Learning Agent-Based Model logic in multiple domains with Genetic Programming

Rory Greig[1] and Jordi Arranz[1]

[1]Improbable, London, UK, E1 6EG
rory@improbable.io, jordiarranz@improbable.io

Program synthesis (PS) and genetic programming (GP) allow non-trivial programs to be generated from example data. Agent-based models (ABMs) are a promising field of application as their complexity at a macro level arises from simple logic at the individual agent level, which can be encoded with relatively few lines of code in the agent update function. Learning only the relatively concise agent update rules makes it feasible to evolve complex emergent behaviours without having to synthesize correspondingly long, complex programs, which remains a challenging problem for current GP techniques.

In this work we demonstrate a new domain-independent approach which is able to evolve interpretable agent logic of an ABM from scratch. Our approach is closely related to *inverse generative social science* (Vu et al., 2019), which is a discovery process for testing multiple hypotheses automatically from data, in order to explain and illuminate observed phenomena. This technique enables us to generate models which are relatively free from existing domain priors and human preconceptions, and may shed light on completely new dynamics which have been overlooked because they are unintuitive or non-obvious. The resulting output is an interpretable symbolic model which can be understood and extended by a human modeller. Automatically learning the model logic for ABMs is not a new idea, however the existing literature, for example Vu et al. (2019), Husselmann (2015), Gunaratne and Garibay (2017) and Zhong et al. (2017), has mainly focused on learning model structure by mutating an existing model which is already adapted to the modelling problem in question, and works by recombining a set of "primitives" which is tailored to the domain, and therefore requires prior domain knowledge. This is sometimes referred to as "structural calibration'.

Listing 1: Simplified flocking generated code

```
params = [-7.0192e7, -1.6739e9, 111829.82, 8.6766e9,
            -1.6476e6, 214.2948, 5.4268e8, 4.9987]
temp_a = cos(params[3])
momentum = (temp_a .+ 1 - 2 ./ params[6]) .* velocity
separation = (temp_a .+ 4 ./ params[6]) .* nearby_fine_position
alignment = nearby_velocity
cohesion = (1 ./ params[6]) .* nearby_coarse_position
new_velocity = momentum .+ separation .+ alignment .+ cohesion
new_velocity = normalise(new_velocity) .* params[8]
```

We successfully induce the agent update rule of models in two different domains — flocking and opinion dynamics — by employing an evolutionary algorithm which evolves a population of individuals consisting of programs expressed in a domain-specific language (DSL). The genetic algorithm is a reimplementation of the one described in Real et al. (2020) and Real et al. (2019). We employ a flexible DSL which consists of basic mathematical building blocks. The learned programs are combinations of the DSL operators and operands. The fitness of each individual in the population is evaluated by executing its program and comparing the output to the output of a reference model. The best programs are then copied and randomly changed — i.e. mutated. This process is repeated until the average fitness of the population reaches a desired threshold. We also perform additional experiments to find out how well the evolved solutions perform with out-of-sample data — i.e., trajectories of the model which have not previously seen during training/evolution. The reference models used were the *Boids* flocking model (Reynolds, 1987) and a threshold opinion dynamics model (Deffuant et al., 2000).

For the flocking model a low loss of 3.558 was achieved, indicating that the best learned behaviour is able to reproduce the target data very accurately. To understand this low loss value we can assess whether it corresponds to model results which are subjectively similar to the reference behaviour, and whether it captures important flocking model characteristics. Figure 1 compares the output of the reference model and the best learned behaviour. The plot shows the agent positions after running each model for 80 timesteps. Subjectively, these plots show clear similarity in the flocking patterns. Many of the agents are in approximately the same position in both plots, and recognisable movement patterns and clusters of agents can be identified.

It is also informative to analyse the generated code itself. Listing 1 shows code for the synthesised flocking behaviour with the lowest loss. This was manually simplified to aid understanding by removing redundant lines, rearranging and combining lines, manipulating mathematically to extract common factors and renaming variables which cor-

Figure 1: Comparison of a single realisation of the reference flocking model in a Cartesian 2D space (top) with the evolved flocking model (bottom). Every trace follows the trajectory of a single agent over time.

respond to interpretable concepts. The simplified code is very similar to the reference model, incorporating variables which are recognisable as the separation, alignment and cohesion concepts found in the original *Boids* model. Being able to interpret this learned symbolic model is an advantage of the PS approach.

Listing 2: Simplified opinion dynamics generated code

```
params = [-44.4, 0.08822]
delta_opinion = opinion_a - opinion_b
temp_b = 2 * params[2] * delta_opinion - delta_opinion ^ 3
        - delta_opinion ^ 5
temp_b /= (params[1] - opinion_a)
temp_b = abs(temp_b) - delta_opinion ^ 4
next_opinion_a = opinion_a
next_opinion_b = opinion_b
if isapprox(temp_b, 0.0, atol=0.001)
        temp_c = delta_opinion /
                 (params[2] * (params[1] - opinion_a))
        next_opinion_a = opinion_a + 2 * temp_c
        next_opinion_b = opinion_b - temp_c
end
return Vec2(next_opinion_a, next_opinion_b)
```

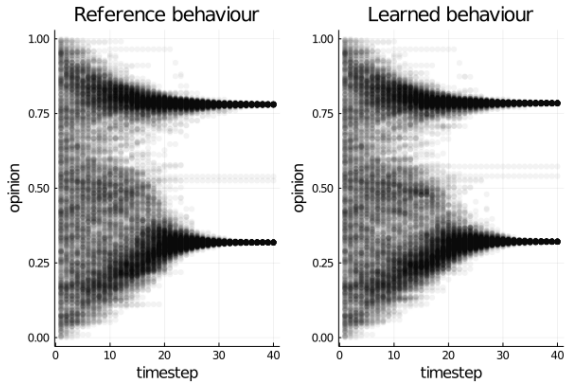

Figure 2: Side by side comparison of 40 time-steps of a single realisation of the reference opinion dynamics model (left) with the evolved opinion dynamics model (right). The *x-axis* shows simulation time and the *y-axis* shows the opinion value for each agent.

As with the flocking experiment, the opinion dynamics experiment also achieves a low loss. The outputs from running the reference and learned behaviours are compared in figure 2. These plots show how the opinions of each agent change over the course of a model run. The output of the learned behaviour is almost identical to the reference model. Listing 2 shows the code for the best learned opinion dynamics behaviour, which has been manually simplified in the same manner as listing 1.

Our results clearly show how PS and GP can be used beyond model calibration to learn full symbolic representations of core model logic, by only providing reference data and without encoding previous domain knowledge.

Despite employing only a single trajectory of the reference model in both cases the resulting models were able to generate identical macro behaviours. More importantly, we have also shown that the evolved solutions generalise very well and are highly interpretable. This improves over existing work in this area by learning models from scratch (starting from empty behaviours) and employing a generic and flexible DSL. This level of accountability offers a huge advantage over most deep learning (DL) techniques which suffer from opacity, opening the door to applications that go beyond modelling such as inverse generative social science in which the synthetic model is employed to explain and illuminate the phenomenon being modelled.

The next natural step we are aiming for is to apply our approach to real-world observed data. Perhaps this could provide new insights into the underlying dynamics of bird flocking behaviour, or any other modelling domain this was applied to, due to the increased flexibility and reduced reliance on human domain priors.

References

Deffuant, G., Neau, D., Amblard, F., and Weisbuch, G. (2000). Mixing beliefs among interacting agents. *Advances in Complex Systems*, 3:87–98.

Gunaratne, C. and Garibay, I. (2017). Alternate social theory discovery using genetic programming: Towards better understanding the artificial anasazi. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, page 115–122, New York, NY, USA. Association for Computing Machinery.

Husselmann, A. (2015). Generating opinion agent-based models by structural optimisation.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. (2019). Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4780–4789.

Real, E., Liang, C., So, D. R., and Le, Q. V. (2020). Automl-zero: Evolving machine learning algorithms from scratch.

Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. In *SIGGRAPH '87 Conference Proceedings*, pages 25–34.

Vu, T. M., Probst, C., Epstein, J. M., Brennan, A., Strong, M., and Purshouse, R. C. (2019). Toward inverse generative social science using multi-objective genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, page 1356–1363, New York, NY, USA. Association for Computing Machinery.

Zhong, J., Cai, W., Lees, M., and Luo, L. (2017). Automatic model construction for the behavior of human crowds. *Applied Soft Computing*, 56:368–378.