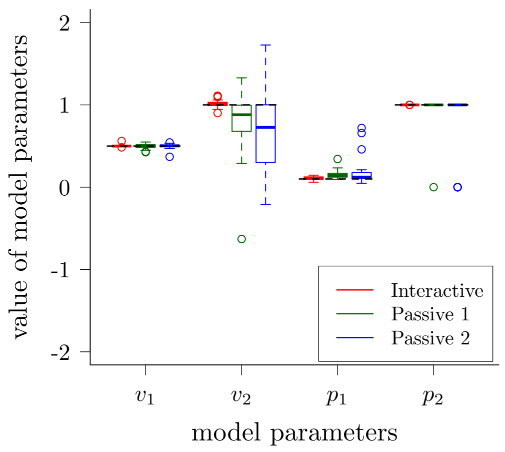
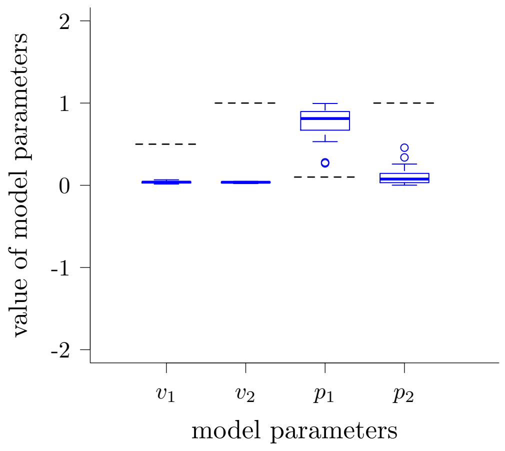


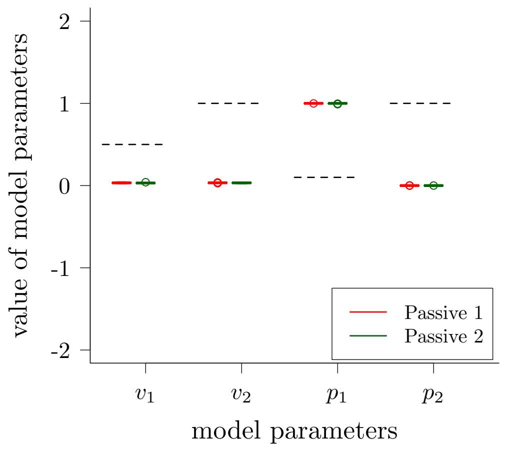
Figure 5.17: This plot shows an example of how the classifier with the highest subjective fitness in different generations (of a particular coevolution run) dynamically change the light intensity to interact with the 2-state agent during a trial.



Turing Learning

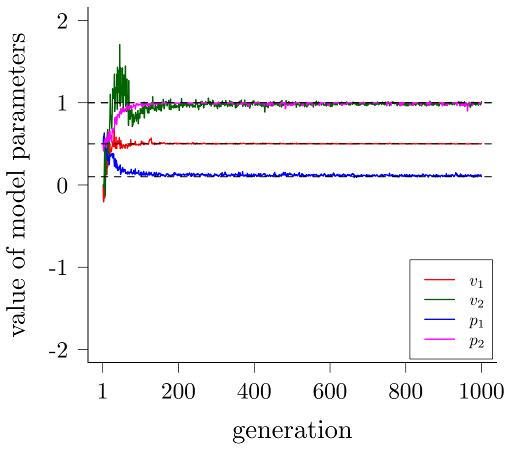


SPEA

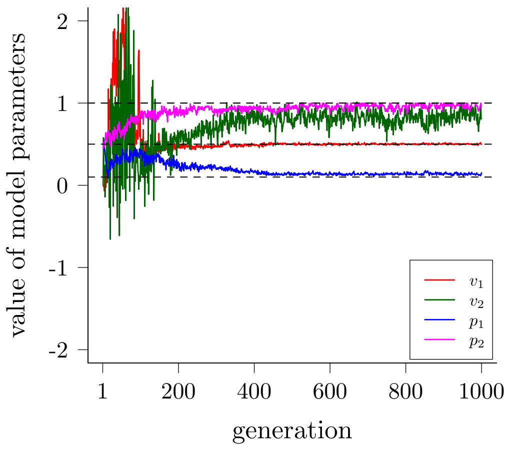


CoEAIM

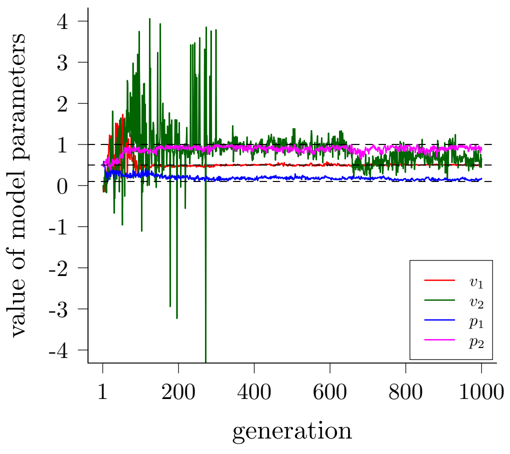
Figure 5.18: This plot shows the distributions of the evolved models with the highest subjective fitness in the 1000th generation in the coevolutions. Each box corresponds to 30 coevolution runs. The dotted lines correspond to the values of the four parameters that the system is expected to learn (i.e. those of the agent). See text for details.



Interactive



Passive 1



Passive 2

Figure 5.19: Evolutionary process of the evolved model parameters for (a) "Interactive", (b) "Passive 1" and (c) "Passive 2" setups of the metric-free method when learning the 3-state agent behavior. Curves represent mean values across 30 coevolution runs. Dotted black lines indicate true values.

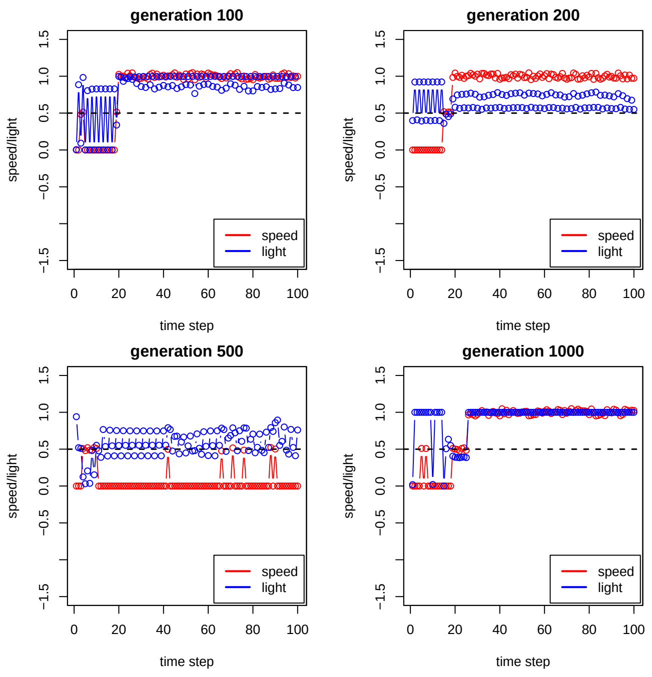


Figure 5.20: This plot shows an example of how the classifier with the highest subjective fitness in different generations (of a particular coevolution run) dynamically change the light intensity to interact with the 3-state agent during a trial.