Bias-generating Agent-based Simulation and its Application to Election Systems

Jiateng Pan[1] , Atsushi Yoshikawa[1] , Masayuki Yamamura[1]

[1]School of Computing, Tokyo Institute of Technology, Japan

Abstract: Agent-based-Simulation can be used to study the impact of individual decisions on the overall outcome. There has been a lot of research focused on giving the agent the ability to learn, especially there are studies that analyze corporate organizations using reinforcement learning. We focus on the election phenomena that there are often cases where popular candidates are suddenly overtaken by others in the election campaign. We try to use the characteristic of overfitting of neural networks and conditional reflex learning, such as the “Pavlov’s dog” effect, to explain the phenomena.

1 Research Background

Agent-based simulation, or ABS for short, is a method that allows a plural number of agents that will interact with each other according to a pre-set process and finally observe the overall changes[1,2,3,4,5,6].

There are also many studies[7,8,11,12,13,14] applying ABS for voting simulation.

Jiménez-Rolland M et al. study[9] suggested that computer simulations can provide valuable insights into the performance of voting methods on different collective decision problems. Meanwhile M.A Wiering et al. simulated the majority voting results using cellular automata[10].

Anaëlle Wilczynski[7] considered the effect of heuristic algorithms on voting results and performed simulations.

But in either case, the thrust of ABS-based research focuses on how to make the "optimal" decision for the agent at all times. This is applicable in many contexts. But in experiments with humans or other intelligence, it may not be so applicable.

Because, human beings are not always "rational", people tend to have some "biases" in the process of learning about the environment, such as treating some random and accidental events as inevitable. It is these biases that make people tend to make emotional decisions.

Therefore, it is difficult to use models to explain why there are often cases where popular candidates are suddenly overtaken by others in the election campaign.

To better perform ABS with human subjects such as voting simulation, it is important to propose an agent learning model that generates "bias" and apply it to ABS.

2 Purpose

The purpose of this study is to develop an agent learning model that generates "bias" and apply it to ABS.

Finally, apply this new ABS to a voting system and observe its effect.

Therefore, this study can be divided into three parts.

Building an agent learning model that generates "bias", building an ABS-based voting model, and combining the two for simulation observations.

3 Bias-generating Agent

3.1 Bias

To build an agent learning model that generates

"bias", it is first necessary to understand "What is bias?".

Lucy Suchman, a professor of anthropology of science and technology at the Department of Sociology at Lancaster University in the United Kingdom, put forward the view that "all human behavior is based on improvisation under certain conditions".

This view implies that people perform actions that are immediate and based on conditioned reflexes in a situation.

Therefore, certain conditioned reflexes that do not correspond to the actual situation in terms of results can be considered as a kind of bias.

From this point of view, the "Pavlov's dog" experiment is a good illustration of the creation of a bias:

Give the dog food along with something special (like a bell). After many times, the dog will salivate even if no food is given, but only the bell is heard.

This is a reflection of a bias (ringing = food -> drooling) that the dog has developed because of a particular circumstance.

I call this phenomenon the "Pavlov's dog" effect.

3.2 Overfitting

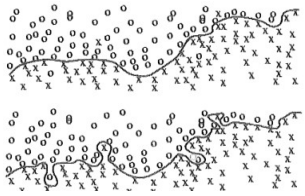

Figure1: Overfitting

Overfitting is a common problem in artificial intelligence learning.

It refers to the fact that excessive learning allows the AI to learn something unnecessary and randomly occurring by chance along with it as correct knowledge.

Usually, in the design of artificial intelligence, we want to avoid the overfitting phenomenon as much as possible.

But learning "random things by chance" as correct knowledge is the same as our perceived bias, so this study would prefer a model that is prone to overfitting.

3.3 Bias-generating Neural network

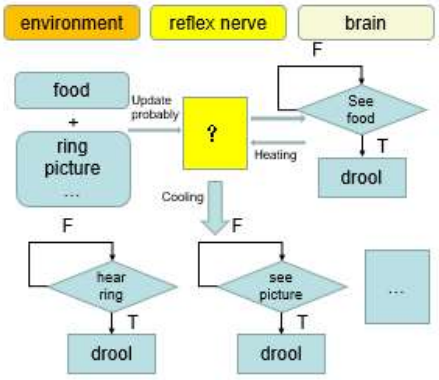

Figure2

In this section, we build a neural network model that uses the overfitting property of neural networks to simulate the "Pavlov's dog" effect.

The dog model is shown in Figure2, divided into two parts, the reflex nerve, and the brain.

The brain simply does the mechanical step of "drooling at the sight of food".

The main focus is on the reflex nerve part.

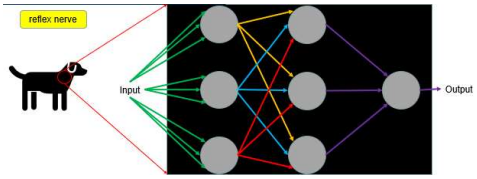

Figure3: neural network

The neural network has 3 inputs, corresponding to the presence or absence of food, bells, and pictures. The output is the presence or absence of drool.

The structure of a single neuron is shown in Figure4. It will accept the input passed from the previous layer and compute the output to the next layer, and will also accept the return value from the next layer and use it to update its weights.

The return value used for neural network backtracking is then the difference between the output of the brain and the output of the neural network.

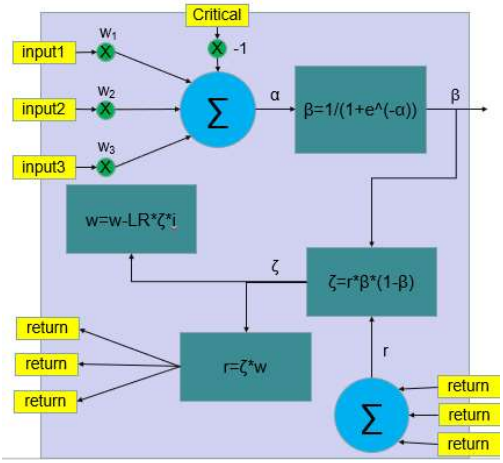

Figure4

At the same time, we not only let the learning rate (LR) be 1 but also each backtracking will be repeated several times for faster overfitting.

3.4 Simulation results

Similar to the real-life "Pavlov's dog" experiment.

For the first input, we gave both food and bell, and after 60 repetitions, we gave both food and picture 100 times, and finally re-given food and bell several times, observing at each moment whether the simulated dog would drool if the bell or the picture were given alone. The results are shown in the Figure5

Figure5

As can be seen from Figure5, the simulated dog will indeed be given "bell" + "food" at the same time several times, and finally do only hear the "bell", will also drool.

And this phenomenon can be changed.

Therefore, it is believed that this neural network can produce the "Pavlov's dog" effect.

4 ABS-based voting model

In this chapter, an ABS-based voting model will be built, in which all agents as voters will vote for the candidates, and the environment will make public the results of counting all agents' voting objects as polls. At the same time, there is an option to tamper with the results of the polls for vote manipulation.

4.1 Agent

Agents will vote based on the results of their social networks and public polls, combined with their confidence level and ranking of their preferences for candidates. The model proposed by Dr. Anaëlle Wilczynski [7] is used here.

4.1.1 Social networks

Social network refers to the set of other agents that an agent can see, where " can see" means that the agent can know the voting objects of other agents in the social network.

The sparseness of social networks affects the agent's reliance on the results of public opinion surveys.

In this study, for convenience, and contrary to Dr. Anaëlle Wilczynski's view, the density of social networks is shown in Figure6.

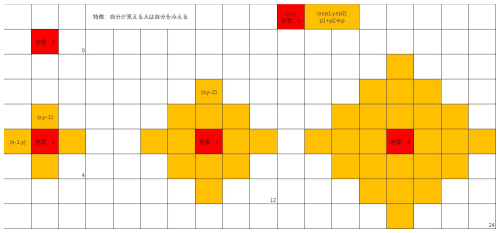

Figure6

This will be a two-way network, using the concept of distance instead of density. When the density is p, the agent's social network is the set of all agents within p distance from itself.

4.1.2 Confidence level

As with the density of social networks, confidence level also affects the agent's voting results.

Each agent combines the results of polls and the

votes of people around them to have a guess, called *TC[ci]*, about the number of candidate ci votes.

And each agent has a preference ranking *R[ci]* for each candidate.

Confidence is used if the value of T*C[ci]* + *confidence* is greater than all other TC[ci], then ci is the possible-winner *PW*.

After identifying all possible winners, the agent will vote for the top possible-winner in the preference ranking *R[ci]*.

4.1.3 Behavior pattern

The behavior pattern of the agent is shown in Argo1.

Argo1：agent_ Think about who to vote for

1. If neither the results of the poll nor the results of the surrounding votes change, the process ends
2. *TC[ci]* was calculated for all candidates by combining the results of public opinion polls and the results of surrounding people's votes
3. For all candidates, judge that the value of *TC[ci]+ confidence* is greater than all other *TC*, if so, add ci to *PW*
4. Find the top ranking in *R[ci]* from *PW* and prepare to vote for him.

4.2 Environment

The environment will count all agent's voting targets and make the poll results public and will tamper with the poll results if necessary, as shown in Argo2.

Argo2： Environment _tamper with voting results

1. Counting the objects voted by each agent at that moment.
2. If no tampering is required, publish the results directly
3. Calculation: Votes that can be manipulated = total number of votes - p(p+1)/2
4. Select a "ghost candidate" , and a "target candidate"
5. Evenly distribute the number of votes that can be manipulated
6. Announcing the results of the tampered ballots

Table1 Parameter table

Parameter	Value
Number of voters	Integer
Number of candidates	Integer
Density	Integer
Confidence level	Integer
Whether to tamper	Boolean
Availability of neural networks	Boolean

4.3 Simulation results

Based on the above steps, six simulations were performed and the results are shown in Figure7.

```
There are 100 voters and 10 candidates in the environment.
Density is 1
Voters don't have neural networks and the confidence level is 1
Voting results will not be tampered with.
The number of wins for each candidate is as follows
   A:0  B:5  C:0  D:0  E:0  F:0  G:0  H:0  I:0  J:0
```

Figure7.1

```
There are 100 voters and 10 candidates in the environment.
Density is 1
Voters don't have neural networks and the confidence level is 1
Voting results will be tampered with.
The GHOST_candidate is J and the TARGET_candidate is H
The number of wins for each candidate is as follows
   A:0  B:1  C:0  D:0  E:0  F:0  G:0  H:4  I:0  J:0
```

Figure7.2

```
There are 100 voters and 10 candidates in the environment.
Density is 7
Voters don't have neural networks and the confidence level is 1
Voting results will not be tampered with.
The number of wins for each candidate is as follows
   A:0  B:5  C:0  D:0  E:0  F:0  G:0  H:0  I:0  J:0
```

Figure7.3

```
There are 100 voters and 10 candidates in the environment.
Density is 7
Voters don't have neural networks and the confidence level is 1
Voting results will be tampered with.
The GHOST_candidate is J and the TARGET_candidate is H
The number of wins for each candidate is as follows
   A:0  B:5  C:0  D:0  E:0  F:0  G:0  H:0  I:0  J:0
```

Figure7.4

```
There are 100 voters and 10 candidates in the environment.
Density is 1
Voters don't have neural networks and the confidence level is 10
Voting results will not be tampered with.
The number of wins for each candidate is as follows
   A:0  B:0  C:0  D:0  E:0  F:0  G:0  H:5  I:0  J:0
```

Figure7.5

```
There are 100 voters and 10 candidates in the environment.
Density is 1
Voters don't have neural networks and the confidence level is 10
Voting results will be tampered with.
The GHOST_candidate is J and the TARGET_candidate is H
The number of wins for each candidate is as follows
   A:0  B:0  C:0  D:0  E:0  F:0  G:0  H:5  I:0  J:0
```

Figure7.6

Comparing 7.1 and 7.5, it can be concluded that the confidence level has an impact on the voting results.

Comparing 7.1 and 7.2, it can be concluded that the heuristic algorithm can induce voters' votes.

Comparing 7.3 and 7.4, it can be concluded that tampering with polls to manipulate voters' votes becomes difficult in the case of a high density of social networks.

5 Bias-generating ABS-based voting model

Differences between this model and the ABS-

based voting model ：

-> Each agent has a Bias-generating Neural network

-> The agent learns after each new poll result is obtained by the neural network.

5.1 Bias-generating Neural network

This neural network is a 3*3 neural network with 3 inputs and 1 output.

The inputs are respectively.

1. Was the candidate he voted elected? 0;1

2.Was the candidate he liked the most elected?0;1

3. Was the candidate with the most votes in his social network elected? 0;1

The output is the confidence level.

The range of self-confidence is between 1 and 10, but the output of the neural network is between 0 and 1.

Thus, we have:

confidence= ⌊ output*10 + 1 ⌋

As mentioned earlier, the confidence level is the agent's prediction of the value of its vote.

So, if the agent voting object does not get to this predicted increase in votes, the confidence should be reduced, and conversely, the confidence should be increased.

We make the change in the number of votes of the agent voter from the last result as △.

So, the return value = (△-confidence)/10

5.2 Voting Process

The process of Bias-generating ABS-based voting model is shown in Argo3.

Argo3：Bias-generating ABS-based voting model

1. Initialize
2. Disclose the initial voting results and skip to step 4
3. Update each intelligence with their confidence level
4. Repeat ***Argo1*** a certain number of times for each intelligence
5. ***Argo2***
6. backtrack for each intelligence and skip to step 3

5.3 Simulation results

Based on Argo3, simulations were performed and the results are as figure8 ：

```
There are 100 voters and 10 candidates in the environment.
Density is 1
Voters have neural networks, and the confidence level is not fixed
Voting results will not be tampered with.
The number of wins for each candidate is as follows
   A:0  B:3  C:0  D:0  E:0  F:0  G:0  H:2  I:0  J:0
```

Figure8.1

```
There are 100 voters and 10 candidates in the environment.
Density is 1
Voters have neural networks, and the confidence level is not fixed
Voting results will be tampered with.
The GHOST_candidate is J and the TARGET_candidate is H
The number of wins for each candidate is as follows
   A:0  B:0  C:0  D:0  E:0  F:0  G:0  H:4  I:0  J:1
```

Figure8.2

Comparing the results of the two simulations, we can see that

Even with a neural network, the heuristic algorithm is still effective in manipulating votes by tampering with the poll results.

However, this sometimes leads to unexpected situations, such as the one on the right, where the ghost candidate J is elected once instead.

I call this situation "tampering that gets out of hand".

Of course, due to the time problem, the number of simulations is small, so it is hard to say that this is not a "special case".

6. Summary

In this study, a Bias-generating Neural network is built and applied to a simple dog model to replicate the "Pavlov's dog" experiment by exploiting the overfitting property of neural networks.

A simple voting model was also built. This model can be used to simulate the winning of a vote under various scenarios (tampering or not, density size of social networks, the number of voters/candidates, etc.).

This voting model shows that heuristic algorithms to manipulate voter voting are effective.

This is in line with the results of the model proposed by Dr. Anaëlle Wilczynski.

At the same time, I applied the neural network model that produces the "Pavlov's dog effect" to this voting model and performed simulations.

Although there is not much data yet, it seems

that the heuristic algorithm has the potential to "get out of hand".

In future work, I will continue to improve the algorithm next while increasing the number of simulations and increasing the randomness of the simulations to get more data.

Acknowledgment

Special thanks to Mr. Takahashi and Mr. Kitazawa. They provided a lot of advice and help in this research. Thanks also to everyone in the Yoshikawa Research Office for their help.

References

[1] Hossein Nasrazadani, Mojtaba Mahsuli. Probabilistic Framework for Evaluating Community Resilience: Integration of Risk Models and Agent-Based Simulation[J]. Journal of Structural Engineering,2020,146(11).

[2] Hwang Yi. Visualized Co-Simulation of Adaptive Human Behavior and Dynamic Building Performance: An Agent-Based Model (ABM) and Artificial Intelligence (AI) Approach for Smart Architectural Design[J]. Sustainability,2020,12(16).

[3]. COVID-19/SARS-CoV-2 News from Preprints; Impact Assessment of Full and Partial Stay-at-Home Orders, Face Mask Usage, and Contact Tracing: An Agent-Based Simulation Study of COVID-19 for an Urban Region[J]. Medical Letter on the CDC & FDA,2020.

[4]. Sociology - Artificial Societies and Social Simulation; Investigators at Technical University of Crete Report Findings in Artificial Societies and Social Simulation (An Agent-based Model for Simulating Inter-settlement Trade In Past Societies)[J]. Politics & Government Business,2020.

[5] 中野統英、周慧：「Q 学習を含むエージェントによエージェントベースシミュレーションにおける流行の流布と初期採用者の配置に関する研究」追手門経営論文集、Vo.20, No.1, pp.27-48,(2014)

[6] Alan Kirman: Learning in Agent-Based Models, Eastern Economic Journal, Vol. 37, No. 1, pp.20-27 (2011)

[7] Wilczynski A. Poll-confident voters in iterative voting[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2019, 33: 2205-2212.

[8] Muñoz D F, Gardida H, Velázquez H, et al. Simulation models to support the preliminary electoral results program for the Mexican Electoral Institute[J]. Annals of Operations Research, 2020: 1-16.

[9]Jiménez-Rolland M, Macías-Ponce J C, Martínez-Álvarez L F. Using simulation in the assessment of voting procedures: An epistemic instrumental approach[J]. SIMULATION, 2020: 0037549720923031.

[10] Dominguez Arroyo P. Artificial Life Simulation of the Majority Vote[D]. , 2020.

[11] Allen T T. Introduction to discrete event simulation and agent-based modeling: voting systems, health care, military, and manufacturing[M]. Springer Science & Business Media, 2011.

[12] Kononovicius A. Empirical analysis and agent-based modeling of the Lithuanian parliamentary elections[J]. Complexity, 2017, 2017.

[13] Riddle R. 2020 US Elections: Modelling Subjectivity in Voting Behaviours Using Active Inference[J]. 2020.

[14] Charcon D Y, Monteiro L H A. A multi-agent system to predict the outcome of a two-round election[J]. Applied Mathematics and Computation, 2020, 386: 125481.