Automatic Symbolic Modelling of Co-evolutionarily Learned Robot Skills

Agapito Ledezma, Antonio Berlanga and Ricardo Aler

Universidad Carlos III de Madrid Avda. de la Universidad, 30, 28911, Leganés (Madrid). Spain

Abstract Evolutionary based learning systems have proven to be very powerful techniques for solving a wide range of tasks, from prediction to optimization. However, in some cases the learned concepts are unreadable for humans. This prevents a deep semantic analysis of what has been really learned by those systems. We present in this paper an alternative to obtain symbolic models from subsymbolic learning. In the first stage, a subsymbolic learning system is applied to a given task. Then, a symbolic classifier is used for automatically generating the symbolic counterpart of the subsymbolic model.
We have tested this approach to obtain a symbolic model of a neural network. The neural network defines a simple controller of an autonomous robot. A competitive coevolutive method has been applied in order to learn the right weights of the neural network. The results show that the obtained symbolic model is very accurate in the task of modelling the subsymbolic system, adding to this its readability characteristic.

1 Introduction

The use of evolutionary computation (EC) techniques for software development suffers in some aspects from analogous problems to other software development methodologies or paradigms. In particular, we will focus in this paper in the declarative representation of the evolutionary generated descriptions; that is, how we (humans) interpret the output of the EC systems (their generated knowledge).

In the case of the application we present here, robot control, there are many types of knowledge that could be acquired by means of EC in order to build such systems. Examples are the internal model of robots, models of other robots, communication strategies, or reasoning heuristics. One way of automating this task consists on learning those models by either applying genetic algorithms [1], evolutionary strategies [2], classifier systems [3], or genetic programming [4]. Another view of this type of tasks is centered on the representation structure of the output: the systems can generate rules [5], neural networks [6], etc. When the output is represented in terms of subsymbolic structures (such as neural networks), it is very difficult to interpret the results in order to extract general conclusions on the correctness of the learned knowledge, its possible drawbacks, or the definition of improvements.

The ability to transform a procedural description of the reasoning process of a given control skill into a declarative representation allows to more easily share knowledge, or reason about other robots behaviors. Specifically, one of our goals was the study of automatic ways of extracting knowledge (models) from non-symbolic representations, such as neural networks. This has been already studied by some authors by analysing the internal structure of the neural network [7]. We propose an alternative that consists on modeling their behavior by observing how they "solve problems": what output they generate from what input.

In this paper, Section 2 describes the task that we have used as the testbed. Section 3 presents our learning approach to symbolic modelling. Section 4 describes the way in which experiments were defined, and presents the obtained results. Finally, Section 5 discusses the obtained results.

2 Co-evolution of skills for robot control

Problems related with robotics have been one of the main fields of application of evolutive computation. A wide variety of robotic controllers, to solve specific tasks, have been investigated; robot planning [8], wall following task [4], collision avoidance [9], etc. The traditional evolutive computation techniques have several disadvantages. Coevolution has been proposed as a way to evolve a learner and a learning environment simultaneously such that open-ended progress arises naturally, via a competitive arms race, with minimal inductive bias [10,11]. The viability of an arms race relies on the sustained learnability [12,13] of environments. The capability to obtain the ideal learner, the better environments where the learning takes place, is the main advantage of the coevolutive method.

In this work, the task faced by the autonomous robot is to reach a goal in a complex bidimensional environment while avoiding obstacles found in its path. In the proposed model, the robot starts without information about the right associations between environmental signals and actions responding to those signals. The number of inputs (robot sensors), the range of the sensors, the number of outputs (number of robot motors) and its description is the only starting information. From the initial situation the robot is able to learn through experience the optimal associations between inputs and outputs.

The input sensors considered in this approach are the ambient and proximity sensors, s_i.

The Neural Network outputs are the wheel velocities v_1 and v_2. The velocity of each wheel is calculated by means of a linear combination of the sensor values, equation 1, using those weights (Equation 1):

$$v_j = f(\sum_{i=1}^{5} w_{ij} s_i) \tag{1}$$

w_{ij} are the weights to be learned, s_i are sensor input values and f is a function for constraining the maximum velocity values of the wheels.

Weight values depend on problem features. To find them automatically, an evolutionary strategy (ES) with uniform coevolution (UC) is used [6] . In this

approach each individual is composed of a 20 dimensional-real valued vector, representing each one of the above mentioned weights and their corresponding variances. The individual represents the robot behavior resulting from applying the weights to the equation 1. The evaluation of behaviors is used as the fitness function for the ES.

From all the general controllers obtained for navigation purposes using UC, a controller has been selected for automatic adquisition of its model. The main characteristic of this controller is that it only determines the speed of wheel v_2. The speed of wheel v_1 is fixed to the maximum velocity.

3 Automatic acquisition of models

The behaviour of a reactive robot can be understood in terms of its inputs (sensors readings) and outputs. Therefore, there is a clear analogy with a classification task in which each input parameter of the robot will be represented as an attribute that can have as many values as the corresponding input parameter. In terms of a classification task, this allows to define a class for each possible output. Therefore, the task of modelling (generating a declarative representation of a robot behavior) has been translated into a classification task.

For this problem, any classification technique c could be employed: instance based learning [14], learning decision trees [15], learning rules [16,17], or neural networks [18]. However, we want to obtain a declarative symbolic representation. This constrains the type of technique to be used to those that generate symbolic representations, such as decision trees, or rules.

In a previous paper we have presented results for agents whose outputs is discrete [19]. Given that the outputs of the robot control task are wheel velocities, which are continuous values, two different approaches can be used: either discretize the output and use a typical symbolic classifier (like C4.5 [17]), or use a symbolic algorithm that is able to deal with continous outputs (like regression trees [20,21]). Here, we have followed both approaches.

In the preliminary results presented here, the robot to be modelled is controlled by a neural network. The symbolic techniques to model this robot are C4.5 [17,22][1] and M5 [21]. C4.5 generates rules and M5 generates regression trees. The latter are also rules, whose then-part is a linear combination of the values of the input parameters.

The actual learning task is as follows:

- Inputs:
  - Set of attributes that model the input parameters (sensors) of robot r_1
  - For each attribute, the set of values that its corresponding input parameter can have (in this case, they are continuous variables)
  - Set of possible outputs in the case of discrete classes, and continuous range in the case of continuous classes

[1] We have used WEKA's C4.5 rules implementation [22] rather the original Quinlan's algorithm

- Set of training instances T
- A classification technique c

– Output: a declarative classifier that provides the same (or approximate) output as the robot r would provide given the same input instances

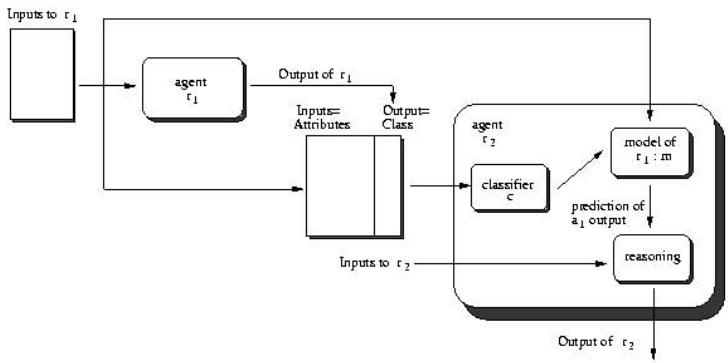

Figure 1. Architecture of the modelling of robots behavior.

4 Experimental Setup and Validation

The general framework is described in Figure 1 which shows the interrelation between the robot r_1, the modeler r_2 that tries to learn and reason about a model of r_1, the classification technique c used for modelling its behavior, and the obtained classifier m (model of r_1). This classifier m should model the behavior of robot r_1, in such a way that if one presents the same set of input patterns (sensory data) to both r_1 and m the error between the output provided by r_1 and m should be minimal.

To validate m (i.e. how closely r_2 knowledge models r_1 behaviour) we carried out ten-fold cross-validation. Testing data, which is different from the training data used in the previous section, was obtained in a similar way, by running r_1 and logging its inputs and outputs. In C4.5, the closeness of the performance of both r_1 and r_2 is measured as the number of examples in which the predictions of r_2 and r_1 differ (for the same sensory input). In the experiment that uses M5 to model r_1 behavior we use the correlation coefficient to measure the model error. The correlation coefficient is the measure of the correlation between the predic ted values and the real values of test instances. If correlation coefficient is 1, the predicted and real values are perfectly correlated. If the correlation coef ficient value is close to 0 there are no correlation. A -1 value means that they are inversely correlated. The next two subsections explain the validation carried out for the m obtained by C4.5 and M5, respectively.

4.1 Generating rules with C4.5

As C4.5 can only predict discrete outputs, wheel velocities have been discretized into five classes (see Table 1).

Table 1. Velocity Range

Class	**Velocity**	Number of instances
Slow	-1.000 to -0.500	54
Middle-slow	-0.499 to -0.003	26
Null	0	1
Middle-high	0.010 to -0.500	36
High	0.501 to 1.000	46
		165

The total number of testing instances is 165, which are distributed in five classes (see Table 1). Testing results are shown in Table 2. The first column shows the 10-fold crossvalidation modeling accuracy of pure C4.5, whose output is a decision tree. The second column shows the accuracy for C4.5 when it generates a set of rules. In short, the model m generated by C4.5-RULES is able to guess the output of the neural net 88 times out of 100, which is a quite good result.

Table 2. Results using C4.5.

	Hits/c4.5	**Hits/c4.5 Rules**
Cross-Validation	86.66%	88.48 %
Mean Absolute Error	0.0642	0.0534
Root Mean Squared Error	0.2221	0.2089

The rules learned can be seen in Table 3.

4.2 Generating regression trees with M5

The total number of instances is 976 with continuos classes (the class is the velocity of one wheel of r_1). We applied the M5 algorithm to generate a regression tree. The results of 10-fold crossvalidation are: Correlation Coeficient: 0.995, Mean Absolute error: 0.034 and Root Mean Square Error: 0.064. The regression tree predicts almost perfectly r_1 neural network.

The regression tree obtained is shown in Table 4. Each rule from a regression tree corresponds to a Linear Model (table 5) that estimate the class value (velocity of wheel v_2).

Table 3. Rules generating by C4.5 RULES.

SENSOR-4	> -0.046587 AND
SENSOR-4	> 0.362908: slow (30.0)
SENSOR-1	> 0.186667 AND
SENSOR-2	≤ 0.453333: slow (21.0/1.0)
SENSOR-4	≤ -0.424854 AND
SENSOR-3	≤ 0.16 AND
SENSOR-1	≤ 0.053333 AND
SENSOR-2	≤ 0.28: high (43.0)
SENSOR-4	> -0.046587 AND
SENSOR-3	≤ 0.026667: middle-slow (16.0)
SENSOR-4	≤ 0.060036 AND
SENSOR-3	≤ 0.506667 AND
SENSOR-2	≤ 0.293333 AND
SENSOR-5	> 0.058883: middle-high (28.0)
SENSOR-4	≤ 0.060036 AND
SENSOR-3	> 0.413333: middle-slow (8.0/1.0)
SENSOR-4	≤ 0.060036 AND
SENSOR-1	≤ 0.053333 AND
SENSOR-2	≤ 0.293333: middle-high (6.0)
SENSOR-3	≤ 0.026667 AND
SENSOR-1	> 0.053333: middle-slow (4.0/1.0)
SENSOR-3	> 0.026667: slow (4.0)
SENSOR-5	≤ 0.786346: high (3.0)
: null (2.0/1.0)	

Table 4. Rules from the regression tree.

*Sensor*1	*Sensor*2	*Sensor*3	*Sensor*4	*Sensor*5	Model
≤ 0.0333	-	≤ 0.233	≤ -0.841	≤0.29	LM1
≤ 0.0333	-	≤ 0.233	> -0.841 and ≤ -0.743	≤0.29	LM2
≤ 0.0333	-	≤ 0.233	> -0.743	≤0.29	LM3
≤ 0.0333	-	≤ 0.233	≤-0.59	>0.29 and ≤0.761	LM4
≤ 0.0333	-	≤ 0.233	≤-0.59	>0.761 and ≤0.975	LM5
≤ 0.0333	-	≤ 0.233	≤-0.59	>0.975	LM6
≤ 0.0333	-	≤ 0.233	>-0.59 and ≤ -0.453	-	LM7
≤ 0.0333	-	≤ 0.233	>-0.453	-	LM8
≤ 0.0333	-	> 0.233	≤-0.161	-	LM9
> 0.0333 and ≤0.22	-	-	≤-0.161	-	LM10
>0.22 and ≤0.587	-	≤0.213	≤-0.161	-	LM11
>0.22 and ≤0.587	-	>0.213	≤-0.161	-	LM12
>0.587	-	-	≤-0.161	-	LM13
≤0.193	-	-	>-0.161 and ≤0.134	-	LM14
≤0.193	-	-	>0.134 and ≤0.711	-	LM15
≤0.193	-	-	>0.711	-	LM16
>0.193	-	-	>-0.161	-	LM16

5 Conclusions

In this paper, we have presented an approach that allows to acquire a declarative representation of the behavior of a robot, by observing what output it produces from the inputs it receives. That is, instead of inspecting the robot internal model, it is considered as a black box and observed by another agent/robot. In particular, we have first used C4.5 to acquire a symbolic model (set of rules) of a neural-net based robot. Results show that C4.5 is quite good at modeling neural robots. The model obtained by C4.5 could be used by an opponent robot,

either directly, or even better, reasoning about the model, taking advantage of its symbolic representation.

Then, we have used M5 to obtain a regression tree that approximates even better the target robot. However, although there is greater accuracy in this second case, the knowledge obtained is not so easily understandable.

It is important to remark that this method will only be applied successfully to reactive agents. If the agent to model is not reactive (i.e. its output depends on something else, like memory, besides the sensors), models would be quite inaccurate.

Table 5. Linear Models.

Model	Prediction	Independent factor	*Sensor*1	*Sensor*2	*Sensor*3	*Sensor*4	*Sensor*5
LM1:	class =	0.36	-0.0936	+0.0711	-0.895	-0.723	- 0.276
LM2:	class =	0.358	-0.0936	+0.129	-1.25	-0.668	-0.35
LM3:	class =	0.0201	-0.0936	+0.0288	-1.2	-1.15	-0.423
LM4:	class =	0.201	-0.0936	+0.0751	-1.29	-0.969	-0.531
LM5:	class =	0.886	-0.0936	+0.0365	-1.11	-0.427	-0.859
LM6:	class =	0.933	-0.0936	+0.0365	-1.11	-0.427	-0.943
LM7:	class =	0.0814	-0.0936	+0.0257	-0.312	-1	-0.134
LM8:	class =	-0.0041	-0.0936	+0.0202	-1.22	-1.2	-0.453
LM9:	class =	-0.00956	-0.0936	+0.0155	-1.14	-1.12	-0.166
LM10:	class =	-0.073	-2.6	+0.0441	-0.964	-1.03	-0.264
LM11:	class =	-0.149	-2.23	+0.0065	-0.407	-0.895	-0.0413
LM12:	class =	-0.427	-1.33	+0.0065	-0.424	-0.488	-0.0413
LM13:	class =	-0.719	-0.62	+0.0065	-0.226	-0.305	-0.0413
LM14:	class =	-0.0313	-2.87	+0.0897	-1.14	-1.13	-0.33
LM15:	class =	-0.0435	-2.25	+0.0879	-1.11	-1.11	-0.329
LM16:	class =	-0.408	-0.757	+0.0364	-0.38	-0.652	
LM17:	class =	-0.839	-0.126	+0.013	-0.129	-0.135	

References

1. Vicente Matellán, José Manuel Molina, and Camino Fernández, "Genetic learning of fuzzy reactive controllers," *Robotics and Autonomous Systems*, vol. 25, no. 1-2, pp. 33–41, October 1998.
2. Antonio Berlanga, Pedro Isasi, Araceli Sanchis, and José M. Molina, "Neural networks robot controller trained with evolutionary strategies," in *Proceedings of the Congress on Evolutionary Computation*, Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, Eds., Mayflower Hotel, Washington D.C., USA, 6-9 July 1999, vol. 1, pp. 413–419, IEEE Press.
3. Araceli Sanchis, José M. Molina, Pedro Isasi, and Javier Segovia, "Rtcs: a reactive with tags classifier system," *Journal of Intelligent and Robotic Systems*, vol. 27, no. 4, pp. 379–405, 2000.
4. John R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA, USA, 1992.
5. Vicente Matellán, José M. Molina, Javier Sanz, and Camino Fernández, "Learning fuzzy reactive behaviors in autonomous robots," in *Proceedings of the Fourth European Workshop on Learning Robots*, Alemania, 1995.

6. Antonio Berlanga, Araceli Sanchis, Pedro Isasi, and José M. Molina, "A general coevolution method to generalize autonomous robot navigation behavior," in *Proceedings of the Congress on Evolutionary Computation*, La Jolla, San Diego (CA) USA, July 2000, pp. 769–776, IEEE Press.
7. Jude W. Shavlik and Geoffrey G. Towell, *Machine Learning. A Multistrategy Approach.*, vol. IV, chapter Refining Symbolic Knowledge using Neural Networks, pp. 405–429, Morgan Kaufmann, 1994.
8. Simon G. Handley, "The automatic generations of plans for a mobile robot via genetic programming with automatically defined functions," in *Advances in Genetic Programming*, Kenneth E. Kinnear, Jr., Ed., chapter 18, pp. 391–407. MIT Press, 1994.
9. K. Solano, R. A. Chavarriaga, R. A. Nuñez, and C. A. Peña-Reyes, "Controladores adaptables basados en mecanismos de inferencia difusa," in *Memorias del Segundo Congreso Asociación Colombiana de Automática*, 1997.
10. Sevan G. Ficici and Jordan B. Pollack, "Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states," in *Proceedings of the 6th International Conference on Artificial Life (ALIFE-98)*, Christoph Adami, Richard K. Belew, Hiroaki Kitano, and Charles Taylor, Eds., Cambridge, MA, USA, June 27–29 1998, pp. 238–247, MIT Press.
11. S. G. Ficici and J. B. Pollack, "Statistical reasoning strategies in the pursuit and evasion domain," in *Proceedings of the 5th European Conference on Advances in Artificial Life (ECAL-99)*, Dario Floreano, Jean-Daniel Nicoud, and Francesco Mondada, Eds., Berlin, Sept. 13–17 1999, vol. 1674 of *LNAI*, pp. 79–88, Springer.
12. Jordan B. Pollack, Alan D. Blair, and Mark Land, "Coevolution of a backgammon player," in *Proceedings of Artificial Life V*, C. G. Langton, Ed., Cambridge, MA, 1996, MIT Press.
13. Jordan B. Pollack and Alan D. Blair, "Co-evolution in the successful learning of backgammon strategy," *Machine Learning*, vol. 32, no. 1, pp. 225–240, 1998.
14. David W. Aha, Dennis Kibler, and Marc K. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, no. 1, pp. 37–66, jan 1991.
15. J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.
16. Ryszard S. Michalski, "A theory and methodology of inductive learning," *Artificial Intelligence*, vol. 20, 1983.
17. J. Ross Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA, 1993.
18. D.E. Rummelhart, J.L. McClelland, and the PDP Research Group, *Parallel Distributed Processing Foundations*, The MIT Press, Cambridge, MA, 1986.
19. Ricardo Aler, Daniel Borrajo, Inés Galván, , and Agapito Ledezma, "Learning models of other agents," in *Proceedings of the Agents-00/ECML-00 Workshop on Learning Agents,*, Barcelona, Spain, June 2000, pp. 1–5.
20. L. Breiman, J.H. Friedman, K.A. Olshen, and C.J. Stone, *Classification and Regression Tress*, Wadsworth & Brooks, Monterey, CA (USA), 1984.
21. J. Ross Quinlan, "Combining instance-based and model-based learning," in *Proceedings of the Tenth International Conference on Machine Learning*, Amherst, MA, June 1993, pp. 236–243, Morgan Kaufmann.
22. E. Frank and I. Witten, "Generating accurate rule sets without global optimization," in *Proceedings of the Fifteenth International Conference on Machine Learning*. 1998, pp. 144–151, Morgan Kaufmann.