

XGAgent: A Gradient Boosting-Based Deceptive Negotiator for ANL 2026

Fukutoku Yuma,
Tokyo University of Agriculture and Technology,
Fujita Katsuhide Lab, fukutoku@katfuji.lab.tuat.ac.jp

June 19, 2026

Abstract

In the Automated Negotiation League (ANL) 2026, an agent’s success depends not only on finding an optimal agreement but also on managing information asymmetry. We present XGAgent, a negotiator that utilizes an XGBoost regression model to online-learn the opponent’s utility function. By treating negotiation offers as training samples, the agent constructs a predictive model of the opponent’s preferences. Our approach integrates machine learning with classical game-theoretic solutions, such as the Nash Bargaining Solution, to transition from exploration to exploitation as the negotiation progresses.

1 Introduction

In a multi-issue negotiation environment, an agent must navigate a high-dimensional outcome space while operating under uncertainty regarding the opponent’s preferences.

In this paper, we introduce XGAgent. The core philosophy of XGAgent is to treat the negotiation as a supervised learning task. We utilize Gradient Boosted Decision Trees XGBoost to approximate the opponent’s utility function u_{opp} based on the sequence of offers received.

2 The Design of XGAgent

The architecture of XGAgent consists of three main components: a predictive utility engine, an online learning module, and a strategic decision-maker.

2.1 Feature Engineering and Representation

To apply regression techniques, the outcome space $\mathcal{O}$ must be vectorized. For each outcome $o \in \mathcal{O}$, we construct a feature vector $\mathbf{x}_o$ comprising:

1. One-hot encoded representations of all negotiable issues.
2. The agent’s own utility value $u_{\text{self}}(o)$.

This allows the XGBoost regressor to learn a mapping $f : (\text{issues}, u_{\text{self}}) \rightarrow \mathbb{R}$, where the output is the predicted opponent utility $\hat{u}_{\text{opp}}$.

2.2 Online Learning via XGBoost

XGAgent updates its internal model during the negotiation. We treat each incoming offer from the opponent as a labeled sample. To handle the uncertainty of the opponent’s intent, we use a stochastic label assignment:

- If an offer is accepted by the opponent, we assign a high target utility $y \in [0.7, 1.0]$.
- If an offer is rejected (or is our own rejected offer), we assign a lower target utility $y \in [0.2, 0.5]$.

To prioritize recent information and critical offers, we implement importance sampling by assigning a higher weight ($\omega = 10.0$) to the training samples corresponding to the most recent accepted offers.

3 Concealing Bidding Strategy

The primary propose strategy of XGAgent is three types:

1. **Lure Outcomes ($\mathcal{D}_1$):** Outcomes where $u_{\text{self}}(o) > 0.7$ and $\hat{u}_{\text{opp}}(o) > 0.6$. These are relatively fair offers designed to maintain engagement.
2. **Masking Outcomes ($\mathcal{D}_2$):** Outcomes where $u_{\text{self}}(o) > 0.7$ but $\hat{u}_{\text{opp}}(o) < 0.5$. These are deceptive offers that unfavorable to the opponent, intended to obscure our true high-utility targets.
3. **Exploration Outcomes ($\mathcal{O}_{\text{self}}$):** sampled Relatively high outcomes based on the current model of the self.

The agent selects between these strategies using a probability distribution regulated by the `relative_time`. As time $\rightarrow 1$, the agent transitions from exploration to a structured convergence based on the Nash Bargaining Solution (NBS).

4 Acceptance Strategy

To prevent exploitation, XGAgent employs a two-tier acceptance logic:

$$\text{Accept if: } \begin{cases} u_{\text{self}}(o) > 0.8 \cdot \max(u_{\text{self}}) - > u_{\text{self}}(o) > 0.7 - > u_{\text{self}}(o) > 0.5 \end{cases} \quad (1)$$

This allows the agent to capture high-value deals immediately while ensuring it does not walk away from profitable deals during the final stages of the time limit.

5 Opponent Model

The opponent model is implemented via the `PredictiveUfun` class, which wraps the XGBoost regressor into a functional interface compatible with the NegMAS framework. This allows us to calculate the Nash Product:

$$\text{Maximize: } (u_{\text{self}}(o) - r_{\text{self}}) \cdot (\hat{u}_{\text{opp}}(o) - r_{\text{opp}}) \quad (2)$$

where r denotes the reservation values. By using $\hat{u}_{\text{opp}}$ derived from the regressor, XGAgent can propose outcomes that are theoretically optimal under its current belief of the opponent’s preferences.

6 Evaluation

In evaluation, we were able to achieve numerically almost identical results for the basic agents.

7 Lessons and Suggestions

The main challenge with this agent, which is still under development, is the low accuracy of the training data. Another problem is the redundancy of the input space, which I believe can be solved by focusing on aspects such as the value and weight of the choices.

Conclusions

XGAgent can effectively manage information asymmetry in automated negotiation by combining supervised learning and strategic deception. It models the opponent as a regression problem to reach a mathematically valid Nash bargaining solution while maintaining its utility.