How Well Can Machine Learning Techniques Find an Evolutionarily Stable Strategy for a Multiplayer, Symmetric Game?

Theodore A. Ehrenborg[1]

Henry Clay High School

[1] The author would like to thank Professor Adib Bagh at the University of Kentucky for a discussion regarding a formal definition of the game of Mercantilism.

Abstract

Finding solution concepts in games with more than two players is a difficult problem from a computational complexity perspective. This paper examines whether machine learning techniques such as neural networks and evolutionary computation can find solution concepts in practice. These techniques were tested on a multiplayer game, Mercantilism, that has certain desirable properties. The algorithms that use machine learning, along with several that do not use it, played against each other in a tournament. The data were analyzed with a Bayesian process to determine the algorithms that were evolutionarily stable against others. Algorithms that used evolutionary computation experienced mixed results. Despite the theoretical difficulty of finding a Nash equilibrium, algorithms that used a stochastic process to approximate an equilibrium were generally successful—more successful than algorithms that imitated how a group of human participants played Mercantilism. The most successful algorithm used both the stochastic process and a neural network to estimate the value of various positions in the game. Thus it seems that machine learning will become a powerful tool for analyzing real-world situations modeled by multiplayer games. Moreover, computers are likely to become stronger than humans in a variety of multiplayer games, not just Mercantilism.

How Well Can Machine Learning Techniques Find an Evolutionarily Stable Strategy for a Multiplayer, Symmetric Game?

Introduction

The point of game theory is to look for "solution concepts" (Berg & Sandholm, 2017, p. 383). A solution concept is an outcome for a game "possessing in its entirety some kind of balance and stability" (von Neumann & Morgenstern, 1953, p. 36). For instance, an important solution concept is the Nash equilibrium, which is a set of strategies for all players where "each player is maximizing its expected payoff" (Roughgarden, 2010, p. 205) as long as all players use the Nash equilibrium. In Rock-Paper-Scissors the Nash equilibrium is when both players make random choices, choosing each option with probability one-third. If one players uses this strategy, the other player can do no better than to also use that strategy.

Game theory has become useful in economics and behavioral science, even describing "the development of institutions and social norms" (Royal Swedish Academy of Sciences, 1994). Game theory can also predict how species compete and evolve over time (Maynard Smith, 1974). For example, Conlin et al. (2014) explain how modeling interactions among *E. coli* with a two-player game predicts how much of the enzyme β-lactamase the bacteria produce (p. 37).

Literature Review

Game theory contains two opposing trends: More problems are known to be difficult even as computers solve other problems. This clash of ideas needs to be resolved.

Problems Known To Be Difficult

On one hand, computer scientists have shown that more and more problems in game theory are computationally difficult. Roughgarden (2010) provides a survey of recent results. An "easy" problem is solvable in polynomial time: As the problem becomes more complicated, the time required to solve the does not increase drastically. These

polynomial-time problems are said to belong to the class P (p. 200). A "hard" problem requires an exponentially increasing amount of time to solve, so algorithms to solve them are billions of times slower than algorithms for easy problems (p. 194–195). In games with more than two players, an important solution concept is the strong Nash equilibrium, where no player and no group of players has an incentive to change their choices (Gatti et al., 2013, p. 342). However, finding the strong Nash equilibrium is a hard problem. Specifically, that problem is "NP-complete" (p. 342), which means scientists believe the problem is as difficult as many other hard problems (Roughgarden, 2010, p. 203) Although it is not known that NP-complete problems are difficult, it seems highly likely (p. 209).

Despite this, there are special cases of games that can be solved much more quickly. For example, Challet and Zhang (1997) show how in some cases simple algorithms can converge on a solution. Papadimitriou and Roughgarden (2008) prove that finding solution concepts in succinct games is sometimes an easy problem. Succinct games are games that can be described in a much more concise way than other games (p. 7). For instance, the game of hide-and-seek is succinct: It has the same rules whether two or a million people are playing. In contrast, the rules of chess would be enormously more complicated if there were ten players, let alone a million. This property of succinctness is helpful because it is difficult to analyze a game that is extremely complex. Symmetric games, where players are in identical positions at the beginning of the game, are a type of succinct game (Papadimitriou & Roughgarden, 2008, p. 3; von Neumann & Morgenstern, 1953, pp. 256–257). Rock-paper-scissors is a symmetric game, but chess is not because one player moves first, disturbing the symmetry.

Problems Solved By Computers

On the other hand, artificial intelligence (AI) that analyzes games has become impressively strong. The most notable result is from Google's DeepMind branch (Silver et al., 2018), which created the world's strongest chess player, called AlphaZero, which "achieved superior results within a few hours" (p. 1144) of beginning training. Google's

procedure was similar to prior work by the DeepMind researchers, including Lai's (2015) chess program, called Giraffe. Giraffe was based on neural networks as well as temporal-difference reinforcement learning (p. 23), abbreviated TD. The intent of TD is for the algorithm to receive specific feedback for the moves it plays (p. 14). In neural networks, computers simulate small groups of neurons in order to recognize patterns (Werbos, 1990, pp. 1550–1551). Both neural networks and TD have been successfully applied to a wide variety of games, from Samuel's (1959) checkers player to Tesauro's (1995) backgammon player, which invented tactics previously unknown to human players (p. 64). Evolutionary computation, another field of AI, attempts to teach programs how to "evolve" and thus learn from experience (Fogel, 2005).

However, some games are more difficult for computers to solve. For instance, algorithms have struggled to solve poker because it has more than two players—it is a multiplayer game—and players can hide information from each other (Knight, 2017). Although Brown and Sandholm (2018) recently demonstrated an AI that could play poker better than any human, the AI could only play the two-person version of poker (p. 418). Sandholm (2015) points out limitations in the current understanding of multiplayer games and concludes, "Games with more than two players and that are not zero sum also deserve further attention" (p. 123) from the scientific community.

These two trends overlap in the area of symmetric multiplayer games. AI cannot solve multiplayer games yet, but *symmetric* multiplayer games might be easier to solve than other multiplayer games. The current body of knowledge does not address this topic. Thus it is worthwhile to explore how well machine learning techniques can find a solution concept for a multiplayer, symmetric game. Moreover, the success of these machine learning techniques should be measured through the lens of evolutionarily stable strategies (ESS) in order for the powerful techniques of evolutionary computation to be applicable. An ESS is a more specific type of Nash equilibrium that is thought to occur in nature (Maynard Smith, 1974; Palm, 1984). Although finding an ESS is in general a

difficult problem, "Games encountered in practice may have additional structure" making it easier to find an ESS (Conitzer, 2018, p. 5).

The following definition of ESS is adapted from Palm (1984, p. 331) and Broom et al. (1997, p. 935). A less formal definition is located in the next section.

Definition. Consider a symmetric game with n players. Let p be a strategy in that game. Let $U_1(s_1, s_2, \ldots, s_n)$ be the payoff received by Player 1 in the case where Player 1 plays strategy s_1, Player 2 plays strategy s_2, ..., and Player n plays strategy s_n. One says p is an *evolutionarily stable strategy* when there is a neighborhood Q where $p \in Q$ (that is, there is a continuous collection of strategies Q that includes p on the inside) such that for every strategy q in Q where $q \neq p$, the following is true:

$$U_1(p, q, q, \ldots, q) > U_1(q, q, q, \ldots, q)$$

Moreover, when the preceding inequality is true for a particular strategy q, one says p is *evolutionarily stable* (abbreviated ES) against q. If p is not evolutionarily stable against q, one says that q *invades* p.

Methods

To determine whether machine learning is effective in multiplayer games, I designed a quantitative experiment where algorithms played such a game against each other.

A true evaluation of the strength of algorithms that use machine learning is only possible with a comparison to other algorithms. For example, Silver et al. (2018) measured their machine learning algorithm, AlphaZero, against the leading chess algorithm that did not use machine learning, Stockfish. Before the design and comparison of algorithms for this project could take place, it was essential to choose the framework under which the algorithms' strategies would be compared.

As explained in the literature review, the concept of testing strategies for evolutionary stability has existed for decades (see Maynard Smith, 1974). The formal definition is found at the end of the literature review.

To see the basic idea, consider the game Rock-Paper-Scissors. A computer algorithm that always plays Rock is ES against an algorithm that always plays Scissors, since if almost everyone chooses the former algorithm the latter algorithm performs worse than average. In contrast, the algorithm that always plays Rock is not ES against an algorithm that always plays Paper, since if almost everyone chooses the former algorithm the latter algorithm performs better than average. A virtue of using the solution concept of evolutionary stability is that it enables pairwise comparison of algorithms, as opposed to every algorithm being compared to all other algorithms at once.

It was also important to choose a suitable game. The eventual choice was a modified verision of a game that was intended as a simplistic model of world trade ("The game of mercantilism", n.d.), hence the nickname Mercantilism. A full description of the rules of Mercantilism occurs in Appendix D on page 26. This game has several desirable features:

1. Although less complicated than games such as multiplayer poker, Mercantilism does not have a simple winning strategy once there are more than 3 players. A winning strategy must pick high-valued tokens that no other players pick, which is exactly what the other players are trying to accomplish. Thus the game is challenging for all algorithms, no matter how simple or sophisticated.

2. Mercantilism is a symmetric game, which eliminates the chance that certain players would have a built-in advantage. In chess, for example, White always moves first, which slightly perturbs the balance of power.

3. The game is itself a sequence of similar rounds. Though past rounds do affect future rounds, an algorithm might be able to simplify its model of Mercantilism by viewing it as individual rounds. Moreover, the human participants could more easily understand the rules.

4. Mercantilism forbids communication and collaboration among the players. This feature removes the need for players to need to be able to bargain with and threaten

each other, a task for which computers need specialized programming.

To ensure that the algorithms that used machine learning were challenged with a wide variety of competitors, I staged a small-scale tournament with people playing the game against each other. The procedure for that tournament is outlined in Appendices B, C, and D. The actual results of the tournament were not statistically significant because only five games were played. Still, the players had time to develop their own strategies. Versions of those strategies were then adapted into various algorithms.

After the human tournament was complete, sixteen algorithms were designed and compared pairwise in a tournament inspired by Axelrod (1980). The algorithms can be classified into four groups:[2]

Group 1. The algorithms *best_human_strategy*, *exp_2*, *play_highest*, *power_1*, *power_2*, *power_3*, and *uniform* were provided with simple ideas about how to play Mercantilism, such as *power_1*'s strategy: "play a token with probability proportional to the token's value." These strategies were chosen because they seemed plausible, and *neural_nash* had to be tested against simple algorithms as well as complicated ones. I did not design the algorithm *best_human_strategy*. Of the 5 games the participants played in the tournament, one participant won 4 out of 5 games. Fortunately this participant provided a detailed explanation of his or her strategy, which became the algorithm *best_human_strategy*.

Group 2. The algorithms *round_diff*, *round_points*, *round_winner*, *total_diff*, *total_points*, and *total_winner* had access to a stochastic algorithm invented by Young (1993), which allowed them to predict the state of the game one move ahead. Each of these algorithms had a different static evaluator that judged how favorable a certain game position was.

---

[2] Full information about each algorithm's design can be found in the code for this project, available at https://bit.ly/2Utm4Ih.

Group 3. The algorithm *neural_nash* had access to the same stochastic algorithm as the algorithms in the previous category. It did not have a human-defined static evaluator. Instead it used backpropagation (see Werbos, 1990) to train its neural network to judge the relative value of game positions. Since backpropagation is faster than the slow pace of artificial selection used by *neural_evolve*, in theory *neural_nash* could rapidly become a stronger player. To generate better estimates of the value of game positions, the algorithm looked at its results one move ahead, which is temporal-difference reinforcement learning.

Group 4. The inspiration for these algorithms, *quick_evolve* and *neural_evolve*, came from Fogel's (2005) idea of evolutionary computation. The algorithm *neural_evolve* was only given the rules of Mercantilism, so it learned by playing games against versions of itself. The least successful versions of the algorithm—the ones that lost the most games—were deleted. Then the surviving versions created imperfect copies of themselves, where each copy played the game similar to how the original played it. The new generation began playing games again, continuing the cycle. Ideally, the surviving versions would gradually become better at playing Mercantilism. The versions' strategies came from neural networks with different weights.

Although *quick_evolve* lacked a neural network, it had access to the algorithms in Group 1 and could judge their performance. Using artificial selection, it determined how often it should do what *best_human_strategy* would do, how often it should do what *exp_2* would do, and so on.

After collecting trials, the program[3] had to decide whether one algorithm had a statistically significant lead over another, using Bayesian probability. Here is an appropriate comparison: Imagine a coin which may or may not be a fair coin, where a fair coin has exactly a 50% probability of coming up heads and a 50% probability of coming up tails. If the coin is flipped and comes up heads, the coin might be a fair coin, but it might

[3] Most of the computer code for this project was written in Python, but SageMath and TensorFlow were used as well.

also be a biased coin that always comes up heads. However, if the coin is flipped 20 times and always comes up heads, it is quite likely that the coin is biased. In fact, a fair coin has less than a one in a million chance of coming up heads 20 times in a row. The same principle applies to calculating probabilities in this project. If one algorithm establishes a enormous lead over another algorithm after only a few dozen trials, then that lead is statistically significant. If one algorithm establishes a small lead over another algorithm, that lead only becomes statistically significant after hundreds of trials. Since the computer could run trials without interruption, collecting hundreds of trials did not pose a problem.

Only when the sixteen algorithms were ready for testing could a precise hypothesis be made. Assuming that the machine learning techniques worked as well as they have on 2-player games in the scientific literature, the *neural_nash* algorithm ought to discover how to play Mercantilism at least as well as a human could.

Thus the following hypothesis seems credible:

Hypothesis. The algorithm *neural_nash* will be ES against all other algorithms tested.

Results

As there were 16 algorithms, there were $2 \cdot \binom{16}{2} = 240$ comparisons to be made pairwise among them. If the result of a comparison between two particular algorithms was statistically significant after 640 games, the computer stopped conducting trials for that comparison. If the result of a comparison was not significant, the computer continued conducting trials until it reached significance or 8900 games, whichever came first.[4]

Statistical significance means that there was either a greater than 99% chance that one strategy was ES against another, or there was a less than 1% chance that one strategy was ES against another. Almost 560000 games were played in total.[5] Statistical significance was calculated by the Bayesian process outlined in the Methods section. In practice, the

[4] Some comparisons did not become significant. See the limitations on page 14.

[5] The complete dataset may be requested from the researcher at game.of.mercantilism@gmail.com

acceptable margin of error was proportional to the square root of the number of trials.

For about half of the comparisons, it was easy to reach statistical significance. After 660 games where *neural_nash* tried to invade *best_human_strategy*, *neural_nash* had won 83.5% of the points despite only controlling one out of five players. Thus *best_human_strategy* was not ES against *neural_nash*, with a 1 in 10^{32} chance that this result was mistaken.

Surprisingly, many algorithms seemed to play at about the same strength, leading to the high number of statistical ties. After 1040 games where *power_3* tried to invade *quick_evolve*, *power_3* had won 19.98% of the points and controlled 20% of the players. Thus there was only a 50.4% chance that *quick_evolve* was ES against *power_3*, which is barely different than the prior probability of 50%.

Two more terms are useful. When an algorithm is *defending*, it is being tested for being ES and controls all but one players. When an algorithm is *attacking*, it is being tested for ability to invade and controls only one player. Remember that saying that algorithm X invades algorithm Y is equivalent to saying that algorithm Y is not ES against algorithm X. According to the literature, it is vastly more important to determine how well an algorithm defends, and the defensive data thus have greater weight in the conclusion. However, measuring algorithms' defensive ability naturally produces data on the algorithms' offensive ability.

The results of these 240 comparisons are presented starting on page 21 in Tables A1 and A2, which are organized by the defending algorithm and attacking algorithm respectively. Tables A1 and A2 are summarized in Figures 1 and 2.

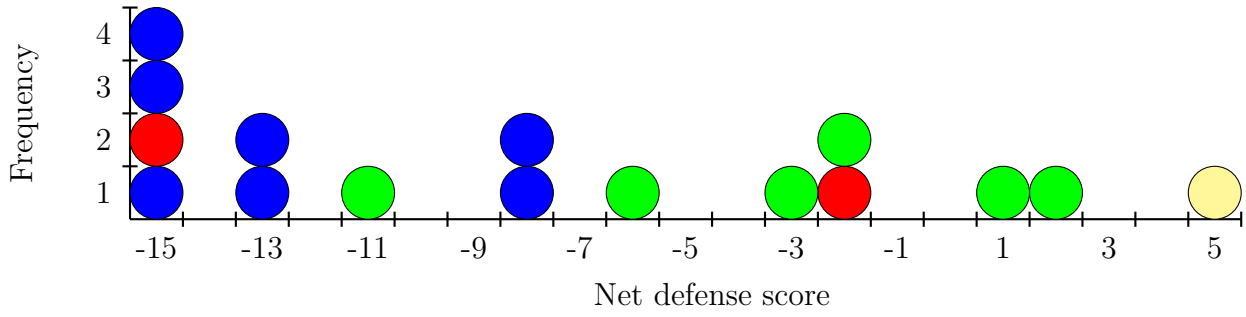

Figure 1. Net defense scores of the 16 algorithms. The colors blue, green, yellow, and red refer to Groups 1, 2, 3, and 4, respectively.

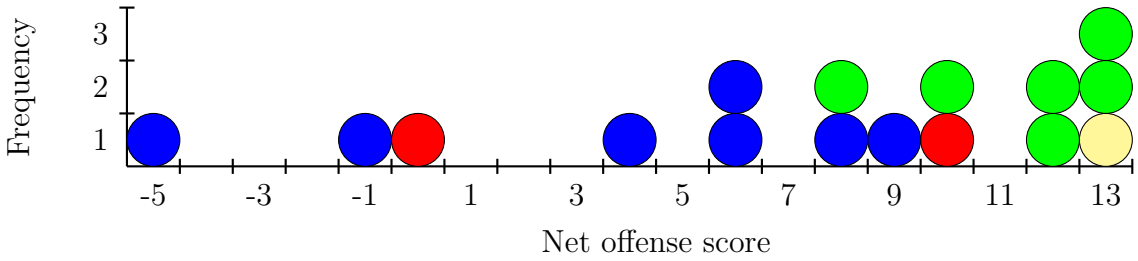

Figure 2. Net offense scores of the 16 algorithms. The colors blue, green, yellow, and red refer to Groups 1, 2, 3, and 4, respectively.

Analysis

The data show that it is harder for algorithms to be ES than to invade other algorithms. Most of the net defense scores are negative, while most of the net offense scores are positive.

The hypothesis is incorrect, in that *neural_nash* is ES against only eight other algorithms, not all of the other fifteen, and is not ES against three algorithms. However, no other algorithm has a net defense score of more than two, so *neural_nash* is the strongest algorithm by this measure.

It is revealing to analyze the algorithms based on their classification.

Group 1: Among these algorithms, *power_3* and *power_2* are the best at defense, with a net score of -8, and *power_3* is the best at offense, with a net score of 9. Many

of these algorithms have significant weaknesses, since *best_human_strategy*, *play_highest*, and *uniform* are not ES against any of their opponents. It is fortunate that none of these simple strategies play Mercantilism extremely well, since I can thus infer that the game is a fair test of the other algorithms' performances. After all, it is more meaningful for a computer to teach itself a difficult game like chess than for it to teach itself Tic-Tac-Toe.

Group 2: Most of these algorithms place in the top half of all algorithms. The only exception is *round_winner*, which has a low net defense score of -11. The algorithms *total_diff* and *total_points* are tied for the best among all algorithms at offense, with a net offense score of 13. Among the algorithms in Group 2, *total_diff* is the best at defense, with a net score of 2.

Group 3: The algorithm *neural_nash* trained itself well, seeing as it is ES against eight other algorithms and has the highest net defense score among all algorithms. It is tied for first place among all algorithms for proficiency in offense, with a net offense score of 13. Since the only design difference between *neural_nash* and algorithms in Group 2 is *neural_nash*'s use of temporal-difference reinforcement learning with neural networks, the difference in performance must be due to that machine learning.

Group 4: During *neural_evolve*'s training, it tended to focus on only choosing the second-highest token—an unusual and ineffective strategy—although it eventually moved away from that strategy. Still, *neural_evolve* is ES against no other algorithms and has a net offense score of 0, the third worst score. In contrast, *quick_evolve* plays much better than *neural_evolve* and better than any algorithm in Group 1, despite all of its strategies being borrowed from Group 1. Thus it has a net defense score of -2 and a net offense score of 10.

Limitations

1. According to Maynard Smith (1974), for any two genuinely different algorithms A and B, either algorithm A is ES against algorithm B, or algorithm A is not ES against B. There is no middle ground. Thus the statistical ties—which make up about 16% of the data in Tables A1 and A2—are a result of time constraints, not a sign that the algorithms are exactly evenly matched. It was computationally infeasible to collect the large number of trials (perhaps in the millions) that would eliminate all statistical ties.

2. The algorithms are ranked by their net offense and defense scores. This choice makes intuitive sense, but the net scores do not measure individual differences. For instance, *neural_nash* is the strongest algorithm at defense, while *uniform* is one of the weakest. The fact that *round_points* invades *neural_nash* is a stronger testament to *round_points*'s offensive ability than the fact that *round_points* invades *uniform*.

3. To draw conclusions about the strength of various algorithms, I chose algorithms that were representative of all conceivable algorithms. It is possible that a larger set of algorithms would generate different results, although the wide-ranging literature review reduces the risk that an important algorithm was missed.

Implications

Surprisingly, the idea most vindicated by the data is not machine learning in general but instead Young's (1993) heuristic for estimating Nash equilibria. Six of the seven best algorithms (ranked by net defense score or net offense score) use this stochastic method to look one move ahead and choose a token. Approximations like this seem to be a feasible way of avoiding the innate difficulties in calculating equilibria in multiplayer games (see Gatti et al., 2013). The only algorithm in the top seven that does not use this stochastic method is *quick_evolve*. Its success and *neural_evolve*'s failure provide mixed support for Fogel's (2005) concept of evolutionary computation.

Since *neural_nash* has the highest net defense score, neural networks combined with temporal-difference reinforcement learning have again proved to be a valuable tool in choosing strategies, as was the case for Lai (2015). Perhaps Brown and Sandholm (2018) will use a variant of these techniques to extend their work to multiplayer poker.

This prediction comes with a caveat: Although Silver et al. (2018) used neural networks to create an algorithm that was significantly stronger than the competing chess algorithms, winning 25 times as many games as it lost (p. 1143), *neural_nash* is not an order of magnitude stronger than its competitors. Perhaps neural networks are not as useful if one does not have Google's computing power, or perhaps multiplayer games are a more difficult problem. In fact, the combination of neural networks and evolutionary computation led to the remarkably unsuccessful algorithm *neural_evolve*, which is about as strong as *uniform* and *power_1*, both of which choose tokens nearly at random. Thus machine learning techniques are only successful in Mercantilism when they are used within a human-designed framework, as is the case for *neural_nash* and *quick_evolve*. Without these auxiliary ideas, such as Young's (1993) stochastic method, machine learning is barely better than random guessing.

Future Steps

1. With more computing power, it will be possible to resolve all the statistical ties, which will simplify the data analysis.

2. One way to improve on the net offense and defense scoring system is through a variant of Google's PageRank algorithm, originally created to rank websites by importance (see Page et al., 1998). PageRank will require a little modification to take into account the fact that each program that plays Mercantilism has a separate score for offense and defense. However, PageRank will provide a more sophisticated ranking of the algorithms, examining each of the 240 comparisons independently.

3. In theory, *neural_evolve* would be able to simulate almost any other algorithm

because of the adaptability of its neural network. In practice, *neural_evolve* is unable to find any successful strategies. It may be possible to improve *neural_evolve* so that it could generate many different strong algorithms that are representative of all possible algorithms. This new group could be tested against *neural_nash*, providing an even more objective way to measure its proficiency.

4. With a stronger theoretical foundation, it may be possible to extend Monte Carlo Tree Search (see Chaslot et al., 2008) to symmetric games like Mercantilism. This search method would enable the computer to look more than one move ahead, an advantage over Young's (1993) heuristic. It could also be combined with machine learning techniques just as Silver et al. (2018) did for chess.

Conclusion

The literature review explored the conflict between the increasing strength of AI and the growing number of intractable game theory problems. Both of these perspectives affected my results. It would have been much easier for the algorithms to play Mercantilism if they could calculate the strong Nash equilibrium. Since that calculation is infeasible (see Gatti et al., 2013), algorithms in Groups 2 and 3 must rely on Young's (1993) heuristic for finding Nash equilibria. Based on the results, this heuristic is effective, although it cannot create algorithms that play as flawlessly as DeepMind's AlphaZero did for chess (Silver et al., 2018). Thus my original hypothesis is too optimistic. Perhaps this is due to the inherent nature of the definition of ES. Maynard Smith (1974) notes that not all games have strategies that are ES against all other strategies (p. 213).

However, several algorithms are able to play Mercantilism well. More sophisticated algorithms, such as *total_diff*, generally perform better than less sophisticated algorithms like *power_1*. In fact, the algorithm that is ES against the greatest number of other algorithms, *neural_nash*, used machine learning.

This result answers the research question: Machine learning is not a panacea for

finding evolutionarily stable strategies, but it is an indispensable tool for developing the most successful algorithms. Sandholm (2015) was uncertain about whether "finding a Nash equilibrium is the right goal in such [multiplayer] games" (p. 123). At least in symmetric games like Mercantilism, the answer appears to be yes.

The future significance of this research seems promising. If scientists wish to simulate a system with many agents (e.g. economists examining the stock market), they can greatly enhance their analysis by assuming symmetry (e.g. all investors start with the same resources) and using machine learning. Since all the algorithms directly based on human strategies are mediocre at offense and weak at defense, it will be possible in the near future to create algorithms that can learn to play some multiplayer games better than any humans can.

References

Axelrod, R. (1980). Effective choice in the prisoner's dilemma. *The Journal of Conflict Resolution*, *24*(1), 3–25. https://doi.org/10.1177/002200278002400101

Berg, K., & Sandholm, T. (2017). Exclusion method for finding Nash equilibrium in multiplayer games. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*, 383–389. https://www.cs.cmu.edu/~sandholm/exclusionMethod.aaai17.pdf

Broom, M., Cannings, C., & Vickers, G. T. (1997). Multi-player matrix games. *Bulletin of Mathematical Biology*, *59*(5), 931–952. https://doi.org/10.1007/BF02460000

Brown, N., & Sandholm, T. (2018). Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, *359*(6374), 418–424. https://doi.org/10.1126/science.aao1733

Challet, D., & Zhang, Y. C. (1997). Emergence of cooperation and organization in an evolutionary game. *Physica A: Statistical Mechanics and its Applications*, *246*(3–4), 407–418. https://doi.org/10.1016/S0378-4371(97)00419-6

Chaslot, G. M. J.-B., Winands, M. H. M., Herik, H. J. V. D., Uiterwijk, J. W. H. M., & Bouzy, B. (2008). Progressive strategies for Monte-Carlo tree search. *New Mathematics and Natural Computation*, *4*(3), 343–357. https://doi.org/10.1142/S1793005708001094

Conitzer, V. (2018). The exact computational complexity of evolutionarily stable strategies. *arXiv*. https://arxiv.org/abs/1805.02226

Conlin, P. L., Chandler, J. R., & Kerr, B. (2014). Games of life and death: Antibiotic resistance and production through the lens of evolutionary game theory. *Current Opinion in Microbiology*, *21*, 35–44. https://doi.org/10.1016/j.mib.2014.09.004

Fogel, D. B. *Evolutionary computation: Toward a new philosophy of machine intelligence*. 2005. https://doi.org/10.1002/0471749214.

The game of mercantilism. (n.d.). https://nj01001706.schoolwires.net/cms/lib/NJ01001706/Centricity/Domain/372/Rules%20for%20the%20Game%20of%20Mercantilism.docx

Gatti, N., Rocco, M., & Sandholm, T. (2013). Algorithms for strong Nash equilibrium with more than two agents. *Proceedings of the AAAI Conference on Artificial Intelligence*, *27*, 342–349. https://www.cs.cmu.edu/~sandholm/algorithms%20for%20SNE%20with%20more%20than%202%20agents.aaai13.pdf

Knight, W. (2017). Why poker is a big deal for artificial intelligence. *MIT Technology Review*. http://www.technologyreview.com/s/603385/why-poker-is-a-big-deal-for-artificial-intelligence

Lai, M. (2015). Giraffe: Using deep reinforcement learning to play chess. *arXiv*. https://arxiv.org/abs/1509.01549

Maynard Smith, J. (1974). The theory of games and the evolution of animal conflicts. *Journal of Theoretical Biology*, *47*(1), 209–221. https://doi.org/10.1016/0022-5193(74)90110-6

Page, L., Brin, S., Motwani, R., & Winograd, T. (1998). The PageRank citation ranking: Bringing order to the Web. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.38.5427&rep=rep1&type=pdf

Palm, G. J. (1984). Evolutionary stable strategies and game dynamics for n-person games. *Journal of Mathematical Biology*, *19*(3), 329–334. https://doi.org/10.1007/BF00277103

Papadimitriou, C., & Roughgarden, T. (2008). Computing correlated equilibria in multi-player games. *Journal of the Association for Computing Machinery*, *55*(3), 1–29. https://doi.org/10.1145/1379759.1379762

Roughgarden, T. (2010). Computing equilibria: A computational complexity perspective. *Economic Theory*, *42*(1), 193–236. https://doi.org/10.1007/s00199-009-0448-y

Royal Swedish Academy of Sciences. *The Sveriges Riksbank prize in economic sciences in memory of Alfred Nobel 1994: Press release*. 1994. http://www.nobelprize.org/prizes/economic-sciences/1994/press-release/

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, *3*(3), 210–229. https://doi.org/10.1147/rd.33.0210

Sandholm, T. (2015). Solving imperfect-information games. *Science*, *347*(6218), 122–123. https://doi.org/10.1126/science.aaa4614

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, *362*, 1140–1144. https://doi.org/10.1126/science.aar6404

Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, *38*(3), 58–68. https://doi.org/10.1145/203330.203343

von Neumann, J., & Morgenstern, O. (1953). *Theory of games and economic behavior* (3rd ed.). Princeton, NJ, Princeton University Press.

Werbos, P. (1990). Backpropagation through time: What it does and how to do it. *Proceedings of the Institute of Electrical and Electronics Engineers*, *78*(10), 1550–1560. https://doi.org/10.1109/5.58337

Young, H. P. (1993). The evolution of conventions. *Econometrica*, *61*(1), 57–84. https://doi.org/10.2307/2951778

Appendix A

Tables

Algorithm's name	ES against	Statistical tie	Not ES against	Net defense score
neural_nash	8	4	3	5
total_diff	5	7	3	2
total_winner	6	4	5	1
quick_evolve	3	7	5	-2
total_points	4	5	6	-2
round_points	4	4	7	-3
round_diff	3	3	9	-6
power_2	2	3	10	-8
power_3	3	1	11	-8
round_winner	2	0	13	-11
exp_2	1	0	14	-13
power_1	1	0	14	-13
best_human_strategy	0	0	15	-15
neural_evolve	0	0	15	-15
play_highest	0	0	15	-15
uniform	0	0	15	-15

Table A1

Proficiency in defense: The algorithms sorted by their net defense score, which is the difference between the number of other algorithms they were ES against and the number they were not ES against. Algorithms with the same score are sorted alphabetically.

Algorithm's name	Invaded	Statistical tie	Did not invade	Net offense score
neural_nash	14	0	1	13
total_diff	13	2	0	13
total_points	13	2	0	13
round_diff	12	3	0	12
round_points	12	3	0	12
quick_evolve	11	3	1	10
round_winner	12	1	2	10
power_3	10	4	1	9
best_human_strategy	10	3	2	8
total_winner	10	3	2	8
exp_2	9	3	3	6
power_2	7	7	1	6
play_highest	8	3	4	4
neural_evolve	7	1	7	0
power_1	7	0	8	-1
uniform	5	0	10	-5

Table A2

Proficiency in offense: The algorithms sorted by their net offense score, which is the difference between the number of other algorithms they invaded and the number they could not invade. Algorithms with the same score are sorted alphabetically.

Appendix B

Informed Consent Form for Survey Participants

I am doing research on how well computers can play games that have more than two players. I would like to compare computers' strategies to humans' strategies.

I have selected you from all students at this high school who are at least 18. You do not have to participate in this study. You may leave at any time.

By signing this form, you agree that you have chosen to participate in this research. You release me and this high school from all liability. Of course, this study is not going to involve any unusual dangers. This study will be just as safe (but less boring) as sitting in a classroom doing nothing.

I will now orally explain the rules of the game. If you are unsure about anything, please ask questions. This study will not take more than 40 minutes of your time.

I may refer to you by your randomly assigned number in my paper. Barring exceptional circumstances, I will not divulge any information that could be used to personally identify you, such as your name or phone number. I will keep this personally identifying information confidential until 100 years from today. After my project concludes, I will contact you with the final results. In unexceptional circumstances, I can only release your personal information with your written permission.

Please fill out the following information:

Your name (printed): _____

Signature: _____

Age: _____

Phone number, email, or other way of contacting you: _____

_____

School that you attend: _____

Today's date: _____

Keep the following pieces of information secret from other participants.

Your randomly assigned number is _____

Your randomly assigned code name is _____

After playing the game, please answer the following questions: What strategies did you follow while playing the game? What strategies do you think would work best for playing the game? Try to be as specific as possible. (For example, if you chose tokens at random, were you equally likely to pick any token or were you more likely to pick certain tokens? If so, which ones?)

_____

_____

_____

_____

_____

_____

_____

_____

Appendix C

Form to Record Game Moves

Your number: _____

Your code name: _____

Round Number	Your Move in Game 1	Your Move in Game 2	Your Move in Game 3	Your Move in Game 4	Your Move in Game 5
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					

Appendix D

Script

Welcome to this study. Please take the time to read these consent forms. Do you have any questions? I will now explain this game:

There are 5 players sitting around a table. On the table there are 15 different tokens, labeled the numbers 1 through 15. The number of players, the number of tokens, and their respective values are common knowledge. Simultaneously, each player selects one token. If a token is selected by just one player, that player receives the token and points equal to the value of that token. If a token is selected by no one, nothing happens to the token. If a token is selected by more that one player, it is removed from the table and no player receives its value.

After players complete one round, they play another round with the tokens remaining on the table. The game ends once there are no tokens left on the table. Once the game ends, each player has as many points as the sum of the values of the tokens they have collected. Any player who has at least as many points as everyone else wins.

Technically, each player is trying to maximize their expected utility. Players who do not win receive a utility of 0. If there are w winners, each winner receives a utility of $\frac{5}{w}$. Thus it is to a player's advantage to not only win, but also ensure that other players do not win. Utility points are not the same as the points received from tokens. The latter determines the former. A good algorithm collects high utility from a game (on average), not necessary many points from tokens. Similarly, a good chess player wins a high percentage of games—a high utility—which is not the same as capturing many pieces in a game.

Players cannot communicate with each other. The players' past moves and number of points are common knowledge. It is common knowledge that each player wants to maximize their chance of winning and is rational.

Do you have any questions?

We will now begin the game. I will write the available tokens on the board. During

each round, you will write your move—the token you select—in the table. For the first game, you will fill out the first column. The other columns are intended for other games. You will have about 15 seconds to decide on a move. I will collect the forms after every round, record players' moves on the board, and then return the forms to you.