AgentRenting2024

Redmar Bakker
Bjarne Daems
Mick Elshout
Daan van Loenen
Yoav Vaizman

Utrecht University

April 2024

Abstract

This report details the working of AgentRenting2024, hereafter referred to as AgentRenting. A negotiation agent build using the NEGMAS framework for bilateral negotiation. AgentRenting uses two strategies depending on the order of the negotiation. Both strategies are accompanied by a neural network which predict the opponent's reservation value during the negotiation. When AgentRenting is the first to offer, it bases its offers on the concession its opponent makes and the passage of time. When it is the last to make an offer, it concedes based on time and uses the predicted opponent's reservation value to determine the best final offer. Multiple experiment were run to compare AgentRenting's performance to agents used in previous ANAC competitions. Lastly, implications and improvements are discussed.

1 Introduction

Each year the ANAC (Automated Negotiation Agents Competition) sets out a challenge ([1] & [2]). For 2024, the challenge is:

"Design a negotiation agent for a bilateral negotiation that has access to its own utility and its opponent utility, but not its opponents reservation value."

With these constraints in mind, the strategy of AgentRenting2024, hereafter referred to as AgentRenting, makes use of the opponent's utility function to estimate the opponents reservation value. We hypothesise that having the final offer impacts the negotiation strategy. This lead us to employ two different strategies, depending on whether AgentRenting has the final offer or not.

To estimate the opponent's reservation value, AgentRenting uses a neural network (NN). During the negotiation, the NN predicts the reservation value. After reaching a certain level of certainty, the prediction is used by both strategies in their own ways.

In the following section (section 2) a detailed description can be found of each component of AgentRenting.

2 Description

This section details the theory and practical implementation of AgentRenting. Each subsection handles a component of the BOA framework proposed by [3].

2.1 Opponent model

2.1.1 Concession Curve

AgentRenting performs linear regression on the opponent's concession curve to estimate what offers the opponent is likely to make. The idea comes from the strategy used for IAmHaggler, the agent which finished

3rd place in the 2010 Automated Negotiating Agent Competition [4]. The last five offers done by the opponent are fitted to a curve with a function of the following form:

$$P(t) = MinUtility + (1 - F(t)) * (MaxUtility - MinUtility) \tag{1}$$

Where P is the the utility of the expected offer done by the opponent at round t, $MaxUtility$ is the offer which gives the opponent the highest utility in a negotiation and $MinUtility$ is the utility that we expect the opponent to concede to in the end, i.e. the utility they will receive from their (predicted) final offer. $F(t)$ is the concession function as a function of time t

$$F(t) = (t/T)^{1/e} \tag{2}$$

Where t is the current round in the negotiation, T is the total negotiation time and e is the 'tactic' variable used as defined by [6].

This form of function took inspiration from the agent HardHeaded [7], which won the ANAC 2011. The same form is used for AgentRenting's own concession curve, used in its bidding strategy, which will be elaborated on later.

$MaxUtility$ can easily be found by iterating over all possible offers in the bidding space. $MinUtility$ and e are the parameters which are found using least squares regression. This is performed by the `curve_fit` function, imported from Python package `scipy.optimize`. $F(t)$ decides the rate of concession of the concession curve, which depends on the value set for e,. Therefore, assuming the opponent to do offers with, for them, descending utilities, any concession curve starting at $MaxUtility$ can be modeled. If the opponent's offers do not adhere to this constraint, the chance increases that the `curve_fit` function will fail to fit the offers to a concession curve of this form.

This curve fitting is performed in AgentRenting by the function `opponent_final_offer`

2.1.2 Neural Network

AgentRenting utilizes a neural network to predict the reservation value of the opponent. It incorporates bid history, including utilities and response times, along with statistical features of the negotiation (up to that point) and the estimated concession curve of the opponent. Since the model is employed throughout the entire negotiation, bid history and statistical features may not always provide a sufficiently precise basis for determining the reservation value. Therefore, we utilize the model only in the final 10 percent of the negotiation. This approach also offers performance advantages, as we have encountered issues with reaching time limits due to the significant time consumption of predictions.

2.1.2.1 Architecture

The neural network (NN) processes two types of inputs: (1) the bid history and (2) the statistical features. This distinction allows us to apply specific functions for interpreting these two data types. The bid history is handled using a Long Short-Term Memory (LSTM) function, a variant of a recurrent neural network (RNN) model, which is effective in learning from three-dimensional step data. Since every prediction is based on all previous bids (a list of lists), this function proves to be highly suitable. The statistical features are processed using the Dense function, a fundamental layer function for NNs. Both inputs are fed into layers of 64 neurons, followed by a second hidden layer of 32 neurons, then a third layer of 32 neurons, and finally, the output layer with 29 nodes. The model predicts the reservation value in categories, dividing the range of the reservation value into steps of 0.025 from 0.0 up to 0.7, with the last step (0.7 - 1.0) forming the final category. These categories allow the model to choose from 29 possible outcomes. The loss function is a modified version of categorical cross-entropy, introducing additional loss based on the deviation from the true label. This encourages the model to predict values as close to the actual reservation value as possible, rather than merely optimizing for accuracy. AgentRenting's strategy benefits more from a directionally correct prediction rather than one that is either exactly right or completely wrong.

$$L_{adjusted} = -\sum_{i=1}^{C} y_{true,i} \log(y_{pred,i}) + |argmax(y_{true}) - argmax(y_{pred})|_n \tag{3}$$

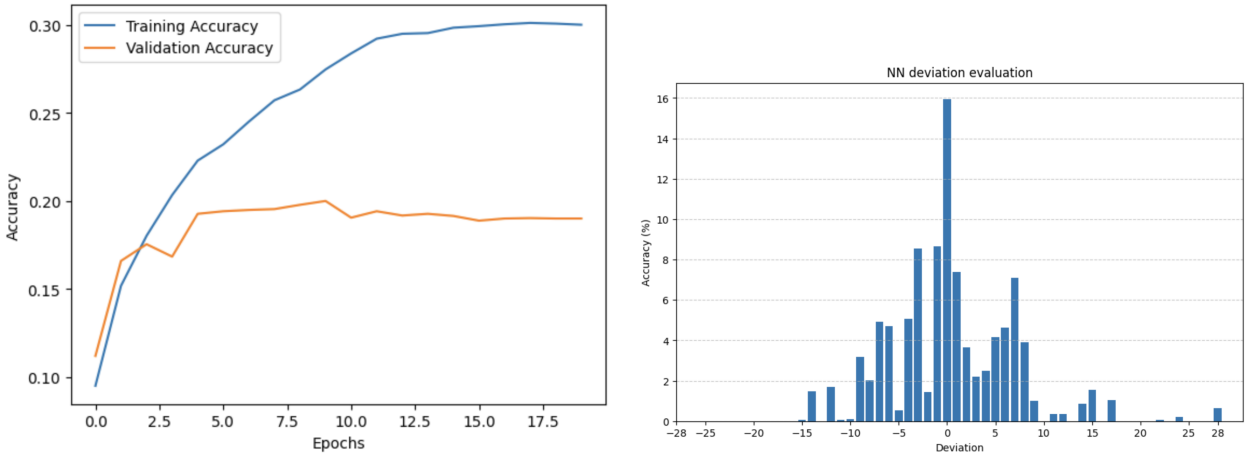
To improve the model, several best practices were employed. Normalization was applied to enhance training efficiency and accuracy. To prevent overfitting, we divided the datasets into training (70 percent), validation (15 percent), and testing (15 percent) datasets, introduced two dropout layers between the hidden layers, and employed a reduce on plateau technique. This method decreases the learning rate of the Adam optimizer when a plateau is reached in the validation dataset's performance during training. The testing dataset is subsequently used to evaluate the model on unseen data, determining its final accuracy.

2.1.2.2 Data construction and training

The model was developed in two phases: (1) the Java phase and (2) the Python phase. Initially, we created a modified version of the ANL tournament function that collected data from negotiations held during the tournament. Then, depending on the phase, tournaments were conducted with selected agents to gather negotiation data. Following this, a script was utilized to construct a trainable dataset from the collected data, including necessary bid history, statistical features for the model and the estimated opponent's concession curve. In the Java phase, approximately 9,000 negotiations were simulated, yielding around 135,000 data points (negotiation steps). For the Python phase, around 4,000 negotiations were conducted, resulting in approximately 480,000 data points. The last dataset is bigger, since we had configured this tournament to run with longer negotiations to mimic the real tournament more.

The first phase aimed to acquaint the model with the negotiation process and several successful strategies. The agents selected for this phase were partially based on a paper by Ragzeghi et al., which discussed the use of deep reinforcement learning to develop a negotiation agent for the ANL contest. The chosen agents, including Yushu, ParsAgent3, Nozomi, IAMhaggler2011, AgentSmith2016, AgentFSEGA, and HardHeaded, were top-rated in previous contest editions, representing a diverse range of frequently employed negotiation styles.

The second phase introduced adjustments for the 2024 edition of the tournament, which is crucial as the 2024 edition replaces time limits with step limits, significantly altering some dynamics. For this phase, the model was retrained with data from the Python-based ANL2024 negotiation tournament, using the standard ANL2024 agents NaiveTitForTat, RVFitter, NashSeeker, MiCRO, Boulware, Conceder and Linear, and AgentRenting with phase 1 version of the NN model. The results are shown in Figures 1 & 2



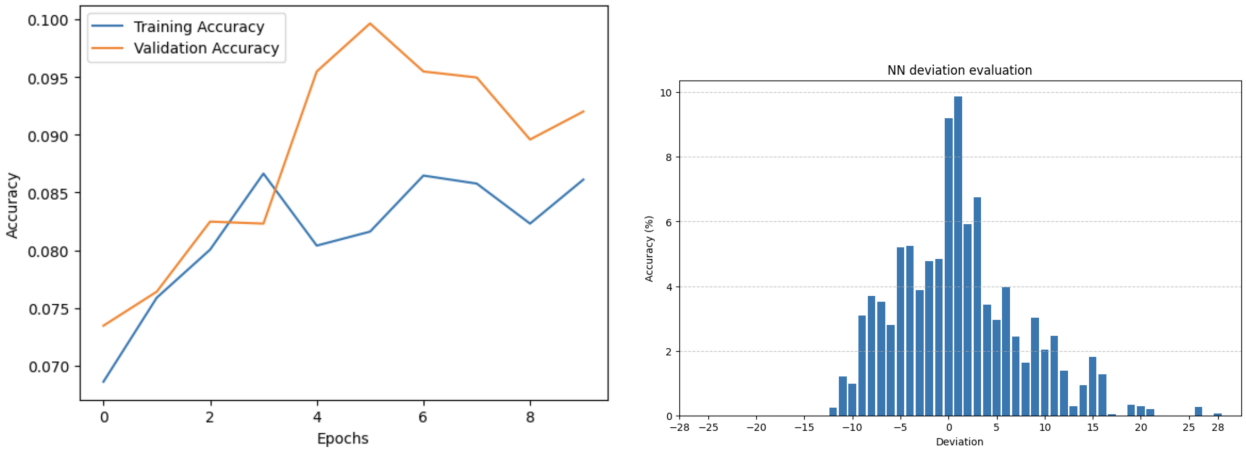
(a) Training

(b) Evaluation

Figure 1: Phase 1

2.2 Bidding strategy

The bidding strategy of AgentRenting is dependent on having the final offer in a negotiation. We believe that the agent which gets to do the final offer has an advantage. When the negotiation comes down to



(a) Training

(b) Evaluation

Figure 2: Training Phase 2

the final offer, the ability to choose what this offer will be rather than whether to accept it is vital, as it can be expected that most agents will accept any offer above their reservation value if the alternative is no agreement.

In the first subsection, the bidding strategy when having the final offer is described. In the second subsection, the bidding strategy when the opponent has the final offer is described. In the code of AgentRenting, this is represented as `bidding_strategy`.

2.2.1 Final offer

Given the nature of a negotiation, a deal has to be struck where both agent have to slowly concede to each other. [5] proposed 3 different conceding strategies for bilateral negotiations:

- Resource-dependent
- Behaviour-dependent
- Time-dependent

In the context of ANAC 2024, a resource conceding strategy is nonrelevant as resource does not play a role in a negotiation. A behaviour-dependent strategy is essentially about imitating the opponents conceding strategy. As both agents have full access to the utility of offers, both agents can use it to map the outcome space of the negotiation. With this information, imitating the opponent's behavior gives little advantage besides protecting oneself from exploitative agents. This leaves only a time-dependent strategy. Which concedes utility as a function of the time left in the negotiation. AgentRenting's conceding function took inspiration from the agent HardHeaded [7], which won the ANAC 2011.

At every round of the negotiation, AgentRenting uses formulas 1 & 2 to determine the utility to which to concede to at that round. Where, with formula 1: $MinUtility$ is initially set to the original Nash Point, $MaxUtility$ is the offer which gives AgentRenting the highest utility in a negotiation and $F(t)$ is the concession function as a function of time t

And with formula 2: t is the current round in the negotiation as a proportion of the total negotiation time, T is the total negotiation time and e is set to 0.02. This value also comes from Hardheaded [7] and means AgentRenting adopts a Boulware tactic. That is to say: AgentRenting starts conceding relatively late, but when it does, relatively fast. In a negotiation with less than 100 total steps, e is set to 0.05, as the few amount of steps in which the agent actually concedes makes it more favorable to start conceding earlier. This Boulware tactic ensures that AgentRenting does not concede utility for the vast majority of a negotiation and quickly concedes in the last stages of the negotiation to the Nash Point. At each round, AgentRenting picks an offer on or above the utility set by Formulas 1 and 2 which gives the highest utility

to the opponent. This is to ensure that all made offers are close to the Pareto Front. Formulas 1 and 2 are represented in AgentRenting as `concession_function`.

During the negotiation, the neural network predicts the opponent's reservation value based on metrics mentioned in section 2.1.2. This is done with function `self.update_partner_reserved_value`.
As running the NN is quite time-consuming, this function is only called in the last 5%. The NN returns a range in which the opponent's reservation value will be with a 0.8 probability. This can be interpreted as the precision with which the opponent's reservation value is predicted. If this range is smaller than 0.2, AgentRenting procedes to flatline. Flatlining is staying at the current utility and not conceding more, resulting in an identical offer being made until the last round in the negotiation.

In the last round of the negotiation, AgentRenting predicts the opponent's reservation value to be the upper bound of the range received from the NN. The upper bound is chosen to play it safe and not end up predicting lower than the opponent's actual reservation value. This predicted opponent's reservation value and AgentRenting's own reservation value are used to redefine the outcome space. From this reduced outcome space, it selects the offer with the most utility for itself. If the opponent's reservation value is predicted by the NN to be above the Nash point, their reservation value is predicted to be at the Nash point. If the prediction is correct and the opponent accepts any offer above its reservation value in the last round, this offer is accepted by the opponent and gives AgentRenting the highest advantage achievable.

If the NN cannot reach the certainty threshold during the negotiation, a backup tactic is employed to ensure a good outcome. After 90% of the negotiation time has passed, AgentRenting starts using the opponent's predicted concession curve, which is modelled as described in section 2.1.1, to update $MinUtility$ from formula 1, the utility that it is conceding to. The opponent's final offer which corresponds to the opponent's $MinUtility$ obtained by the regression is compared to the $MinUtility$ AgentRenting is conceding to itself. If its own $MinUtility$ gives AgentRenting a lower utility than the utility of the offer that its opponent is conceding to, AgentRenting changes its $MinUtility$ to be slightly below this level. This is represented in AgentRenting as `opponent_modeling`. The reason behind this strategy is to prevent AgentRenting from conceding to a point which is less favorable than what we expect our opponent to be willing to give us in the end. The choice to update $MinUtility$ to slightly below what we expect to get is to play it a little safer and not depend blindly on the regression technique.

If the opponent modeling fails and we do not flatline or update our $MinUtility$, the hard headed Boulware tactic is continued until the end, offering the Nash point as our last offer if there is no agreement before the final step.

The entire final offer strategy of AgentRenting is represented in the function `last_offer_strategy`.

2.2.2 Not final offer

AgentRenting's bidding strategy, in the case where it does not have the final offer, is similar to having the last offer but different in some key ways. Suppose we assume our opponent to do offers with descending utilities for them and ascending utilities for us. If this were the case, accepting their final offer would grant us the highest utility. Our only goal should be to convince any opponent modeling, if the opponent uses one, that we have a high reservation value, to convince the opponent to concede to us as much as possible.

A concession curve is used in the same way with the same form of function (formulas 1 and 2), with $MinUtility$ initially set to the original Nash point. However, the tactic variable e is set to 0.05 to be more conceding. To make the opponent concede, we reason it is necessary to concede ourselves as well, as their opponent modeling might fail otherwise, or they might have systems in place to deal with hard headed agents. The opponent's modelled concession curve from section 2.1.1 is used in the same way as in the final offer strategy, updating $MinUtility$ when we expect the opponent to concede to a better offer than we are currently conceding to, to prevent conceding more than necessary. However, the neural network is used in a slightly different way. If the opponent's reservation value range (i.e. precision) which we receive from the NN is below 0.2, and if the current final offer which we concede to seems to give the opponent a lower utility than their reservation value (now predicted to be the upper bound of this range), $MinUtility$ is updated to correspond with an offer which is best for us but still above the opponent's reservation value. Note that this overrides any $MinUtility$ updates which result from the opponent modeling via curve fitting. If we

do indeed update *MinUtility*, this still means our offers will not be rational for the opponent to accept until our final offer. But by making a final offer which we expect the opponent to accept, we try not to be completely dependent on the opponent's final offer. This strategy is represented in AgentRenting as `not_last_offer_strategy`

2.3 Acceptance strategy

During the negotiation, AgentRenting only accepts offers that are at least as good as the utility of the offer AgentRenting makes itself in the same round. If this is the case, AgentRenting accepts the offer made by our opponent. In the final step, if the opponent has the last offer, AgentRenting accepts any offer above its reservation value. This means that AgentRenting completely concedes to the opponent in the final round, if the opponent has the final offer, no agreement has been made before the final round, and the opponent's final offer is above our reservation value. Although this is a big difference with agent HardHeaded [7], which would often not come to an agreement, we believe that getting some utility is better than nothing at all, especially because AgentRenting compensates by being very hard headed when it does have the final offer. This is represented in AgentRenting as `acceptance_strategy`.

References

[1] Tim Baarslag, Koen Hindriks, Catholijn Jonker, Sarit Kraus, and Raz Lin. *The First Automated Negotiating Agents Competition (ANAC 2010)*, volume 383, pages 113–135. 01 2012. ISBN 978-3-642-24696-8. doi: 10.1007/978-3-642-24696-8_7.

[2] Tim Baarslag, Katsuhide Fujita, Enrico Gerding, Koen Hindriks, Takayuki Ito, Nicholas Jennings, Catholijn Jonker, Sarit Kraus, Raz Lin, Valentin Robu, and Colin Williams. Evaluating practical negotiating agents: Results and analysis of the 2011 international competition. *Artificial Intelligence*, 198: 73–103, 05 2013. doi: 10.1016/j.artint.2012.09.004.

[3] Tim Baarslag, Koen Hindriks, Mark Hendrikx, Alex Dirkzwager, and Catholijn Jonker. *Decoupling Negotiating Agents to Explore the Space of Negotiation Strategies*, volume 535, pages 61–83. 01 2014. ISBN 9784431547570. doi: 10.1007/978-4-431-54758-7_4.

[4] Enrico H. Gerding Nicholas R. Jennings Colin R. Williams, Valentin Robu. *IAMhaggler: A Negotiation Agent forComplex Environments*, volume 383, pages 151–158. 11 2011. ISBN 978-3-642-24696-8. doi: 10.1007/978-3-642-24696-8_10.

[5] Peyman Faratin, Carles Sierra, and Nick R. Jennings. Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems*, 24(3):159–182, 1998. ISSN 0921-8890. doi: https://doi.org/10.1016/S0921-8890(98)00029-3. Multi-Agent Rationality.

[6] S. Fatima, Michael Wooldridge, and Nicholas Jennings. Optimal negotiation strategies for agents with incomplete information. volume 2333, pages 377–392, 08 2001. ISBN 978-3-540-43858-8. doi: 10.1007/3-540-45448-9_28.

[7] Thijs Krimpen, Daphne Looije, and Siamak Hajizadeh. *HardHeaded*, volume 435, pages 223–227. 01 2013. ISBN 978-3-642-30736-2. doi: 10.1007/978-3-642-30737-9_17.