Using Neural Networks for a Universal Framework for Agent-based Models

Georg Jäger

Institute of Systems Sciences, Innovation and Sustainability Research, University of Graz, Graz, Austria

ABSTRACT

Traditional agent-based modelling is mostly rule-based. For many systems, this approach is extremely successful, since the rules are well understood. However, for a large class of systems it is difficult to find rules that adequately describe the behaviour of the agents. A simple example would be two agents playing chess: Here, it is impossible to find simple rules. To solve this problem, we introduce a framework for agent-based modelling that incorporates machine learning. In a process closely related to reinforcement learning, the agents learn rules. As a trade-off, a utility function needs to be defined, which is much simpler in most cases. We test this framework to replicate the results of the prominent Sugarscape model as a proof of principle. Furthermore, we investigate a more complicated version of the Sugarscape model, that exceeds the scope of the original framework. By expanding the framework we also find satisfying results there.

1. Introduction

1.1. Agent-based modelling

Agent-based models [1–4] are successfully used in the fields of complexity research [5], game theory [6,7], traffic simulations [8], social sciences [9], economics [10–12] and human systems [13]. The main idea of agent-based modelling is to simulate a system not from the top down, i.e. starting from the whole system and rules and equations that govern it, but from the bottom up, i.e. with the individual components (agents) that comprise the system as a starting point. In many systems this approach has huge benefits. While a top-down approach needs complete understanding of all the processes that lead to the dynamic of the system, like feedback, synergies and nonlinear effects, a bottom-up approach only needs understanding of the rules or equations that govern the individual behaviour of each agent. Other effects can then emerge due to interactions between agents [14]. Especially in areas, where the agents represent human beings or entities influenced by human behaviour and decision making (corporations, governments), agent-based modelling is a promising technique that is getting more and more popular recently [15–17], but also faces unique challenges [18–20].
For every agent-based model it is necessary to define the agents and the environment of the modelled system via certain qualitative or quantitative properties and find rules or equations that govern how the agents interact with each other and with their environment. The rules should include what information the agents have access to and what action they then take considering bounded rationality [21–23]. While the input the agents use for their decision is relatively easy to find and to justify, specifying the rules that lead to a decision is much more difficult and often relies on assumptions from psychology or economics, which are often hard to back up empirically or by theories. This makes the search for valid rules for agent behaviour to one of the biggest challenges in agent-based modelling.

To solve this problem of rules that are unknown to us, using a form of machine learning to find them is intuitive. This idea was explored in [24], where a framework for agent-based modelling was presented and used to replicate Schelling’s prominent segregation model [25]. The main idea of the framework is closely related to reinforcement learning [26], in the sense that agents learn how to behave to optimize their score or utility function. However, the goal is completely different. While reinforcement learning tries to find optimal solutions and provides the Neural Network with as much information as possible, the presented framework limits the available information to things the agents can actually perceive and also allows for non-optimal decisions. The goal is to emulate a realistic decision process, not find an optimal solution. The application to systems, in which reinforcement learning leads to the same results, like the Segregation model, can be seen as a stepping stone to systems, which do not have an optimal solution or where the optimal solution is never reached, like systems featuring a social dilemma. However, in order to verify that the framework works correctly, one has to first apply it to systems with known solutions. Only after this proof, systems that cannot be described using reinforcement learning can be tackled. Such a test of the framework was done successfully in [24].

This paper is now an expansion of the work performed in [24]. While a proof of principle was given, the investigated system was extremely simple:

- The agents only had access to one action, so their decision was Boolean: perform the action or do not perform any action. It was unclear, if the framework could also handle non-Boolean decisions.
- The link between sensory input and optimal decision was straightforward: Numerically adding all the input was sufficient to make a choice and no actual spatial information was needed.
- Random decisions lead to realistic states of the agents. However, this is only true for few systems. In general, one cannot expect to encounter the same system states via random and via goal-oriented decisions.

In this study, we expand the ideas from [24] to address the issues stated above by investigating an implementation of the Sugarscape model. Since here, agents can move in all directions or remain stationary, the decision is non-Boolean. Also, the spatial information hidden in the sensory data is paramount to arrive at a correct decision. What makes the system especially interesting is the fact, that intelligent agents will converge in one area, while agents making random decisions will disperse, which
means that the system is difficult to explore using just random decisions. Furthermore, the system offers a simple way of adjusting the difficulty of the problem by introducing competition between the agents. Thus we can construct a system, in which the original framework breaks down and can adapt it to increase its scope. For those reasons Sugarscape is an ideal candidate for these investigations.

The paper is organized as follows: Section 1.2 gives a short introduction to Neural Networks. The presented framework is described in Section 2.1, while Section 2.2 applies this framework to the prominent Sugarscape model in order to close the identified research gap. Results of this application and the expansion of the framework are presented in Section 3. Section 4 discusses these results and the larger implications of this framework. Furthermore, possible expansions and further research opportunities are outlined.

1.2. Artificial neural networks

Artificial Neural Networks [27–30] are computing systems that are capable of calculating an output from an arbitrary input in a similar way as biological neural networks, i.e. human or animal brains, do. Artificial Neural Networks can be capable of deep learning [31,32] and can solve complex problems, like for example classification problems [33]. They have a huge range of applications in many different fields like medical diagnosis [34–36], face detection [37,38], finance [39] or hydrology [40]. Neural networks can also be used for nonlinear adaptive control in multi-agent systems [41,42].

An Artificial Neural Network consists of several layers of nodes, similar to the neurons in a brain. Nodes within one layer are not connected to each other, but each node has a link from each node of the previous layer and a link to each node in the next layer. For every input in the input layer, there is a deterministic output in the output layer. The weights of the individual links govern which input leads to which output. The core idea of Artificial Neural Networks is to optimize the link weights in such a way, that every input yields the expected output. There are various techniques for this training step [43,44] with different advantages and disadvantages. Once an Artificial Neural Network is trained it can not only produce outputs for the data it trained with, but also for completely new data.

A standard problem, for which Artificial Neural Networks are successfully used, is the classification problem. A certain input (a picture, a text or more abstract, an array of numbers) needs to be identified as a member of one specific class. During training a database is used that includes different inputs and the class they belong to. After training, the Artificial Neural Network is capable of classifying completely new input very efficiently. In this study, we use an Artificial Neural Network to solve such a classification problem in order to model the decision making process of agents in an agent-based model.

2. Methods

2.1. The framework

The goal of the presented framework is to provide a universal technique for agent-based models, in which the decision making process of the agents is not determined by theory-
driven or empirically found rules, but rather by an Artificial Neural Network. The process itself can be separated into four phases:

1. Initialization
2. Experience
3. Training
4. Application

In the first phase, Initialization, the important features of the agents and their environment need to be defined. Agents need some kind of input that can be both qualitative or quantitative. In the simplest case this is sensory input or general knowledge, but also other inputs that influence decision making are thinkable, like memory or individual preferences. One also needs to define the decision an agent needs to make in each time step. Most generally this will be the choice between several possible actions. Each agent also needs a target or a goal, mathematically expressed as a score or utility function that the agent wants to maximize. For simple economic systems, profit is a score that can easily be quantified and used as a target, but many other goals can be used, possibly including fairness [45,46] and social preferences [47,48]. Depending on the system, completely different properties can be used as goals (e.g. minimization of travel time for traffic systems) and they could be different for each agent. Once the system, the agents, the input for the agents, the decision and the goal of each agent is defined, the Initialization phase is finished. Note, that defining a utility function is in most cases easier than finding a rule set that leads to optimizing this function. Think about a game of chess: The utility function is easy to define (1 for a win, 0 otherwise), but finding a set of rules that gets the position of all pieces as input and a realistic move (probably even related to player skill) is nearly impossible. In that sense, the Initialization phase is relatively simple, when compared to traditional agent-based models.

In the second phase, Experience, agents make random decisions to collect information in a database that can then be used to train the Artificial Neural Network. In every time step agents first observe their environment and store all the relevant data in the database. They also calculate their current score. Next, they make a random decision and recalculate their score once more after performing the action they chose. The result of this decision is then rated as positive, if the score increased, negative, if the score decreased, or neutral, if there was no or minimal change in score. The complete data set of one experience thus includes the inputs, the decision that was chosen and the result of this decision. The reason, why the result of the decision is only stored qualitatively, is because we do not assume that the agents have quantitative knowledge about their own utility. They only sense if their situation improved or not, but could not give an accurate number to quantify it. The whole process is depicted in the upper panel of (Figure 1). In order to generate a sufficient pool of information, many time steps are necessary, in which agents should encounter and rate a huge amount of combinations of input, decision and result. Note, that in this phase the input has no influence on the decision making of agents, but is only stored in the database to enable the training of the Artificial Neural Network in the next phase.

In the Training phase, the Artificial Neural Network is trained to solve the following classification problem: The Neural Network is presented with the input of the agent and
a decision and should estimate whether this is a good, a neutral or a bad decision. Various methods could be used for this type of problem. In order to keep the framework versatile, a hidden layer approach \([49,50]\) is used here. After training the Artificial Neural Network, an unused part of the database gathered in phase two is used for cross validation \([51]\). The Artificial Neural Network is implemented using scikit-learn \([52]\).

In the last phase, Application, the trained Neural Network is used for decision making. Agents are reset to their original initial conditions so that the actions performed during the Experience phase have no direct influence on the Application phase. In each time step agents gather inputs and use the Artificial Neural Network for decision making. The current inputs are combined with every possible decision and the Neural Network estimates whether such a decision would be good or bad. The agent then chooses the option with the highest confidence for a positive result. This process is depicted in the lower panel of (Figure 1).

Compared to the conventional approach to agent-based modelling, using this framework has various advantages. First and foremost, the most difficult task in developing an agent-based model, namely the definition of the rules and equations governing agent behaviour is translated to the definition of the goals of each agent and which parts of the system they can observe. The connection between input and decision is then handled objectively by an Artificial Neural Network. This also means that the model is highly adaptive. If the goals of the agents, their input or properties of the system change, retraining the Neural Network is the only adaptation that is necessary. Section 3 showcases this flexibility with different examples. In addition to the framework’s flexibility and objectivity, it also enables an intuitive way to include bounded rationality in a model. Agents always decide on what they think has the highest chances of being a good decision. If they have incomplete or wrong

---

**Figure 1.** An agent during the experience and the application phase. While the agent decides randomly during the gathering of experience, an Artificial Neural Network is used during the application phase.
information, we do not need to find special rules that would rely heavily on assumptions, but are still able to use the same process, i.e. an Artificial Neural Network that is just trained differently or uses incomplete/wrong input. In that case, the Neural Network acts as more than just a method to classify possible decisions into good, bad and neutral decisions: it models the decision process of an agent realistically, in the sense that, depending on the gathered experience, the choice might not be optimal all the time.

To showcase that the framework can be used to implement agent-based models without the need of manually defining rules for agent behaviour, it is applied to reproduce the prominent Sugarscape model [53] in the following section.

### 2.2. Application of the framework

The Sugarscape model is a simple agent-based model that is used to explain and investigate wealth distribution in societies. There are many variations and expansions, but the main principle is always the same: Agents inhabit a grid; each point of this lattice is assigned a certain value of sugar as an abstract measure for wealth. Agents then move around the grid, trying to find an optimal position to maximize their sugar. The actual rules for their behaviour are highly dependent on the specifics of the model. Some variations use instantly replenishing sugar, while others assume that a certain time needs to pass for the sugar to be replenished, which of course has a huge impact on the competition between agents and therefore the rules that they obey. More complicated expansions, like taxation, pollution or combat [54,55] are possible and again require a new set of rules and equations that govern agent behaviour and decision making. In its most basic form, the primary result of the Sugarscape model is that wealth is not distributed evenly among the population, and not distributed Gaussian around a mean value, but rather follows a Pareto distribution [56,57], meaning that the number of wealthy agents is small, while the number of agents with little wealth is large. In the following, we will apply the presented framework to implement a basic Sugarscape model in order to reproduce this result without defining the rules that govern agent behaviour manually.

In our implementation of Sugarscape, 75 agents are positioned randomly on a $21 \times 21$ grid with periodic boundary conditions. The size of the system has only a small influence on the model, so those values were chosen to facilitate visualization. Each point in the grid is assigned a certain value of sugar $s$, according to

$$s(x, y) = 15 - \sqrt{(10 - x)^2 + (10 - y)^2}$$  

leading to a maximum of 15 sugar in the middle of the grid at $(x, y) = (10, 10)$ and a negative sugar gradient in each direction outwards, with minimal, but positive, sugar at the corners of the world. In each time step, agents gather the sugar at the position they are currently at. If another agent already gathered at the same position, gathering yields less and the sugar gain is multiplied by an arbitrarily chosen competition factor $cf < 1$. Here we chose $cf = 0.8$, which ensures that an empty space neighbouring an occupied space will always be favourable, since the competition factor outweighs the sugar gradient defined in (1). Agents have the choice to remain stationary or move 1 step north, south,
east or west on the grid. Their goal is to maximize the sugar they can gather. In order to find the rules for agent behaviour, we proceed as follows.

In the Initialization phase, agents are positioned randomly on the grid. Their input, i.e. the information they have access to is defined: The agents can observe the amount of sugar on their patch and the amount of sugar on each of the 4 neighbouring patches. In addition, they can also see the number of agents on their current patch and on each neighbouring patch. The score, which is used to determine if a decision was good or not, is the amount of sugar they gathered in their turn. The decision the agent faces is which of the 5 possible actions it should perform: remaining stationary, moving north, moving south, moving east or moving west.

During the Experience phase agents make random decisions and add new entries to the database consisting of a vector with all their sensory input, the randomly chosen action and the result, i.e. if the score increased, decreased or stayed the same due to this decision. To gather a sufficient amount of data, the Experience phase lasted for 5000 time steps, which is computationally relatively cheap and can be calculated in a few seconds using a single core. This number was chosen to ensure that there is enough data for training. See Section 3 for more details on the amount of data required to successfully train the Neural Network. Note, that the sensory input consists of just numerical values arranged in a vector, as depicted in (Figure 2): There is no information about the meaning of the individual elements of the vector, or connections between the elements. For example, Element 1 is the sugar amount on the patch directly north of the agent and Element 5 is the information how many agents are currently on this patch. In the database however, this structure is completely unknown and needs to be learned by the Neural Network in the following phase. This can be seen as a significant difference to reinforcement learning, where one would process the input data in order to obtain better results, e.g. by calculating the effective sugar amount of each patch using ((1)) and the competition factor. In contrast to that, we provide only input that comes directly from an agents senses. In this example, this approach leads to the same results as reinforcement learning, which is necessary for the validation of the framework, but for more complex

![Figure 2. Creation of a sensory vector. The agent observes its surroundings and collects all available data. All information is stored in a vector without any indication which element stores which information. Thus, connection between any elements (e.g. between element 1 and element 5) need to be learned by the Neural Network.](image-url)
systems, this may not be the case and the framework can lead to a more realistic description of the agents behaviour.

The Training phase uses the gathered data to train a Hidden Layer Neural Network to solve the underlying classification problem. Here, we use a Multi-Layer Perceptron [52], but other methods could be used as well. Utilizing a Random Forest approach, for example, leads to nearly identical results, with the main difference being computation time, which is not the focus of this study.

A part of the data is used for cross validation, showing a success rate of roughly 90%. Note, that the goal is not to reach a success rate of 100% in any condition, but rather to depict a decision process realistically, possibly leading to wrong decisions. The trained Neural Network can then be used to classify a combination of input variables and a decision into the three classes good decision, bad decision and neutral decision, which is necessary for the Application phase.

For the Application phase, all agents are again positioned randomly. In every step they observe their environment and use the trained Neural Network to classify each of the actions they could perform. To decide on the best action, each action is rated on the following basis:

\[ R = c_p - c_n, \]

with \( c_p \) being the confidence of the decision being positive and \( c_n \) the confidence of the action being negative. Of all the possible actions, the action with the highest rating \( R \) is chosen. Ties are resolved randomly.

In addition to this basic version of Sugarscape, we also explore the flexibility of the framework by making modifications to the model. In the first modification we remove the competitive component of the model, by removing the penalty of harvesting on a patch that has already been harvested in the same turn. Agents should therefore ignore other agents and just head to middle of the world. Without any modification of the framework and a simple change of the competition factor \( cf \) to 1.0, the framework should be able to show correct agent behaviour.

After this variant of the Sugarscape model, we will investigate a more complicated one, specifically tailored to exceed the original scope of the framework. This necessitates an expansion of the original framework, largely increasing its scope.

3. Results

3.1. Traditional Sugarscape

(Figure 3) shows a simulation run of the basic version of Sugarscape implemented using the framework. The left panel depicts the spatial distribution of the agents after 20 turns. The agents are clustered around the centre and no patches are occupied by more than one agent. The right panel shows a histogram of the wealth distribution of the agents, i.e. the sugar they gathered in the last turn. The resulting distribution is a Pareto distribution, so the implementation reproduces the result of the original Sugarscape model, without the need for defining specific rules for agent behaviour. This cannot be taken for granted and is not simply an effect of the system or the sugar gradient. When using agents that make completely random decisions this result cannot be reproduced, as shown in (Figure 4).
Here the spatial distribution of the agents (left panel) is completely random. The wealth distribution (right panel) is not a Pareto distribution, but rather a Gaussian distribution. This shows, that the result of the original Sugarscape model cannot be reproduced by
randomly acting agents. A more sophisticated approach to decision making is necessary, and while the original idea to define fixed rules for agent behaviour is of course valid, the method presented here works as well and offers unique advantages.

One of these advantages is showcased in (Figure 5). It shows the results for a variation of the Sugarscape model, in which there is no competition between the agents. The spatial distribution (left panel) shows that all agents found the maximum and moved there. This can also be seen in the wealth distribution (right panel), where it is visible that all agents obtained maximal wealth. Cross validation of the Neural Network shows, that it reaches 100% accuracy for the presented classification problem, since without interaction of agents with each other, the problem becomes much simpler. Note, that the structure of the classification problem has not changed. There is still input that relates to the occupancy of neighbouring patches. During training, the Neural Network successfully learns that this input is irrelevant for solving the classification problem.

3.2. Exploring the limits of the framework

In the previous section we were able to show that the framework also works for non-Boolean decisions that require spatial information. However, one final question remains: What about systems in which the relevant states for Application are never reached by agents making random decisions? The Sugarscape Model offers an opportunity to investigate such systems: During the Experience phase it will be rare that two agents are on the same patch. System states in which three agents are on the same patch are even rarer and there is definitely not enough data on these states to successfully train a Neural Network. For the previously investigated systems it was not important if two or three agents share a patch, since it did not change the

Figure 5. Results of a Sugarscape implementation, in which there is no competition between agents. Agents automatically learn how to act under these conditions and all gather in the area with the most sugar.
fact whether a patch was desirable or undesirable. Thus, the agents decided correctly, even though they did not encounter such situations before. If we, however, change the rules of competition we can explore the limits of the framework.

We now change how competition works: Competition now only has an effect, when three or more agents share the same space. If two agents are on the same patch, both of them are awarded 100% of the available sugar. We can assume that the framework will have difficulties with such a system, since the agents would need to encounter many situations in which three agents share the same space, which is rare for randomly moving agents. Using the framework in the same manner as before, we find the results presented in (Figure 6).

As we can see, the framework fails. Here, agent behaviour does not take all the properties of the system into account. Many occupy the same patch, because they did not encounter enough situations in which this behaviour was unfavourable. In order to find out, if a longer Experience phase can remedy that, we take a closer look at the training process in (Figure 7).

Here we compare the relative number of good decisions (i.e. decisions which do not decrease the score) during Training and Application phase. We see that at a certain amount of training data, the positive decisions during training saturate close to 1 and show no improvement with more data. The relative number of good decisions during the Applications phase, however, remains low throughout and does not show any improvement with more training data. This means, that additional training data collected in the same manner will not solve this problem. Since during the Experience phase the agents mainly encounter system states that are not relevant for the system, they have no chance of learning about the system, as they will encounter it during the Application phase. A modification to the framework is needed.

![Figure 6. The framework fails for complicated competition rules.](image-url)


### 3.3. Expanding the framework

In the previous section, we found an example in which the presented framework fails, because the states encountered during Experience phase and Application phase differ too much. We will now adapt the framework, in order to increase its scope to such systems.

Instead of a sequential approach of training followed by application, we switch to an iterative approach:

1. Initial Experience (Random decisions)
2. Training of Neural Network (NN)
3. Advanced Experience (decision based on NN + random decisions)
4. Training of Neural network (NN)
5. Go to 3

The first two steps are identical to the original framework, but in step 3 we combine experience and application: The agents make their decision based on the current NN, but additionally choose a random, different action. The random action is only evaluated and stored as experience, but not actually performed. This ensures, that the encountered states are those relevant for the current NN while also giving the possibility to explore different actions. It would also be possible to evaluate all possible actions, but the chosen approach has the significant advantage, that calculation time does not scale with the number of available actions.

In this Advanced Experience phase, the agents encounter new system states, which are more relevant for application, since they are based on decisions that are more realistic than random choices. Nevertheless, we cannot assume that those states are the ones encountered in the application, since the NN will change after training and so will the encountered states. Thus, we have to use an iterative approach. Only if the NN does not change significantly, the process converged and the encountered states in the last experience phase will be similar to those encountered during application.

![Figure 7. Success during training and application phase for increasing training data.](image-url)
We try this new approach on the system described in Section 3.2. The results are depicted in (Figure 8). We can see that the agents find an optimal solution, even for this system. Most spaces are occupied by exactly two agents; all agents which occupy a space alone cannot improve their score by moving. Thus, we reached a Nash equilibrium [58,59].

The number of iterations that is needed for convergence is not known in the beginning of the process. Strictly speaking, one has to check how much the NN changed between two steps, i.e. if it leads to the same agent behaviour. In practise, the use of much simpler proxies is possible to check convergence. Here, we use the average agent score during each iterative step, shown in (Figure 9). Convergence of this observable is a strong hint for the convergence of agent behaviour, although strictly speaking it is possible to think systems in which different behaviour leads to the same score. For such systems, other proxies need to be investigated as well, or the actual NN needs to be checked for differences.

Figure 8. An iterative process leads to optimal results. Position of the agents (left) and distribution of score (right).

Figure 9. After 6 iterative steps, the average agent score converges. A closer inspection of the system reveals, that also the NN converged at that point.
Thus, we expanded the framework to an iterative process, thereby increasing its scope to systems that cannot be explored well using random agent decisions.

4. Conclusion

We showed that by using the presented framework it is possible to implement an agent-based model without the need to manually find rules or equations for agent behaviour, which is the most challenging step for most agent-based models. Within the framework, agents first make random decisions and gather experience. Then a Neural Network is trained to be able to judge a combination of (sensory) input and a decision, classifying this decision as positive, negative or neutral. Here, the Neural Network is not used as a form of optimization, but rather as a realistic depiction of a decision process, including the possibility of errors in judgement.

We demonstrated the advantages of this approach by applying it to reproduce the results of the prominent Sugarscape model. To show the flexibility of the framework, we then made slight changes to the modelled system by removing the competition between the agents. While a traditional approach to agent-based modelling would require a reformulation of the rules for agent behaviour, here the Neural Network is automatically retrained to accommodate the changes in the system and we naturally end up with realistic agent behaviour.

We also explored the limits of the framework and found that the original approach fails, once system states that are relevant, if agents act to reach a goal, do not appear during an Experience phase that only features random decisions. Therefore, we expanded the approach to include an iterative learning process and a way to check for convergence. Thus, we expanded the framework to a much larger class of systems.

This work only serves as a stepping stone to application on systems that cannot be investigated using reinforcement learning. Many expansions are possible to extend the scope of the framework. Currently, agents only think one step ahead, which works well for the Sugarscape model, but may not be enough for other systems. However, the framework can be expanded for thinking ahead straightforwardly. Currently, the decision the agents face is what action to take next. This can be easily expanded to the decision what the next 2 or 3 actions should be. The concept would not change, the training database would only gain one additional entry for each turn the agents think ahead. In the same manner, memory could be included, so that agents also have access to the information which decisions they made earlier.

The next step on the path to a framework that can be used universally for agent-based models of arbitrary systems is to apply it to different, more complicated systems. For now, only systems that can be solved via reinforcement learning have been investigated, to give a simple way of model validation. The main application area of this framework, however, will be systems without clear solutions. Cooperation games, social dilemmas, public good games and similar systems are interesting future applications. Here we could compare against data collected in social experiments and analyse if the framework leads to better results than traditional application of machine learning or agent-based modelling techniques.

The goal, however, is to keep the framework generic and not tailor it specifically to suit one specific system. Once it is tested thoroughly it will be made available as open source code, so that the scientific community can use it as a flexible and universal tool for agent-
based modelling, giving easy access to an approach to modelling that becomes more and more relevant for many research areas.

Disclosure statement
No potential conflict of interest was reported by the author.

Funding
Open access funding was provided by the University of Graz [Open Access Funding].

ORCID
Georg Jäger http://orcid.org/0000-0001-9751-0768

References
[1] N. Gilbert, Agent-based Models, Vol. 153, Sage, Los Angeles, 2008.
[2] D. Helbing, Agent-based modeling, in Social Self-organization, Springer, Heidelberg, 2012, pp. 25–70.
[3] F. Castiglione, Agent based modeling, Scholarpedia 1 (2006), pp. 1562. doi:10.4249/scholarpedia.1562.
[4] A. Getchell, Agent-based modeling, Physics 22 (2008), pp. 757–767.
[5] R.M. Axelrod, The Complexity of Cooperation: Agent-based Models of Competition and Collaboration, Princeton University Press, Princeton, 1997.
[6] S. Parsons and M. Wooldridge, Game theory and decision theory in multi-agent systems, Auton Agent Multi Agent Syst 5 (3) (2002), pp. 243–254. doi:10.1023/A:1015575522401.
[7] E. Semsar-Kazerooni and K. Khorasani, Multi-agent team cooperation: A game theory approach, Autom 45 (10) (2009), pp. 2205–2213. doi:10.1016/j.automatica.2009.06.006.
[8] B. Chen and H.H. Cheng, A review of the applications of agent technology in traffic and transportation systems, IEEE Trans. Intell. Transp. Sys 11 (2) (2010), pp. 485–497. doi:10.1109/TITS.2010.2048313.
[9] J.M. Epstein, Agent-based computational models and generative social science, Complexity 4 (5) (1999), pp. 41–60. doi:10.1002/(SICI)1099-0526(199905/06)4:5<41::AID-CPLX9>3.0.CO;2-F.
[10] J.D. Farmer and D. Foley, The economy needs agent-based modelling, Nat 460 (7256) (2009), pp. 685–686. doi:10.1038/460685a.
[11] C. Deissenberg, S. Van Der Hoog, and H. Dawid, Eurate: A massively parallel agent-based model of the european economy, Appl. Math. Comput. 204 (2) (2008), pp. 541–552. doi:10.1016/j.amc.2008.05.116.
[12] L. Tesfatsion, Agent-based computational economics: Growing economies from the bottom up, Artif. Life 8 (1) (2002), pp. 55–82. doi:10.1162/106454602753694765.
[13] E. Bonabeau, Agent-based modeling: Methods and techniques for simulating human systems, Proc. Natl. Acad. Sci. U.S.A 99 (Supplement 3) (2002), pp. 7280–7287. doi:10.1073/pnas.082080899.
[14] J.L. Dessalles, J. Ferber, and D. Phan, Emergence in Agent-Based Computational Social Science: Conceptual, Formal, and Diagrammatic Analysis, in Intelligent Complex Adaptive Systems, IGI Global, Pennsylvania, 2008, pp. 255–299.
[15] R. Conte and M. Paolucci, On agent-based modeling and computational social science, Front Psychol 5 (2014), pp. 668. doi:10.3389/fpsyg.2014.00668.
[16] S.H. Chen, Agent-based Computational Economics: How the Idea Originated and Where It Is Going, Routledge, London, 2017.

[17] M. Paolucci and R. Sacile, Agent-based Manufacturing and Control Systems: New Agile Manufacturing Solutions for Achieving Peak Performance, Crc Press, Boca Raton, 2016.

[18] J.S. Lee, T. Filatova, A. Liguori-Zielinska, B. Hassan-Mahmooei, F. Stonedal, I. Lorscheid, A. Voinov, j.g. Polhill, Z. Sun, and D.C. Parker, The complexities of agent-based modeling output analysis, J. Artif. Soc. Soc. Simul. 18 (4) (2015), pp. 4. doi:10.18564/jasss.2897.

[19] A. Klabunde and F. Willekens, Decision-making in agent-based models of migration: State of the art and challenges, European, J. Popul. 32 (2016), pp. 73–97.

[20] A. Smajil and O. Barreteau, Empirical Agent-based Modelling-challenges and Solutions, Springer, Heidelberg, 2014.

[21] H.A. Simon, Models of Bounded Rationality: Empirically Grounded Economic Reason, Vol. 3, MIT press, Cambridge, 1982.

[22] W.B. Arthur, Inductive reasoning and bounded rationality, Am Econ Rev 84 (1994), pp. 406–411.

[23] H.A. Simon, Theories of bounded rationality, Decis. Organ 1 (1972), pp. 161–176.

[24] G. Jäger, Replacing rules by neural networks a framework for agent-based modelling, Big Data and Cognit. Comput 3 (4) (2019), pp. 51. doi:10.3390/bdcc3040051.

[25] T.C. Schelling, Dynamic models of segregation, J. Math. Sociol. 1 (2) (1971), pp. 143–186. doi:10.1080/0022250X.1971.9989794.

[26] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen, et al. Massively parallel methods for deep reinforcement learning, arXiv Preprint arXiv:1507.04296, (2015), pp 1507.04296.

[27] J.J. Hopfield, Artificial neural networks, IEEE. Circuits.Devices. Mag 4 (5) (1988), pp. 3–10. doi:10.1109/101.8118.

[28] R.J. Schalkoff, Artificial Neural Networks, Vol. 1, McGraw-Hill, New York, 1997.

[29] B. Yegnanarayana, Artificial Neural Networks, PHI Learning Pvt. Ltd, Delhi, 2009.

[30] S. Haykin and N. Network, A comprehensive foundation, Neural Networks 2 (2004), pp. 41.

[31] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, nat 521 (7553) (2015), pp. 436. doi:10.1038/nature14539.

[32] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, Deep Learning, Vol. 1, MIT press, Cambridge, 2016.

[33] G.P. Zhang, Neural networks for classification: A survey, IEEE. Trans. Syst. Man. Cybern. Part C (Appl. Revi) 30 (4) (2000), pp. 451–462. doi:10.1109/5326.897072.

[34] Z.H. Zhou and Y. Jiang, Medical diagnosis with c4. 5 rule preceded by artificial neural network ensemble, IEEE. Trans. Inf. Technol. Biomed 7 (1) (2003), pp. 37–42. doi:10.1109/TITB.2003.808498.

[35] J. Khan, J.S. Wei, M. Ringner, L.H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C.R. Antonescu, C. Peterson, et al., Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks, Nat. Med. 7 (6) (2001), pp. 673. doi:10.1038/89044.

[36] W.G. Baxt, Application of artificial neural networks to clinical medicine, The Lancet 346 (8983) (1995), pp. 1135–1138. doi:10.1016/S0140-6736(95)91804-3.

[37] H.A. Rowley, S. Baluja, and T. Kanade, Neural network-based face detection, IEEE Trans. Pattern Anal. Mach. Intell. 20 (1) (1998), pp. 23–38. doi:10.1109/34.655647.

[38] S. Lawrence, C.L. Giles, A.C. Tsoi, and A.D. Back, Face recognition: A convolutional neural-network approach, IEEE Trans. Neural Netw. 8 (1) (1997), pp. 98–113. doi:10.1109/72.554195.

[39] J. Kamruzzaman, Artificial Neural Networks in Finance and Manufacturing, IGI Global, Pennsylvania, 2006.

[40] R.S. Govindaraju and A.R. Rao, Artificial Neural Networks in Hydrology, Vol. 36, Springer Science & Business Media, Berlin, 2013.
[41] G. Wen, C.P. Chen, and B. Li, Optimized formation control using simplified reinforcement learning for a class of multiagent systems with unknown dynamics, IEEE. Trans. Ind. Electron. 67 (9) (2019), pp. 7879–7888. doi:10.1109/TIE.2019.2946545.

[42] G. Wen, C.P. Chen, Y.J. Liu, and Z. Liu, Neural network-based adaptive leader-following consensus control for a class of nonlinear multiagent state-delay systems, IEEE. Trans. Cybern 47 (8) (2016), pp. 2151–2160. doi:10.1109/TICYB.2016.2608499.

[43] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Networks 61 (2015), pp. 85–117.

[44] M.T. Hagan, H.B. Demuth, M.H. Beale, et al. Neural Network Design, Vol. 20, Pws Pub, Boston, 1996.

[45] E. Fehr and K.M. Schmidt, Fairness, incentives, and contractual choices, Eur. Econ. Rev 44 (4–6) (2000), pp. 1057–1068. doi:10.1016/S0014-2921(99)00046-X.

[46] G.E. Bolton and A. Ockenfels, Self-centered fairness in games with more than two players, Handb. Exp. Econ. Results 1 (2008), pp. 531–540.

[47] R.O. Murphy and K.A. Ackermann, Social value orientation: Theoretical and measurement issues in the study of social preferences, Pers. Soc. Psychol. Rev 18 (1) (2014), pp. 13–41. doi:10.1177/1088863313501745.

[48] G. Charness and M. Rabin, Understanding social preferences with simple tests, The Quarterly, J. Econ 117 (2002), pp. 817–869.

[49] L.K. Hansen and P. Salamon, Neural network ensembles, IEEE Trans. Pattern Anal. Mach. Intell. 12 (10) (1990), pp. 993–1001. doi:10.1109/34.58871.

[50] A. Krizhevsky, I. Sutskever, and G.E. Hinton, Imagenet classification with deep convolutional neural networks, in Advances in neural information processing systems. Lake Tahoe, 2012, pp. 1097–1105.

[51] A. Krogh and J. Vedelsby, Neural network ensembles, cross validation, and active learning, in Advances in neural information processing systems, Denver, 1995, pp. 231–238.

[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the, J. Mach. Learn. Res. 12 (2011), pp. 2825–2830.

[53] J.M. Epstein and R. Axtell, Growing Artificial Societies: Social Science from the Bottom Up, Brookings Institution Press, Washington, 1996.

[54] A. Al, A. Eiben, and D. Vermeulen, An experimental comparison of tax systems in sugarscape, in 5th Joint Conference on Information Sciences (JCIS 2000), Atlantic City, NJ, USA. Durham, USA: Assoc Intelligent Machinery. 2000, pp. A937–A940.

[55] A. Bigbee, C. Cioffi-Revilla, and S. Luke, Replication of sugarscape using mason, in Agent-Based Approaches in Economic and Social Complex Systems IV, Springer, Heidelberg, 2007, pp. 183–190.

[56] M.E. Newman, Power laws, pareto distributions and zipf’s law, Contemp. Phys. 46 (5) (2005), pp. 323–351. doi:10.1080/00107510500052444.

[57] A. Chatterjee, S. Yarlagadda, and B.K. Chakrabarti, Econophysics of Wealth Distributions: Econophys-Kolkata I, Springer Science & Business Media, Berlin, 2007.

[58] E. Kalai and E. Lehrer, Rational learning leads to nash equilibrium, Econometrica: J. Econom Soc 61 (5) (1993), pp. 1019–1045. doi:10.2307/2951492.

[59] C. Daskalakis, P.W. Goldberg, and C.H. Papadimitriou, The complexity of computing a nash equilibrium, SIAM J. Comput 39 (1) (2009), pp. 195–259. doi:10.1137/070699652.