Multi-issue Bargaining with Deep Reinforcement Learning

Ho-Chun Herbert Chang

February 19, 2020

Abstract

Negotiation is a process where agents aim to work through disputes and maximize their surplus. As the use of deep reinforcement learning in bargaining games is unexplored, this paper evaluates its ability to **exploit**, **adapt**, and **cooperate** to produce fair outcomes, in comparison to classical game theoretic results.

Two actor-critic networks were trained for the bidding and acceptance strategy, against time-based agents, behavior-based agents, and through self-play. Gameplay against these agents reveals three key findings. 1) Neural agents learn to **exploit** time-based agents, achieving clear transitions in decision preference values. The Cauchy distribution emerges as suitable for sampling offers, due to its peaky center and heavy tails. The kurtosis and variance sensitivity of the probability distributions used for continuous control produce trade-offs in exploration and exploitation. 2) Neural agents demonstrate **adaptive behavior** against different combinations of concession, discount factors, and behavior-based strategies. 3) Most importantly, neural agents learn to **cooperate** with other behavior-based agents, in certain cases utilizing non-credible threats to force fairer results. This bears similarities with reputation-based strategies in the evolutionary dynamics, and departs from equilibria in classical game theory.

Contents

Chapter 1

Introduction

Negotiation is the process where parties interact to settle issues, discover surplus, and create contracts. Because negotiation is so essential to society, it has been widely studied by different fields, in economics [54, 58], artificial intelligence [34, 35, 26], business [82, 38, 20, 30], communication [5, 43], and behavioral psychology [64, 17].

However, negotiation is typically very costly. Automated negotiation is a field that, for the past 20 years, has promised to reduce the costs of human negotiation, avoid social confrontation, and augmented the abilities of human negotiators. It has found great success in e-commerce and supply chain management [21]. Recent success in deep reinforcement learning (DRL) in games, such as Chess [71], Go [72], Poker [10] and Atari games [48] have inspired DRL's application to complex human tasks, including negotiation [39]. However, the direct application of DRL on negotiation, formally known as bargaining games within game theory, is unexplored. The central goal of this dissertation is to understand the behavioral dynamics of negotiation bots trained with DRL, and assess its ability to **exploit**, **adapt**, and **cooperate**.

Beyond reducing costs, understanding the behavior of DRL agents is paramount because, with the deployment of any technology, there comes potential harm. When DRL was applied to natural language generation for negotiation, agents were shown to misrepresent their intentions within negotiation dialogues [39]. As natural language generation techniques mature, it is easy to imagine a future where complex transactions, contracts, and negotiations are facilitated through bots. Already, more than 80% of WeChat transactions are mediated through chatbots and conversational marketing— the use of customized chatbots in marketing— has been forecasted as the most dominant interface between consumers and businesses [19]. As it is well-established that suppliers hold more information that buyers [3], DRL bots may exploit asymmetries when mediating markets, exposing consumers to representational harm.

To be explicit about our problem, Baarslag divides negotiation into three pillars— the *bidding strategy*, *opponent modeling*, and *acceptance strategy*. This dissertation evaluates the extent in which actor-critic models can learn end-to-end bidding and acceptance strategies through gameplay, solely through the opponent's behaviors alone. As mentioned prior, the three behavioral traits of interest are **exploitation** (ability to utilize an opponent's weaknesses), **adaptation** (ability to play against different opponents), and **cooperation** (ability to work with other agents for fairer and more efficient results). Related sub-problems include 1) when sub-optimal bidding results occur, 2) what the variables that produce these limitations are, and 3) how neural agents learn fair strategies against more complicated agents. The emergence of fairness is often studied through the lens of game theory [81, 80]; we adopt a similar approach for analysing our agent's behavior.

This project aims predominately to contribute to **automated negotiation**. We contribute by implementing DRL within the bargaining domain, through competition against time-based agents, tit-for-tat agents, and self-play. The first half of the results focuses on precision— the agent's mechanics and exploitative abilities, whereas the second half on behavioral dynamics. There are three key results for behavior. First, we demonstrate exploitative and adaptive behavior against time-based agents and the simple behavioral agent, showing clear switching behavior in both acceptance and bid strategy, up to 80% and 90% (Pareto) efficiency respectively. Second, the neural network cooperates with more complex behavior-based agents to produce fairer results. Furthermore, in a game where agents are allowed to accrue interest, it learns to cooperate during self-play. Third, and most interestingly, results demonstrate the emergence of fairness through the adoption of non-credible threats This is considered irrational in classic game theory. We draw a comparison to results in evolutionary game theory, where a similar divergence from assumptions of rationality.

Additionally, we contribute an analysis that features contemporary reinforcement learning research directions [4]— negotiation with deadlines is synonymous with the cliff-walking problem and construing offers aligns with control over continuous state space. Using derivative-based analysis based on marginal utility for optimal stopping, I demonstrate a negative relationship between time error and the second derivative. Secondly, successful continuous control on this domain requires careful control over exploration and exploitation— the Cauchy distribution arises as a candidate due to its peaky center, heavy tails, and sensitivity to change in variance.

Chapter 2

Background and Terminology

Due to the expensive nature of negotiation, attention to automating the process has gained considerable traction in the past twenty years [8], since the development of Contract Net Protocol by Smith in the 1980s [75]. It spurned the promise of finding better outcomes than human negotiators [9, 18, 32, 55, 79, 41].

Baarslag [6] identifies three key aspects of negotiation strategy— *bidding strategy*, *opponent modeling*, *acceptance strategy*. The bidding strategy asks what concessions to make in the process of counter-offering. The acceptance strategy asks when and which offers should be accepted. Opponent modeling focuses on understanding what the opponent wants, in order to make better decisions about bidding and acceptance.

Negotiation is simultaneously collaborative and competitive. A good negotiation outcome is characterized by 1) win-win situations 2) avoiding no agreement, 3) and avoiding exploitation [6]. While the first two activities are collaborative, the third is competitive. Negotiators are typically unwilling to share information in fear of being exploited [51, 14, 58]. Thus, one main challenge of negotiation is overcoming the information barrier, which makes bargaining a game of incomplete information.

Therefore, opponent modeling can be regarded as the most important in the field of negotiation . Like poker, successful negotiation arises from understanding your opponent and generating profits off their behavioral heuristics and weaknesses. Especially given the diversity of negotiation agents, it is difficult to produce a singular agent that plays well against multiple opponent typologies. Therefore, a focus of the field is less on *optimal play*, but *exploitative play* using adaptive agents [6].

Before reviewing the state-of-the-art, we layout the necessary terminology for negotiation, beginning with the mechanism, measures of fairness and optimality, then discuss the common negotiation strategies used to benchmark negotiation models.

2.1 Overview of Negotiation

A *negotiation setting* contains a *protocol*, *agents*, and *scenario*. The protocol determines the rules of how agents interact with each other. The *scenario* takes place in a *negotiation domain* which determines an *outcome space*, denoted as Ω. A negotiation domain can have a single or multiple issues. *Issues* refer to the resources under contention, such as the price of an object or level of service. Thus, an outcome $\omega \in \Omega$ can be described as a specific division of the issues. Agents have *preference profiles*, which determines specific outcomes they prefer.

2.1.1 Protocols

We use single-issue bargaining as a preliminary illustration. Given a unit pie, two players A and B are asked to split it amongst themselves [24]. Suppose Agents A and B negotiate N rounds to divide a unit pie, by alternately proposing outcomes called *bids* or *offers*, until a player accepts. We denote an offer $x = (x, y)$, such that $x + y = 1$.

This process of alternating offers is known as the Rubenstein's Bargaining Protocol. Games with one round, are known as an *ultimatum game* [65]. In ultimatum games, A makes the first and only proposal. B can only accept or reject it, which means A has all the power. Similarly, if there are two rounds, then Player B has the advantage. In a game of repeated offers, it is necessary to introduce some form of discounting factor— otherwise, players would negotiate forever. The discount factor δ makes a portion of the pie go bad at every round. Thus, it is in the best interest for players to finish the game as soon as possible.

The Rubenstein Bargaining Protocol is widely used because it accurately simulates many real-world scenarios [65]. Multi-issue bargaining is more complex, as multiple issues are under contention and requires further protocol restrictions describe how each issue is resolved. Common ones are [35]:

1. **Package-deal Procedure:** All issues addressed at once.
2. **Simultaneous Procedure:** All issues are solved independently. It is equivalent to m single-issue problems.
3. **Sequential Procedure:** Negotiates one issue at a time, with a predetermined sequence. Cannot negotiate prior or future issues.

An alternative protocol is the *monotonic concession protocol* [63], where agents disclose information about how they value each issue, and their subsequent offers must have less utility than their prior ones. Other protocol considerations include [23]:

1. **Time Constraints:** Beyond the discount factor δ, there is often a deadline T. If negotiation does not end by T, players earn 0 utility (known as the *conflict deal*).

2. **Divisibility:** Issues may be atomic and discrete, or divisible and continuous.

3. **Lateral-ness:** Whether negotiation is between two parties (bilateral) or with multiple parties (multilateral).

4. **Reserve Price** r**:** The minimum an agent is willing to accept.

2.1.2 The Scenario

The utility is defined as the *cumulative utility*,: a combination of sub-utility functions. Most commonly used is the linear additivity. With $\vec{x}$ the division for Player A (PA) and $\vec{y}$ for Player B (PB), the aggregate utility is of PA is:

$$U_1(x,t) = \begin{cases} \delta^{t-1}\vec{W}^T\vec{x} = \sum_{i=1}^{m} w_{i,a}x_i\delta^{t-1} & \text{if } t \leq T \\ 0 & \text{otherwise (Conflict Deal)} \end{cases} \tag{2.1}$$

where $w_{i,a}$ is the value (weight) PA ascribes to issue i, δ the discount rate, and x_i the division for issue i. This can be viewed as the discounted dot product of weights $\vec{W}$ and issue division x. In many cases, however, utilities are not linear in combination— for instance, in the auctions of multiple items, combinations of items yield greater rewards, to the effect of the sum being greater than the parts, due to synergistic effects. These are modeled with *non-linear utility functions* [31].

The action space is defined by three possible actions: $A_i = \{\text{Offer(x,y)}, \text{reject}, \text{accept}\}$. Offers are made after rejections, and should an agent choose to accept an offer the negotiation ends. Each issue is often normalized such that $x_i + y_i = 1$. For games with only one issue, the offer consists of the division of one pie. For multiple issues, offers are represented as vectors, subject to $\vec{x} + \vec{y} = 1$. For this dissertation, the outcome space is assumed to continuous, linear, and normalized.

2.1.3 Outcome Spaces

Each player has a preference ordering, called the *preference profiles*, on all possible outcomes. An outcome ω' is weakly preferred to ω if $u(\omega') \geq u(\omega)$, which is denoted $\omega' \geq \omega$. Similarly, ω' is strictly preferred to ω (denoted $\omega' > \omega$) if $u(\omega') > u(\omega)$. For linear additive utilities, the preference profile can be inferred directly by the weights.

Now we present metrics used to evaluate our three criterion. An outcome is called *Pareto Optimal* if there exists no outcome ω' that a player would prefer without worsening their opponent's outcome. Formally:

$$(\omega' >_A \omega \wedge \omega' \geq_B \omega) \ \vee \ (\omega' >_B \omega \wedge \omega' \geq_A \omega)$$

The *Pareto Frontier* describes all Pareto optimal solutions, which we denote as Ω_P. When an offer is not Pareto Optimal, then through negotiation there is potential to reach an outcome without players conceding anything.

There are two other useful metrics. Let $\omega_P \in \Omega_P$ denote the set of outcomes that are Pareto optimal. The *bid distribution* denotes the mean distance to the Pareto frontier, shown in Eq. 2.2. A high bid distribution indicates bids are on average far away.

$$BD(\Omega) = \sum_{\omega \in \Omega} \frac{\min d(\omega, \omega_P)}{|\Omega|} \tag{2.2}$$

Usually, simultaneous maximization of outcomes is not possible, as there is a region of disagreement between players. Another useful metric is the product of utilities ($U_A \cdot U_B$), known as the *Nash Product*. A fair outcome is often characterized using the *Nash solution*, the outcome that maximizes the product of utilities, shown in Eq. 2.3.

$$\omega_{Nash} = \max_{\omega \in \Omega} U_A(\omega) \cdot U_B(\omega) \tag{2.3}$$

2.2 Strategies

In cases of perfect information, it is possible to determine the optimal bidding strategy [23]. However, as previously mentioned, perfect information is unlikely in bargaining as agents are unwilling to give away their preferences in fear of exploitation. This motivates the development of negotiation tactics under imperfect information. These negotiation tactics can broadly be classified as *time-dependent* or *behavior-dependent* tactics, based on a *decision-function* that maps state to a target utility.

2.2.1 Baseline Strategies

Two are commonly used. The *Hardliner* always bids maximum utility for itself, which emulates the "take-it-or-leave-it" attitude. The *Random walker* denotes agents that bid randomly, serving as a standard baseline.

2.2.2 Time-dependent Strategies

Time-dependent Strategies denote functions that produce offers solely based on time. At every round, the agent calculates their *decision utility* which determines whether they accept an offer or not. For time-dependent agents, this is:

$$u(t) = P_{min} + (P_{max} - P_{min})\dot{(}1 - F(t)) \tag{2.4}$$

P_{max} and $P_{min} \in [0, 1]$, thus parametrizing the range of the offers. Frequently, $F(t)$ is parametrized as an exponential function:

$$F(t) = k + (1 - k) \cdot (\frac{t}{T})^{1/c} \tag{2.5}$$

where c is the concession factor. k is often set to 0 for simplicity. Fig. 2.1 shows the decision utilities of different agents. If $0 < c < 1$, then the agent concedes towards the end and is known as *Boulware*. Otherwise, if $c \geq 1$, the agent concedes quickly and offers its reservation value, thus it is known as a *Conceder*. $c = 1$ means the agent's decision utility decreases linearly.

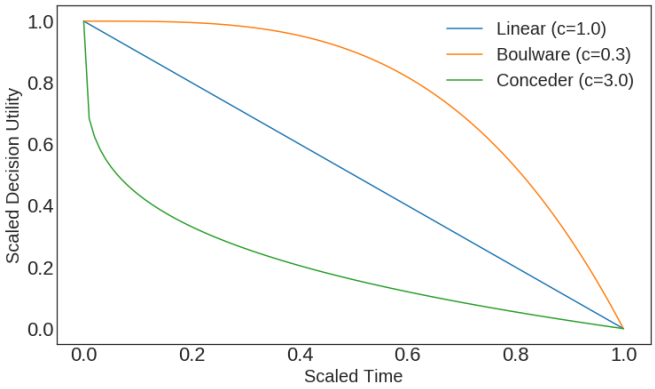

Figure 2.1: Decision utilities of time-based agents with different concession factors.

2.2.3 Behavior-based Strategies

Behavior-dependent and *imitative* bidding strategies observe the behavior of the opponent to make their own decisions on what to offer and what to accept. The most well-known is *tit-for-tat*, which produces cooperation through reciprocity. It's three central mantras are 1) never defect first (play nice as long as the opponent plays nice), 2) retaliate if provoked and 3) can forgive after retaliation.

In negotiation, the *relative tit-for-tat* (TFT) strategy reciprocates by offering concessions proportional to their opponent's concessions from δ rounds

prior:

$$x_{a\to b}^{t_{n+1}}[j] = \min\left(\max\left(\frac{x_{b\to a}^{t_{n-2\delta}}[j]}{x_{b\to a}^{t_{n-2\delta+2}}[j]} x_{a\to b}^{t_{n-1}}[j], \min_j^a\right), \max_j^a\right) \tag{2.6}$$

Here, $x_{a\to b}^{t_{n+1}}[j]$ is the offer for issue j. This value is determined by the ratio of the opponent's prior concessions, which then scales the agent's own prior offer $x_{a\to b}^{t_{n-1}}[j]$. The min and max values ensure offer values are within range.

2.3 State-of-the-Art in Negotiation

Machine learning methods in the domain of negotiation can be broadly separated into the following types: Bayesian learning, non-linear regression, kernel density estimation, and artificial neural networks. These methods have been applied to mostly model an opponent's (acceptance and bidding) strategy, then derive an analytic response. This is because if an agent knows the opponent's bidding strategy, then the agent can compute its optimal strategy [6].

For estimating the opponent acceptance strategy, techniques can be siloed into the estimation of individual variables. Zeng and Sycara provide a popular and intuitive Bayesian approach for estimating the reserve price, using historical data. The model generates a set of hypotheses on the opponent's reserve price, then attaches a likelihood using the history. The estimate is a weighted sum of the hypotheses based on their likelihoods [87]. This technique has been adapted to estimate the deadline for time-dependent tactics [74]. In general, acceptance strategy estimation uses some form of Bayesian learning [78, 86, 73, 27, 62], augmented with non-linear regression [2, 86, 74, 28, 29], kernel density estimates [22, 55, 14], polynomial interpolation [66], genetic algorithms [46, 33], and more recently neural networks [21].

In contrast, neural methods have been applied much more aggressively to the bidding strategy [6]. In simpler cases where the general bidding formula is known, regression is sufficient as the problem reduces down to parameter estimation. If no formula is known, then neural networks are employed to approximate the opponent's bid strategy, typically using a large database of bid history. Oprea [53] uses a time-series approach on single-issue negotiations, taking in only the opponent's current bid. By 2008, early efforts for opponent move prediction using neural networks [11], who focused on predicting human bidding strategies. This was particularly relevant in e-commerce and supply chain management, as forecasting bids is useful in determining automated strategies [37, 12, 49]. When the domain is general, researchers have found success using deep learning with multilayer perceptrons. Masvoula shows reliable predictions using single deep networks both with and without historical knowledge [45, 44]. Papaioannou and Rau et al.

have shown the concession factor and weight of each issue can be predicted if the opponent is known to be time-dependent, using multilayer neural nets [57, 61] or single layer, radial basis function neural nets [56].

Reinforcement learning approaches to negotiation began as early as the late 20th century, often denoted as adaptive learning [60]. Today DRL has more frequently been used with natural language processing [25, 15]. Lewis et al. implement an end-to-end DRL negotiation dialogue generator [39]. They curated a set of human-human dialogues with Mechanical Turk, then trained on four gated recurrent units [13], a type of long-short term memory neural net [50]. However, this study focuses on emulating human language, with less concern on optimality— for instance, their DRL agent present 58.6% and 69.1% Pareto Optimality against simple autonomous agents and humans respectively, on a very limited, discrete action space (around 200 offers).

As illustrated with this brief survey, there is an immense number of agent designs for negotiation. The primary weakness in best performing models, such as Bayesian models in acceptance or bid prediction, is they require specific domain assumptions and architectures. Another weakness is for these negotiators to perform well in populations of different strategies, an additional opponent classifier is needed, which introduces further uncertainty. Additionally, opponents can use more complex behavioral strategies and mixed strategies— pure strategies associated with a probability—that requires higher levels of adaptability to play against.

All of this motivates an adaptive agent with a fixed architecture that can perform well against different opponents. An end-to-end negotiation agent is desirable as the only required input is the offer, time step, and public knowledge, and can adapt online during gameplay. Although deep learning often comes at the expense of explainability, a fixed architecture playing end-to-end means we do not need additional classifiers and assumptions about the opponent. The success of AlphaZero in chess is largely because it did not rely on hand-crafted heuristics and assumptions like other engines [72]; likewise, Libratus learned to exploit specific human opponent idiosyncrasies in poker [10]. An end-to-end, adaptive neural agent is the analogous solution for negotiation. It is a convenient coincidence that the negotiation domain also aligns with the current interest in continuous control positions deep reinforcement learning.

Chapter 3

Methods

Each bilateral negotiation scenario takes these as an input: *two utility weights* w_1 and w_2, discount factors δ, a deadline of 20, reserve price of 0, and two agents. These two agents then negotiate to an agreement or disagreement (conflict deal). For simplicity, our agent's utility weights are $(1, 2, 3)$ whereas the opponent's is $(3, 2, 1)$. Experiments consist of the neural agent playing against a second agent until reaching stopping criteria.

3.1 Deep Reinforcement Learning

Multi-agent Reinforcement Learning is formally the study of n-agent stochastic games [70], described as a tuple $(N, S, \vec{A}, \vec{R}, T)$. N is the number of agents. S is the set of states and $\vec{A} = A_1, \ldots, A_n$, with each A_i the set of actions agent i can take. In the most basic case, by treating the environment as static, the **single-agent Q-learning** algorithm developed by [83] gives the optimal policy in an MDP with unknown reward and transition.

$$Q(s,a) += \alpha[R(s,a) + \gamma V(s')] \qquad V(s) \leftarrow \max_{a \in A} Q(s,a) \tag{3.1}$$

$Q(s, a)$ estimates the value of taking action a when on state s, and $V(s)$ the value of the state by taking the best action. Extension of this paradigm to multiple agents is difficult. One approach is to assume the environment as passive, each agent with their own reward and transition functions. However, this falsely assumes agent actions do not influence each other [69]. Another approach is to define the $Q-$value function over all agents actions, but introduces a dynamic programming challenge in updating V.

In recent years, reinforcement learning has been applied successfully in conjunction with deep learning, using deep neural networks to approximate value functions. A breakthrough comes from *policy-gradient methods*. Traditionally, RL algorithms are *action-value methods*: after learning values of the action, algorithms select actions based on the estimated action values.

In contrast, policy-gradient methods learn a parametric policy without consulting the value function [76]. By policy we mean an agents strategy— what it does at a given state and time.

Additionally, in cases where the environment is dynamic, it may be optimal to acquire a stochastic policy— a probability distribution over possible actions. This distribution is updated to associate actions with higher expected rewards with higher probability values. Since probabilities can be over discrete or continuous action spaces, DRL is a useful control framework for negotiation, as the decision to accept or reject an offer is discrete, whereas bidding is on continuous space (on $[0, 1]^n$ given n issues).

3.1.1 Policy Gradients

Call the policy π and let parameters θ define a probability distribution. The probability of action a is denoted as $\pi(a|s, \theta) = \mathbf{P}\{A_t = a|S_t = s, \theta_t = \theta\}$, that is, the probability of taking action a at time t given that the state s and parameters θ. Similarly, a learned value function, such as using a neural network to approximate the value, can be represented as $\hat{v}(s, w)$, where $w \in \mathbf{R}^d$ is its weights.

As with action-value RL, policy parameters are optimized to maximize a scalar performance measure $J(\theta)$:

$$J(\theta) = \mathbf{E}[\sum_{t=0}^{T-1} r_{t+1}] \tag{3.2}$$

which describes the expected future aggregate rewards (sum of rewards from $t = 0$ until the end). The policy values are updated according to J through gradient ascent:

$$\theta_{t+1} = \theta_t + \alpha\widehat{J(\theta_t)}$$

For discrete actions, actions are selected by estimating a *numerical preference value* or *logit* $h(s, a, \theta)$, based on the state, action, and parameter values (weights in a neural net). Actions are then selected using the softmax distribution:

$$\pi(a|s, \theta) = \frac{e^{h(s,a,\theta)}}{\Sigma_b e^{h(s,a,\theta)}} \tag{3.3}$$

For instance, for the acceptance strategy, an agent can reject or stop. Associate with these actions h_R and h_A respectively, and a stochastic policy is defined.

However, updating the policy in respect to J requires the *policy-gradient theorem*, which provides guaranteed improvements when updating the policy parameters [76]. The theorem states that change in performance is proportional to the change in the policy, and a full statement is given in Appendix A.1. The theorem yields a canonical policy-gradient algorithm— REINFORCE [76, 84, 77]. The parameter updates is:

$$\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(a_t|s_t, \theta_t)}{\pi(a_t|s_t, \theta_t)} \tag{3.4}$$

where G_t is the observed reward. Intuitively, the update is the reward multiplied by the gradient of the action probability divided by the action probability. If G_t is high, this increases the chances of visiting that state in the future. Note, the policy gradient is often expressed as $\nabla \ln \pi(a_t|s_t, \theta_t)$, which yields the fraction through the chain rule.

3.1.2 Deep Reinforcement Learning for continuous variables

Secondly, actor-critic models are useful because they separate the policy space and action space, which means policy selection can occur on a continuous domain. For instance, in a uni-variate control problem, the choice of action can be sampled from a normal distribution. The policy approximation with a normal distribution is:

$$\pi(a \mid s, \theta) = \frac{1}{\sigma(s,\theta)\sqrt{2\pi}} \exp\left(- \frac{(a - \mu(s,\theta))^2}{2\sigma(s,\theta)^2} \right)$$

During back-propogation, the θ values are updated such that μ and σ reflect a better reward using the Equation 3.4.

3.2 Actor-Critic Implementation

We have arrived at our main method. Unlike REINFORCE, which only learns a policy, actor-critic models simultaneously learn a value function approximation and a policy. Intuitively, the value function critiques whether an action undertaken by the policy is good, rather than being an absolute measure. Thus, we make a modification to Equation 3.4, substituting the reward G_t with the value estimate $\hat{q}(s_t, a_t, \vec{w}_t)$ in Equation 3.5:

$$\theta_{t+1} = \theta_t + \alpha \nabla \ln \pi(a_t|s_t, \theta_t) \hat{q}(s_t, a_t, \vec{w}_t) \tag{3.5}$$

The process of negotiation thus requires two actor-critic nets— one for the acceptance strategy and another for the offer strategy. The algorithmic procedure is shown in Fig. 3.1, with pseudo-code provided by Algorithm 3 in the Appendix. We use univariate and single-issue interchangeably, as with multivariate and multi-issue. Next, we describe the architectures of the acceptance and bidding strategy.

3.2.1 Acceptance Net Architecture

The first neural network approximates the **accecptance strategy**. For the univariate case, the input x is a two-element vector consisting of the

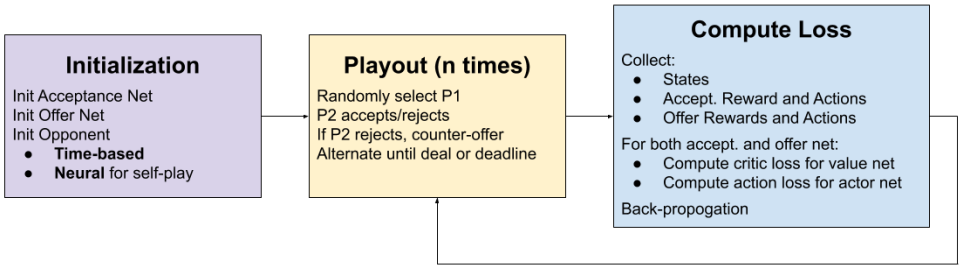

Figure 3.1: Flowchart of training algorithm.

opponent's offer and the current time step. For the multivariate case, the input is four-dimensional.

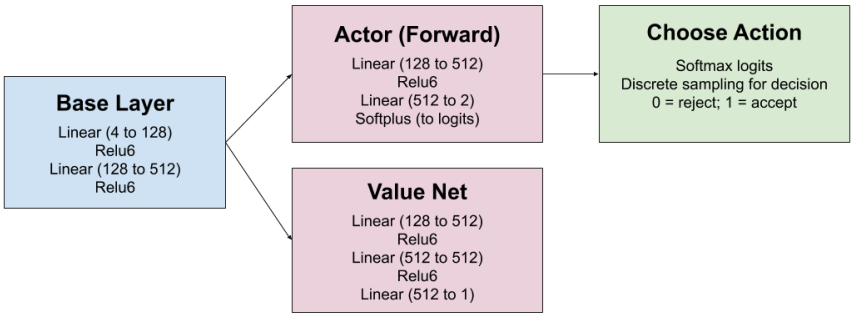

Figure 3.2: Architecture of the Acceptance Strategy.

At every time step, Accept Net takes in the opponents offer, encodes it to a 512 hidden state using two affine-Relu6 pairs. This base layer is shared between the actor and value network, which facilitates a shared representation [47]. The actor takes in the embedded state and outputs two logit values, which are softmaxed to choose the appropriate action. Similarly, the value network outputs the expected reward estimate.

The Relu6 is a variant of the Relu functions ($\max\{0, x\}$), but capped at 6. Relu6 layers have been shown to train faster (due to the limit on byte representation) and to encourage the learning of sparse features earlier on [36]. This is important since gameplay is path-dependent and, against a mixed set of opponents, states may be sparse, which we confirmed during preliminary testing. Hyper-parameters were also chosen through testing, reducing layers and the number of hidden states until training behavior changed. The full architecture is shown in Fig. 3.2.

After playout, the critic loss is calculated by taking the mean-squared error (MSE) of the temporal difference— the difference between the observed rewards and the value network's forward pass. Learning parameters are

given in Table 3.1.

for *Every Reward, State and Action* **do**
  TDLoss = Reward $- \hat{q}(s_t, a_t, \vec{w}_t)$;
  C_{loss} = (TD Loss ;
  LogProbs = $\ln P(a|\pi)$;
  $A_{loss} = -$ LogProbs $\cdot$ TDLoss;
  Backprop on C_{loss} and A_{loss};
end

Algorithm 1: Acceptance Net Actor-Critic Update

3.2.2 Offer Net Architecture

Next, assuming the agent has rejected the offer, the agent now takes in the same input and decides a counter-offer. Since we have three issues and offers operate on continuous space, the Offer Net must output a vector $o \in [0, 1]^3$. To do so, we implement DRL with continuous control, sampling from three different three types of distributions: 1) multivariate Gaussian, 2) three beta distribution and 3) three Cauchy distributions.

The *multivariate Gaussian* is parametrized by a vector of means $\vec{\mu}$ and covariance matrix Σ. However, a common assumption in deep learning is that the neural network will capture interdependencies between variables. Hence, an estimate of individual standard deviations along each dimension will suffice. The probability density and policies are given explicitly below.

$$\pi(a \mid s, \theta) \sim \vec{\mu}(s, \theta), \Sigma(s, \theta) \qquad f_X(x_1, x_2, x_3) = \frac{\exp\big(-\frac{1}{2}(\vec{X} - \vec{\mu})^T \Sigma^{-1} (\vec{X} - \vec{\mu})\big)}{\sqrt{(\pi}^3 |\Sigma|} \tag{3.6}$$

The *beta distribution* is defined on the $[0, 1]$ interval, and defined by two positive shape parameters α and β. This is useful as offers are held to a finite span. The PDF is:

$$\frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathrm{B}(\alpha, \beta)} \qquad \textit{with} \qquad \mathrm{B}(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha + \beta)} \tag{3.7}$$

where Γ denotes the gamma function (fractional factorial). Some useful properties of the beta distribution include its intuitive mean and relatively simple expression for variance:

$$\mathrm{E}[X] = \frac{\alpha}{\alpha + \beta} \qquad \mathrm{var}[X] = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)} \tag{3.8}$$

Lastly, the *Cauchy distribution* is parametrized similarly to the normal with μ and γ denoted more generally as location and scale. It's density function is given as :

$$\frac{1}{\pi\gamma(s,\theta)\left[1+\left(\frac{x-\mu(s,\theta)}{\gamma(s,\theta)}\right)^2\right]} \tag{3.9}$$

The neural architecture follows the same actor-critic model described in Section 3.2. A base layer inputs into the value network and six other neural blocks, estimating the distribution parameters— three means (locations) and sigmas (scale). For the beta distribution, these are estimates on α and β. These six variables are then used to sample the offer, which serves as the action used during loss calculation.

$$\text{Offer} = (o_1, o_2, o_3, t) \sim dist(\theta, s) \tag{3.10}$$

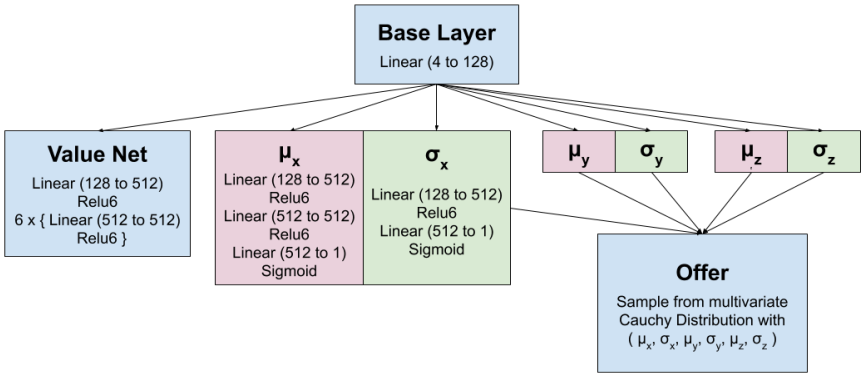

Figure 3.3: Architecture of the Offer Strategy for multivariate normals and Cauchy distributions. For the beta distribution, the estimated variables were α and β, and the final forward pass layer is a ReLu layer, instead of a Sigmoid.

The value network consists of seven layers of affine-Relu6 layers. Neural estimates of the mean were conducted with two affine-Relu6 layers, followed with an affine-sigmoid layer to constrain the output between 0 and 1. Sigma estimates used one affine-Relu6 layer and one affine-sigmoid layer. For the beta distribution, the network estimated three pairs of α and β. Since $\alpha, \beta > 0$, the final sigmoid layer was replaced with a Relu layer. Fig. 3.3 shows the architecture in full. Apart from a similar justification for the use of Relus for Accept Net, Relus have documented success for continuous control as well [40]. Hyper-parameter choice was chosen in a similar way.

Training was undertaken using Adam with learning parameters are given jointly in Table 3.1. The exact computations for back-propagation are given in Algorithm‘2. Note, because back-propagation occurs on a continuous domain, log-probabilities of continuous density functions can be positive

when variance is small.

for *Every Reward, State and Action* **do**
  TDLoss = Reward $- \hat{q}(s_t, a_t, \vec{w}_t)$;
  C_{loss} = (TDLoss)^2 ;
  LogProbsX = $\ln P(a_X|\pi)$;
  $AX_{loss} = -$ (LogProbs + Entropy) $\cdot$ TDLoss;
  Compute AY_{loss}, AZ_{loss};
  Backprop on C_{loss}, AX_{loss}, AY_{loss}, AZ_{loss}
end

Algorithm 2: Offer Net Actor-Critic Update

Here, the entropy is defined as $= 0.5 + \log(2\pi) \cdot \log(\sigma_Y)$. Adding entropy introduces noise to enable action exploration. Also, high variance means higher loss, so overtime, the variance decreases to improve the precision of the evolved strategy.

	Learning Rate	Epochs	Optimizer
Accept Net	$3e-5$ to $5e-5$	8000	Adam
Offer Net (Gaussian & Cauchy)	$1e-4$ to $5e-4$	4000	Adam
Offer Net (Beta)	$1e-3$	5000	Adam
Self-Play	$1e-4$	3000	Adam
Tit-for-tat	$1e-4$	5000	Adam

Table 3.1: Deep Learning Training Parameters. Early stopping criteria was convergence in play-out time for 500 epochs. This varied by the concession factor and discount rate

3.2.3 Reward Scheme

Accept Net and Offer Net share the same reward scheme. With deadline T, value weights w, and final offer x, the reward given to the neural agent is:

$$R_t = \begin{cases} w^T x & \text{if } t_f < T \\ -K & \text{if } t_f = T \text{ (conflict deal)} \end{cases}$$

This reward function encourages the agent to increase its offer x but not so much it forces a conflict deal and receives low reward. Unless specified, $-K = -1$.

3.2.4 Self-Play: Characterizing Behaviors with Game Theory

Within bargaining game theory, a focus has been how mechanisms induce norms of fairness, particularly from a branch of game theory called evolutionary game theory (EGT). EGT originates from biology, where it studies

the dominance of species through evolutionary pressure, and has been extended to behavioral economics to understand the evolution of behavioral traits. Nowak et al. showed fairness could be induced if reputation was taken into consideration for the repeated ultimatum game [52], where agents play many one-round negotiations with other agents.

Their most important finding was that, if agents learned to reject offers they deemed too low, a population of *fair agents would emerge*. Thus, reputation refers to the trait of intentionally reject low offers, and has been confirmed with computational and empirical results [59]. We implement a similar study to compare results, using the neural actor-critic model instead of evolutionary methods, and on multi-round negotiation rather than the ultimatum game. Details of implementation are given in Section 4.4.2, where we demonstrate a similar appearance of fairness.

3.2.5 Against Behavior-Based Agents

Lastly, we train our agent against two behavior-based agents. The first is the relative tit-for-tat described in Section 2.2.3, and the decision function in Eq. 2.6. Furthermore, we implement a Bayesian Tit-for-tat, by estimating the opponent's value weights. The Bayesian tit-for-tat agent first measures the opponent's concession using its own utility function. Then, it mirrors the amount of concession. Finally, this offer is made as attractive as possible using a Bayesian opponent model [7].

To do this, we first take the ratio of the opponent's offer at $t - \delta$ and $t - \delta - 1$ to update the decision utility. If the opponent concedes, then we concede; if they increase their share, we increase ours. Then, we estimate the opponent's utility weights as the mean value of their offers. For instance, if an opponent offers $[1, 1, 1]$ then $[1, 1, 0]$, then the utility is estimated as

$$V_{opp} = [v_x, v_y, v_z] = 6 * \sum_{i=0}^{t} \frac{[x, y, z]}{x + y + z} \tag{3.11}$$

We then implement the Simplex algorithm [16] to maximize this value, fixed upon the decision utility we calculated prior. While this assumes the opponent makes concessions in particular (preference-based) fashion, it remains a question whether the neural agent can uncover the correct concessions to make.

Chapter 4

Results

4.1 Theoretical Decision Utilities

4.1.1 Utility Moments

Before proceeding to results against time-based agents, we first derive the theoretical optimal strategies. Denote the decision utility of the time-based opponent as

$$U_{opp}(c,t) = P_{res} + (1 - P_{res})(1 - (\frac{t}{T})^{\frac{1}{c}})$$

This is in essence the same as Equation 2.5. In our case, P_{min} is the reserve price P_{res} and P_{max} is normalized to 1. Then our utility is:

$$U(d) = \Big(1 - U_{opp}(c,t)\Big)d^t \tag{4.1}$$

The maximal point must be one where the marginal utility $\frac{\partial U}{\partial t}$ is 0. Before we take the derivative of $U(d)$, we first take the derivative of U_{opp}.

$$\begin{aligned}\frac{\partial U_{opp}}{\partial t} &= \frac{\partial}{\partial t}P_{res} + (1 - P_{res})(1 - (\frac{t}{T})^{\frac{1}{c}} \\ &= (1 - P_{res})\frac{\partial}{\partial t}(1 - (\frac{t}{T})^{\frac{1}{c}}) = -\frac{1 - P_{res}}{cT^{\frac{1}{c}}}t^{\frac{1-c}{c}}\end{aligned}$$

We then solve for our marginal utility by the product rule.

$$\begin{aligned}\frac{\partial U}{\partial t} &= d^t \ln d(1 - U_{opp}) + d^t\Big(-\frac{\partial U_{opp}}{\partial t}\Big) \\ &= d^t\Big(\ln d(1 - U_{opp}) + \frac{1 - P_{res}}{cT^{\frac{1}{c}}}t^{\frac{1-c}{c}}\Big)\end{aligned} \tag{4.2}$$

Setting the reserve price to 0 in our experiments, we can derive a much more

elegant expression, as $U(c,d,t) = \left(\frac{t}{T}\right)^{\frac{1}{c}} d^t$.

$$\frac{\partial U}{\partial t} = \left(\frac{1}{T^{\frac{1}{c}}}\right)\left(\frac{1}{c}t^{\frac{1-c}{c}}d^t + t^{\frac{1}{c}}d^t \ln d\right) = \frac{t^{\frac{1}{c}}d^t}{T^{\frac{1}{c}}}\left(\frac{1}{ct} + \ln d\right) \tag{4.3}$$

It is a simple matter to check the second derivative is negative, hence the expression for the condition for the maximal point is

$$t = \frac{-1}{c \ln d} \tag{4.4}$$

Interestingly, this value does not depend on the total time T and since P_{res} is a linear transformation of the utility function, this optimal time depends only on the concession factor and discount rate. The optimal stopping time can be expressed as:

$$T_{OPT} = \begin{cases} T & \text{if } \frac{-1}{c \ln d} > T \\ \frac{-1}{c \ln d} & \text{otherwise} \end{cases} \tag{4.5}$$

The theoretical values are shown in Fig. 4.1. A strong phase transition occurs along the $\frac{-1}{c \ln d}$, demarcated by the clearly lighter region.

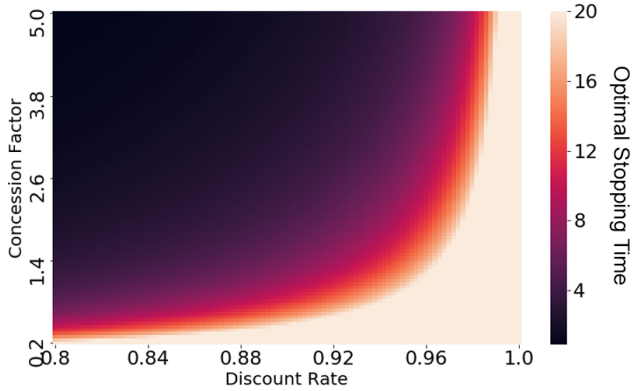

Figure 4.1: Theoretical optimal stopping time over concession factor and discount rate.

Additionally, we compute the second derivative of the utility, as it accounts for the error analysis of neural agent in Section 4.2, and the n-th moment for generality.

$$\begin{aligned} \frac{\partial^2 U}{\partial t^2} &= \frac{d^t}{T^{\frac{1}{c}}}\left((\ln d)^2 t^{\frac{1}{c}} + \frac{\ln d}{c}t^{\frac{1-c}{c}} + \frac{1}{c}\left(\ln d\, t^{\frac{1-c}{c}} + \frac{1-c}{c}t^{\frac{1-2c}{c}}\right)\right) \\ &= \frac{d^t}{T^{\frac{1}{c}}}\left((\ln d)^2 t^{\frac{1}{c}} + \frac{2\ln d}{c}t^{\frac{1-c}{c}} + \frac{1-c}{c^2}t^{\frac{1-2c}{c}}\right) \end{aligned} \tag{4.6}$$

$$\frac{\partial^n U}{\partial t^n} = \frac{d^t}{T^{\frac{1}{c}}}\sum_{i=0}^{n}\binom{n}{i}(\ln d)^{n-i}\frac{1}{c^i}\left(\prod_{j=1}^{i} 1-(j-1)c\right)t^{\frac{1-ic}{c}} \tag{4.7}$$

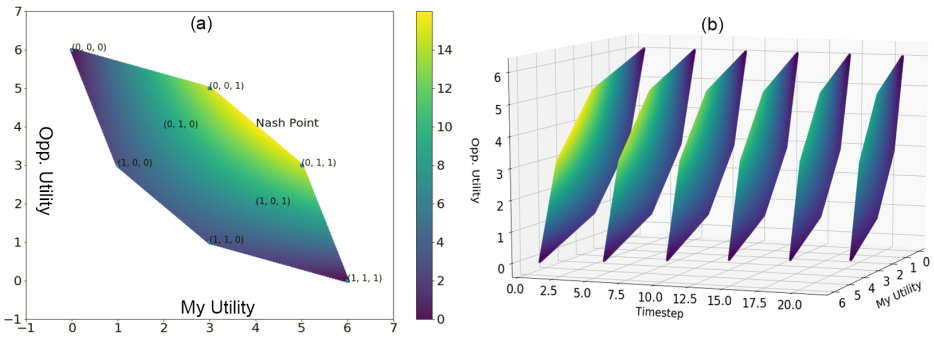

Figure 4.2: Outcome space of Negotiation. Fig. 4.2a) shows the outcome space at $t = 1$. The vertices of the outcome space polytope is mapped from the vertices of the action space, at $(0, 0, 0), (0, 0, 1), (0, 1, 1), (1, 1, 1)$. Fig. 4.2b) shows the evolution of the polytope over time, with a discount rate of 0.9. The decision to counter-offer is made by the subsequent time-step, rather than the current. The colors denote the Nash Product, with the Nash solution lying at $(4, 4)$.

4.1.2 Outcome Space

In the field of automated negotiation, preferences are typically visualized through an *outcome space plot*. The axes are utilities of Player A and B. Possible outcomes $\omega \in \Omega$ are mapped to $(u_A(\omega), u_B(\omega))$. Fig. 4.2 shows this plot for our negotiation process. In a), the Pareto frontier is shown by the right-most edges of the polytope.

By theorems of fixed points and the simplex algorithm [85, 16], the vertices in the outcome space must come from the vertices in the action space $[0, 1]^3$. The action space vertices that outline the frontier are found to be $(0, 0, 0), (0, 0, 1), (0, 1, 1), (1, 1, 1)$. Intuitively, these are points that offer the greatest marginal utilities to P1 and P2, based on their value weights $w_A = (1, 2, 3)$ and $w_B = (3, 2, 1)$. The piece-wise equation for the Pareto Frontier $PF(U_A, U_B)$ is given as follows, in Equation 4.8:

$$PF(U_A, U_B) = \begin{cases} U_B + \frac{1}{3}U_A - 6 = 0 & \text{for } 0 \geq U_A \geq 3 \\ U_B + U_A - 8 = 0 & \text{for } 3 \geq U_A \geq 5 \\ U_B + 3U_A - 18 = 0 & \text{for } 5 \geq U_A \geq 1 \end{cases} \tag{4.8}$$

This equation allows us to calculate the bid distribution and determine the *efficiency* of an agent's bid strategy, provided in Section A.2 in the Appendix. Furthermore, the Nash Solution $(\max U_A U_B)$ lies at $(4, 4)$ given by the offer $(0, 0.5, 1)$, provides a benchmark for the *fairness* when playing against behavior-based agents.

4.2 Acceptance Strategy

4.2.1 Behavioral dynamics: Cliff-walking vs optimal play

The central question for an acceptance strategy is when given an offer, whether or not to accept or wait for potentially better future offers. However, if the agent fails to accept before the deadline, then the conflict deal is enacted and both agents do not receive any reward. Given a discount rate and opponent concession factor, the goal is to find the best moment to accept an offer, inferring from their prior offers.

Thus, the acceptance strategy can be seen as an optimal stopping problem with an additional cliff-walking problem to solve. Fig. 4.3 shows the loss, rewards, and playing time as the network trains against a linear agent ($c = 1.0$) with no discount ($d = 1.0$). Through stochastic sampling of new points, the agent notices greater reward by waiting, illustrated by gradual trends in playing time (green). However, once the agent reaches the deadline at 20 rounds, the conflict deal is enacted and a reward of -1 is issued, producing a large loss. We present only the multivariate case, as results for the univariate case are the same but with lower complexity.

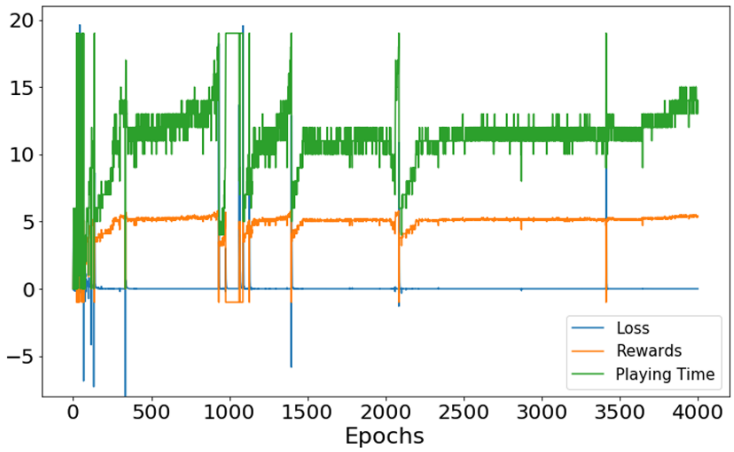

Figure 4.3: Loss, rewards and total time of the DRL agent training against a time-based agent with $c = 1.0$. After a few epochs of random search, the agent learns that increased playtime comes with greater reward. However, as this time increase to the deadline,the reward drops sharply.

To analyze the stopping time, we consider the evolution of acceptance probabilities during gameplay against time-based opponents. Fig. 4.4 shows the logit values used in Eq. 3.3 and acceptance probabilities against Boulware, Linear, and Conceder agents.

The first row shows the acceptance probabilities at each time step. The cumulative probability (orange) denotes the likelihood the game ends at a

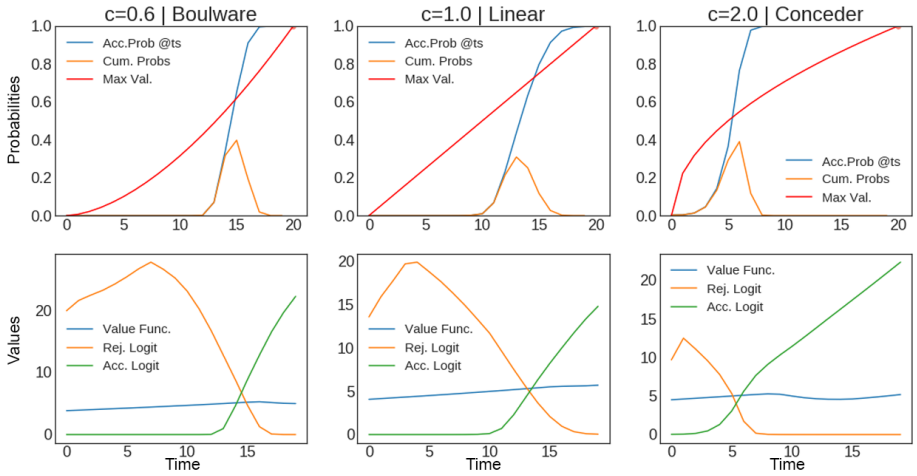

Figure 4.4: Evolution of acceptance probabilities and logit values without discounting ($d = 1.0$). Row 1 shows the acceptance probabilities at each time step (blue), the cumulative probability of first success (orange), and max value at each time step. The pink point denotes the optimal value (Eq. 4.5). As the concession factor c increases, stopping time decreases. Row 2 shows where the logits "change places," corresponding to where the cumulative probability is maximal.

certain time step, given as:

$$P_{cum}(t) = P_t(\text{Accept}) \prod_{i=0}^{t-1} P_i(\text{Reject}) \tag{4.9}$$

Since the discount rate is 1.0, the optimal value is waiting until the final point in time. The decrease in stoppage time shown by the right-shifting cumulative probability is sub-optimal, although this is not uncommon in conservative agents. In the value function (blue, second row), there is also a slight decrease after the logit values cross. This indicates that the expected reward at these times may be the same.

Another way to see this is to consider the marginal utility over time. Since Boulware agents only concede towards the end, the Neural agent is forced to wait to achieve comparable results, whereas it may be "satisfied" earlier against Conceders. The marginal utility of the Boulware agent is thus much greater towards the end, whereas marginal utility is high at the beginning against Conceders. Explicitly, for $U = 1 - 1 + \left(\frac{t}{T}\right)^{\frac{1}{c}} = \left(\frac{t}{T}\right)^{\frac{1}{c}}$, the expression for marginal utility is:

$$\frac{\partial U(c)}{\partial t} = \frac{1}{cT^{\frac{1}{c}}} t^{\frac{1-c}{c}} \tag{4.10}$$

This analysis is corroborated further once we introduce the discount rate. Fig. 4.5 shows the acceptance probabilities and logits once discounting is

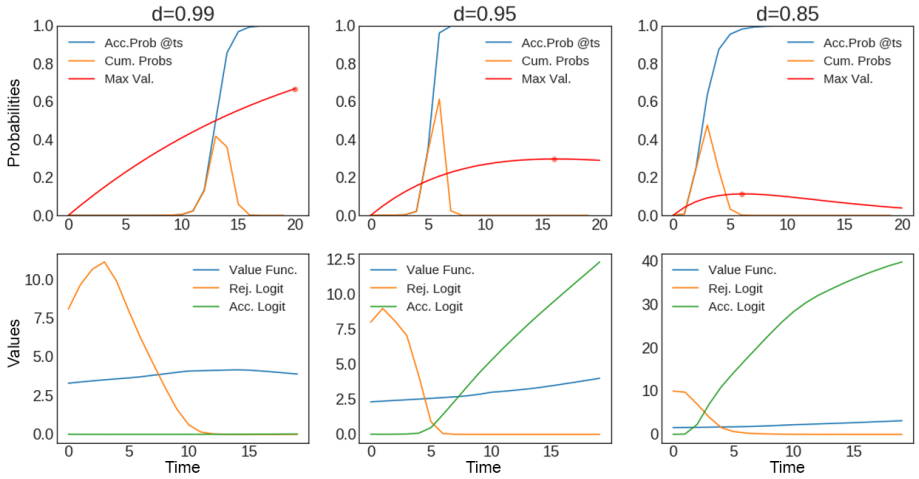

Figure 4.5: Evolution of acceptance probabilities and logit values with discount. Row one shows the acceptance probabilities at each time step (blue), the cumulative probability of the first success (orange), and the pink point denotes the theoretical maximum prescribed by Equation 4.5. As the discount rate increases, the optimal maximum and stoppage time decreases.

introduced. Looking at the red curve, $d = 0.99$ has positive marginal utility and for $d = 0.85$, the marginal utility is negative from time step 7 onwards. For $d = 0.95$, the utility function is relatively flat after time step 10, which means the marginal utility is close to 0. Here, we observe the greatest time deviation.

4.2.2 Marginal Analysis: Marginal Utility determines Error

Due to the stochastic nature of deep learning, it's difficult to construct a precise mathematical proof of how changes in marginal utilities push against each other. However, we can test this empirically. The neural agent played against a set of different agents, with concession factors of 0.95, 1.5, 2, 3, 5, 10. In Fig. 4.6a), the curves show our max utility (Equation 4.1) and the red dot shows the optimal stopping time given by Equation 4.5. Note, as c increases, the curves grow sharper and since $c > 1$, the magnitude of the second derivative strictly increases.

Fig. 4.6b) shows an inverse relationship between the time error and the reward error. The "peakier" the curve, the more likely the Neural net selects the optimal time. However, deferral by even one time step leads to large amounts of diminished utility, hence creating the larger reward error. In contrast, using the second derivative derived in Equation 4.6, we observe in Fig. 4.6c) that as the second derivative approaches 0, the time error increases.

Having shown what produces the reward and time errors, we can address

our sub-problem about limitations. For future work, we may dynamically reduce the learning rate using the second derivative and distance to the deadline for better convergence. Numerical results are summarized in Table 4.1.

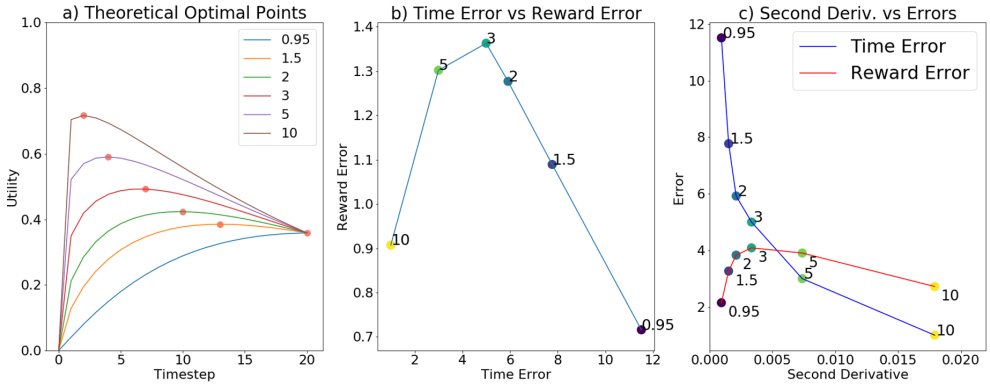

Figure 4.6: Optimals vs second derivative from 100 gameplays. Rew. error scaled by 3.

c	0.3	0.95	1.5	2	3	5	10
Time Error	3.95	11.62	7.81	5.86	4.99	3.0	**1.0**
Reward Error	**-0.0234**	-0.712	-1.054	-1.278	-1.366	-1.302	-0.907
Second Deriv. ($x100$)	0.0722	-0.094	-0.152	-0.223	-0.389	-0.777	-1.886

Table 4.1: Tabular Results of 100 gameplays sat different concessions.

4.2.3 Preference-based concessions produce fairer outcomes

Finally, we consider optimality. Since the final offers depend on the time-based agent, so do the optimality measures. Thus, the way opponent agent algorithmically constructs their offers will appear differently in the outcome space. Fig. 4.7 shows the distribution of accepted offers after 400 gameplays, with $c = 1.0$ and $d = 0.94$. The first randomly samples from the plane that satisfies the following condition:

$$U_d(t) = w^T X = w_1 x_1 + w_2 x_2 + w_3 x_3 \tag{4.11}$$

where U_d is the decision utility at time t and w_i is the weighed utility for issue x_i. The second uses a *preference-based, monotonic concession* strategy—it satisfies Equation 4.11, but concedes starting from the issue it values the least ($\min\{w_1, w_2, w_3\}$). Multivariate Gaussian noise with a standard deviation of 0.05 is added to prevent deterministic offers. When the time-based agent uses preference-based, monotonic concession strategies then this guarantees offers to lie on the Pareto Frontier.

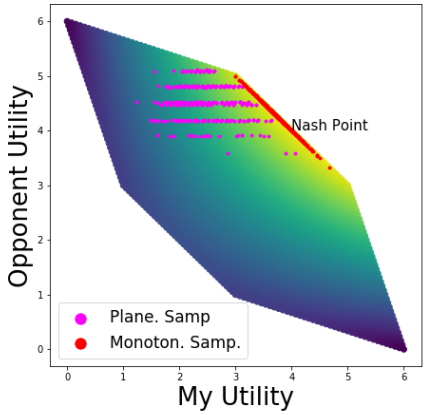

Figure 4.7: Distribution of final, accepted offers ($c = 1.0$, $d = 0.94$). The preference-based bidding (red) produces offers on the Pareto Frontier. Planar sampling (magenta) occur in intervals as the time-based agent samples points based on $u_B(t)$.

	d_{Nash}	$BD(\Omega)$ (d_{Pareto})	Av. Reward	Av. Time
Planar Samp.	1.64	0.54	1.84	4.89
Preference-based Concession	**0.46**	**0.00**	**2.61**	**5.7**
Pure Random	1.92	1.24	N/A	N/A

Table 4.2: Sampling Results

The preference-based method produces offers that lie on the Pareto Frontier. Because of this optimality, the neural agents play on average a longer time when its opponent follows this strategy. Random planar sampling yields considerably better results than pure random sampling, with a bid distribution difference of 0.7 (shown in Table 4.2). The magenta points in Fig. 4.7 arise because, for every time step, the decision utility is fixed for fixed c. Table 4.2 summarizes the mean outcomes of the gameplays, with the preference-based concession performing the best.

4.3 Bidding Strategy

4.3.1 Precision in Single-Issue Negotiation

Before evaluating performance on the multivariate case, we verify the univariate case. Fig. 4.8 shows the action policies given by Cauchy distributions for specific decision utilities. As the concession factor increases, the distribution of means transfers from right- to left-skew. This can be attributed to the magnitude of the marginal utility. Cauchy means are clustered tightly

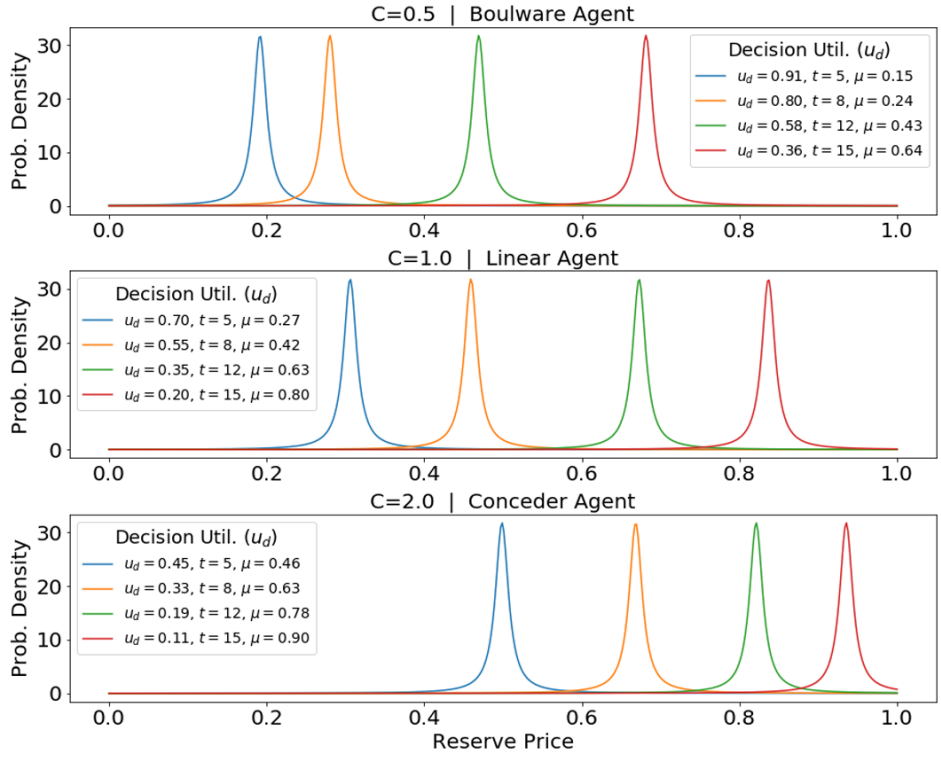

Figure 4.8: Cauchy distributions based on concession factor and decision utility. When c is low (Boulware), Cauchy means are clustered tightly for low t, spread out for high t. When c is high (Conceder), values are clustered for high t and spread out for low t.

for the Boulware agent when t is low, as the marginal utility is low early on. However, as time passes and the Boulware agent begins to concede greatly, the distance between means increase. Conversely, when the opponent is a Conceder, concession begins early, so marginal utility is large when t is small, leading to right-skew. The Conceder case is not as pronounced as the Boulware case due to the cliff at the deadline. As expected, means are spaced out linearly against linear agents.

In sum, the change in decision utility affects the distribution of the means. For completeness, Fig. A.1 in the Appendix shows a heat map of how the neural agent's utility changes in respect to the opponent's.

4.3.2 Multivariate Training Dynamics

Next, we present the multivariate case. We trained on a grid of concession factors for fixed discount rates. We denote the three issues as issues X, Y, and Z. Fig. 4.9 shows the first 1000 epochs, using a multivariate Cauchy distribution with $c = 0.3$ and no discount. Unlike training Accept Net, cliff-

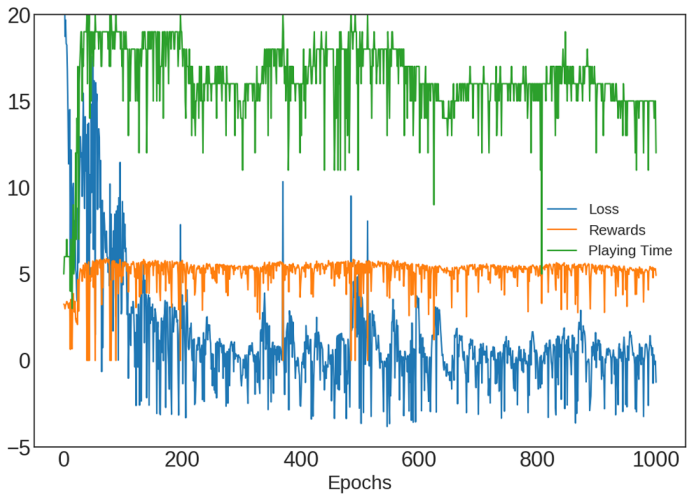

Figure 4.9: Training statistics of the Cauchy Offer Net. The neural agent learns to induce rejection from the time-based agent so negotiation ends near the deadline.

walking is less present, as the final action (accept) lies with the opponent and time-based agents are very likely to accept. The agent quickly learns to wait longer, converging at a higher time step.

However, unlike accept net, this is not simply a binary action where rejecting an offer leads to the next round. The agent has to produce offers that induce rejection from the time-based agent. Fig. 4.10 shows how Cauchy means vary over time. The first row shows the individual Cauchy means for issues X, Y and Z respectively. Shown in light blue is also the normalized utility (divided by the total possible utility of 6) and the equilibrium payout from 100 gameplays. The second row shows the opponent's decision function (blue) and our maximum utility at each time step. The value estimate given by the value net is shown in green. At the bottom, the red line shows the mean stoppage time, with the distribution of times shown with a kernel density estimate.

Since the cliff-walking aspect is not as prominent (the only case where the conflict deal is enacted is if the agent proposes the full amount at the end), all stoppage times are relatively high, although diminishing stoppage time is still observed when the marginal utility is lower. For instance, when $c = 1$, the marginal utility is constantly $\frac{1}{20}$ and the second derivative is 0, the stoppage time is 15.1. Comparably, when $c = 0.3$, the marginal utility is increasing and the mean stoppage time is 16.9.

Note issue X varies the most, either through concession (Fig. 4.10a)) or increase of the offer value as shown in Fig. 4.10c). At a glance, this may be counter-intuitive, since a change in Y or Z would yield the most marginal gains for the agent. However, the opponent values issue X the most, which

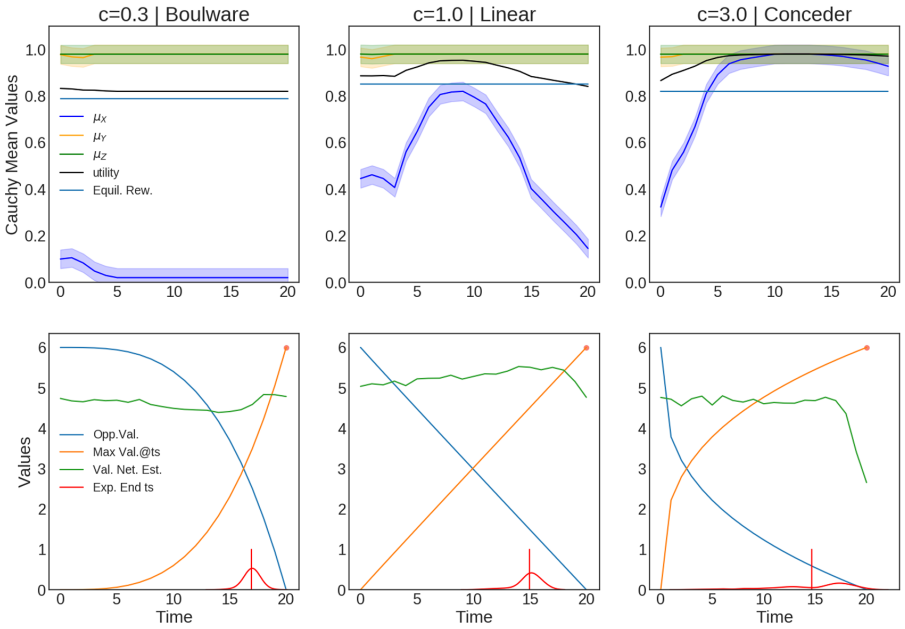

Figure 4.10: Variation of Cauchy means over time with no discount rate. The expected reward hovers around 84 percent of the full utility. All stoppage times are high, but still decreases as the concession factor increases due to reduced marginal utility.

means X produces the largest amount of gradient for the least amount of loss during concession. As a result, the agent learns the negotiate close or along the Pareto Frontier, which we show later in the distributional analysis. Secondly, there is a clear progression between Boulware and Conceder strategies when comparing the linear agent to the Boulware agent.

Additionally, in the bottom row, the green value function remains fairly constant throughout, until the drop off towards the end induced by the deadline. The value function remains flat since the expected value is constant—so long as the neural agent sticks to its strategy, the payout will not change. While performance is not optimal, it achieves more than 80% of the optimal which is typical of risk-averse agents whose behaviors are generally conservative on estimates [68].

Next, we compare this to the case when discounting is introduced, using the beta distribution as an example. Fig. 4.9 shows the gameplays of a neural agent using the beta-distribution. The first row shows the evolution of the multivariate distribution means (blue, orange and green for X, Y, and Z respectively), the evolution of the normalized utility (black), and the reward under 100 gameplays. The second row shows the theoretical maximums (orange) and mean stoppage time (red). Immediately, we observe that the agent begins sampling around the same initial values, then alters its

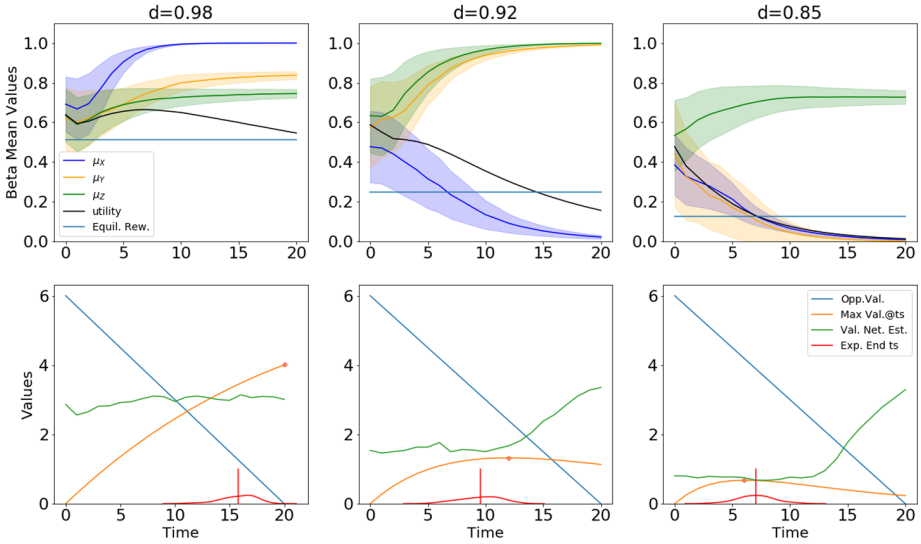

Figure 4.11: Variation of beta distribution means over time with discount rate. Each mean is calculated using Equation 3.8. Gameplay results fall consistently within 10% of the expected optimal stoppage time. Values begin around the same region with high variance as the agent is unsure what opponent it is playing against, but decreases with time as certainty towards its opponent's strategy grows. $c = 1$.

strategy as it learns more about the opponent. The way it alters its strategy varies depending on its own inherent discount rate, which demonstrates *adaptability*.

Finally, issue X is again the issue with the most variation, with the same argument that its change produces the highest gradients during gameplay, due to the opponent valuing X the most. Mean stoppage time is close to the optimal, with 8% mean deviation. While this is not precisely optimal, it is quite good. To understand what causes this limitation, we analyze the probability distributions.

4.3.3 Offer Strategy requires Sensitivity to Variance

We compare the outcome space of agents using Gaussian, Cauchy, and beta distributions, after playing 3000 rounds against batches of mixed opponents. Fig. 4.12 shows the distribution of final offers given by each agent, with the addition of a random agent, when playing against a linear agent with no discount rate. Since time-based agents make monotonic concessions, lower y-values imply longer gameplay times.

As expected, the random agent produces offers distributed randomly in the outcome space. At a glance, the beta distribution outcomes bear the most resemblance to the random agent. This is due to the beta distribution's initial high variance. This can be adjusted by increasing the constant added

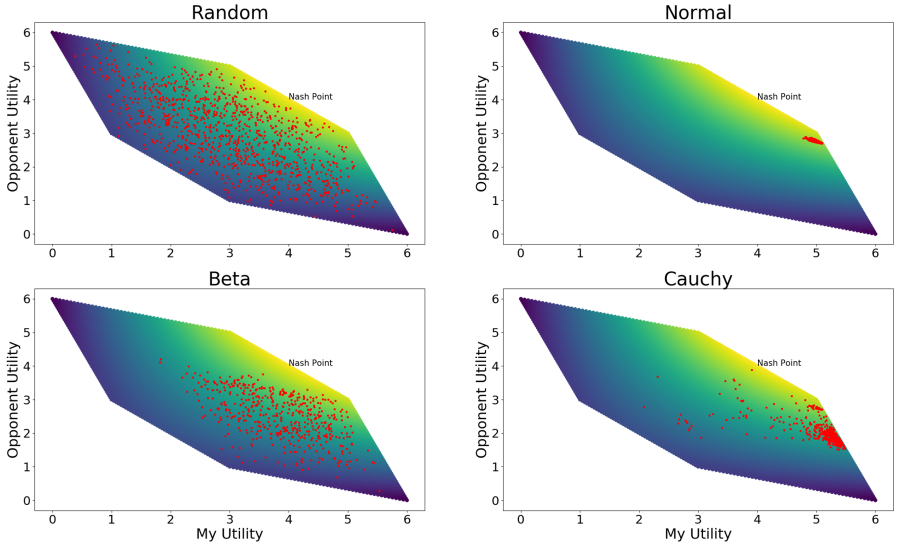

Figure 4.12: Distribution of outcomes based on different sampling distributions. Neural agents were trained on 3000 games, then played against a linear agent. The random agent plays. Statistics are summarized in Table 4.3

to the initial values of α and β. However, also note that the scattered points are on average greater than 3.

The normal distribution produces the most consistent results, with results clustered around the $(0, 1, 1)$ vertex. The Cauchy distribution performs similarly, but on average performs better, with a maximum value of 5.5 and average 5.3. However, it also has a much greater variance when compared to the Normal distribution. We can conclude convergence to optimal play requires sufficient initial variance to prevent convergence to local optima. Additionally, the maximum value achieved by any of these distributions was achieved by the beta distribution. These values are presented in Table 4.3.

	d_{Nash}	$BD(\Omega)$	Av. Reward	Av. Time	Reward. Range
Rand. Samp.	1.901	1.246	NA	NA	NA
Beta	1.741	1.0018	3.587	10.11	5.378
Normal	**1.585**	**0.0815**	4.993	11.03	**0.294**
Cauchy	2.261	0.218	**5.051**	**14.66**	4.403

Table 4.3: Gameplay outcomes from 400 games against linear agent. Sampling through the normal distribution givese the fairest and most consistent results, whereas the Cauchy provides the highest expected reward.

The normal distribution produces consistent results, with the lowest bid distribution and reward range, and also is closest to the Nash Solution. In expectation, the Cauchy distribution produces better results but with a higher bid distribution and is farther from the Nash Solution. However,

having a large distance from the Nash Point is not necessarily bad, as exploiting the opponent's strategy leads to higher rewards. Without discount, the neural agent can improve its own outcomes by waiting.

The four panels in Fig. 4.12 reveal two opposing forces that make continuous DRL difficult in this domain. Convergence to optima requires high variance, yet avoiding the conflict deal requires low variance to prevent sampling the conflict deal. For further proof, consider that the time error decreases with discount rate, comparing Fig. 4.10 and Fig. 4.11. The discount rate shifts the optimal away from the cliff, thus sampling around the optimal produces less error due to slow change in marginal utility, and the smooth reward function allows more accurate function approximation [40, 76].

The next step is to compare the variances of the three distributions, and their sensitivity to parameter change. The Normal's variance is directly parametrized by the action network. In contrast, the variance of the beta distribution depends on both shape parameters α and β:

$$Var_\beta = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

which means large, simultaneous increases in both α and β is required to lower variance, leading to slow convergence. In contrast, the Cauchy distribution famously does not have a theoretical mean, variance or kurtosis, due to laws of integration.

This points to why the Cauchy distribution works better, all things held equal. It is parameterized as directly as the normal distribution, but is also has "heavy-tailed" and "peakier" than the Gaussian. This slower decay in the tails means lower variance sensitivity, hence avoiding convergence to local optima. At the same time, the Cauchy distribution also has a much higher peak than the Gaussian, which means there is less cost in accuracy when sampling.

In regards to our study's objectives: the results from Sections 4.2 and Section 4.3 show slow convergence when variance is high and sub-optimal convergence when variance is low due to lack of action exploration as the primary limitation. This suggests the learning rate can be adjusted through marginal utilities, the distribution's kurtosis (peaky-ness and tail-behavior), and the variance sensitivity for faster convergence and efficient outcomes. A more aggressive learning rate can curtail distributions with lower variance sensitivity. Furthermore, variation in concession factor and discount rate yields different strategies from Offer Net, thus demonstrating adaptivity.

4.4 Self-Play: The Emergence of Fairness

So far, we have addressed sub-problems related to training barriers and demonstrated exploitative capabilities against time-based agents. While

play against time-based agents provides clear benchmarks due to monotonic time-based concession, play against behavior-based agents is required to evaluate behavioral traits such as fairness. In this section, we first present a game-theoretic framework of our games, then the results for single- and multi-issue self-play, then against two variants of tit-for-tat agents.

4.4.1 Game-theoretic Framework

We introduce a few game-theoretic concepts required for in-depth behavioral analysis. An *extensive game* consists of a set of players N, a set of sequences H that denote possible game trajectories. A *game tree* describes this trajectory of states, round-by-round. A *Nash Equilibrium* (NE) denotes an outcome where no player wants to willingly deviate. In extensive games, a strategy profile $s*$ is an NE if

$$O(s^*-1, s_i^*) \geq O(s^*-1, s_i) \qquad \forall s_i \in S_i$$

S_i denotes the strategy set of player i. Let $(< a_1, a_2, ... >, < b_1, b_2, ... >)$ denote the strategy profiles, where each bracket contains a player's sequence of moves [54].

Figure 4.13: Centipede Game

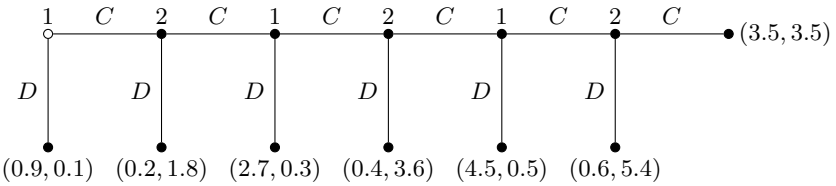

Figure 4.14: Game tree of the centipede game. Every round, the "pie" grows by 1, ending with 7. Players can split it evenly at the end (cooperate every turn), or defect. By backwards induction, the SPNE is $(< D, D, D >, < D, D, D >)$, with reward$= (0.9, 0.1)$.

In extensive games, the concept of *sub-games* describes part of the game tree which function as a game itself [54]. Fig. 4.13 shows the game tree of the *centipede game*, a canonical game in game theory, and Fig. 4.15 shows the game tree of a bargaining game. In this instance of the centipede, the total size of the pie increases by 1 at every time step. The players can choose to wait or defect. Consider the rightmost node in Fig. 4.13 labeled 2, denoting P2's decision to cooperate or defect. Since the pay-off of defecting yields a reward of 5.4 over 3.5 from cooperating, P2 will defect if they are rational. The sub-tree stemming from 2 can then be reduced to $(0.6, 5.4)$.

Once P1 realizes P2 will defect, P1 will also defect as this yields a higher reward. This process continues until P1 defects in round 1. The process of iteratively reducing up the tree is known as *backwards-induction*. The result

at the end is a *sub-game perfect Nash Equilibrium* (SPNE), a type of NE that is also the equilibria of sub-games. The SPNE in this game is found to be $< D, D, D >, < D, D, D >$. Ironically, if both players waited until the end, they would receive higher rewards. Hence, for the centipede game, we expect to see cooperative agents wait until the end, while rational agents defect at the beginning.

Figure 4.15: Bargaining Game

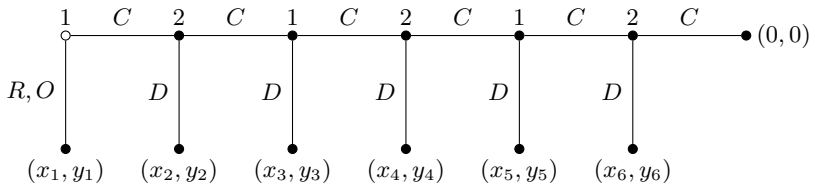

Figure 4.16: Game tree of a bargaining game. With an initial size of 1, the size diminishes by 0.9 each time step, ending with 0 after the sixth and last round, when the conflict deal is enacted. The sum of the rewards are subject to $x_i + y_i = 0.9^{t-1}$.

4.4.2 Univariate self-play results

Before training with self-play in the multi-issue domain, we consider a simplified version. Instead of giving offers in $[0, 1]^3$, consider the case where bidding actions are constrained to a binary decision—either offering $(0.5, 0.5)$ or $(0.9, 0.1)$. Thus, agents can either offer a low amount to their opponent (rational behavior), or a fair amount. These four choices can be summarized as:

(L, L): Offer low, reject nothing. This is typically the SPNE, thus **rational (G1)**.
(H, L): Offer high, reject nothing. This is **altruistic (G2)**.
(H, H): Offer high, accept high. This agent is **fair (G3)**.
(L, H): Offer low, accept high. This one is often disregarded, as it is a hardliner **G4**.

Fig. 4.17 shows the training results for the bargaining and centipede game, with the discount factors set to 0.9 and 1.3 respectively. Note, by setting the discount rate to greater than 1, the bargaining game effectively becomes a more complex version of the centipede game. For the bargaining game in Fig. 4.17a), the reward for P2 is initially low, then increases with time approaching 1 round. Conversely, the reward for P1 decreases. We infer that P2 learns to reject P1's offer, and P1 learns to accept. This play is close to rationally optimal. If the agents were perfectly rational, the game would end immediately. However, P2 adopts a strategy that forces play to go on— we analyze the reason for this further in the multivariate case.

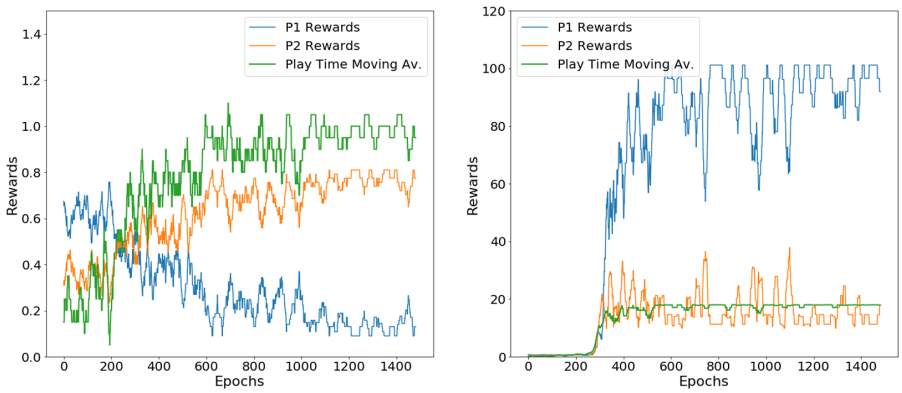

Figure 4.17: Training results for self-play, for the centipede game and bargaining game. Depicted are P1's results(blue), P2's results(orange) and the total playing time (green).

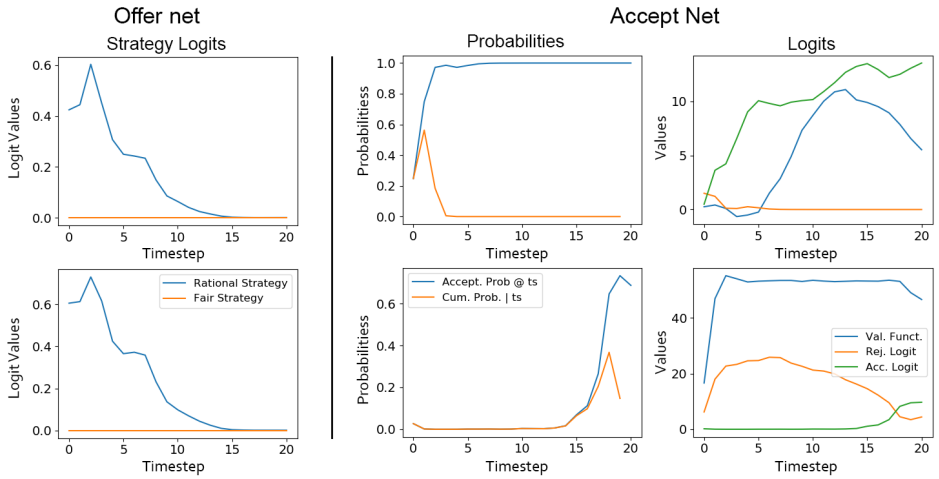

Figure 4.18: Comparison of offer and acceptance logits for the (a) bargaining and (b) centipede game. In both, Offer Net initially gives rational offers, then over time shifts to fair offers to increase acceptance probability. Accept Net accepts early in the bargaining game, and late for the centipede game, aligned with the optimal stopping times.

In the centipede game in Fig. 4.17b), the players learn to play close to 20 rounds, maximizing the "interest" accumulated. The total size of the pie is $1.3^{19} = 146$. By the final round, P1 holds a mixed strategy yields 47% rational offers $(0.9, 0.1)$ and 53% fair offers $(0.5, 0.5)$. As the time series shown are the running averages, the big dips show brief spans where the P1 adopts a fair strategy. These dynamics can be seen more clearly by observing how decision logits evolve during gameplay. Fig. 4.18 shows the probabilities of giving rational and fair offers at each time step, and the

probabilities of accepting an offer (for one game trajectory).

In both cases, agents start-out giving low offers to their opponents. However, as time moves forward, the probability of a fair offer increases, to increase the probability of acceptance. For the bargaining game (a), the stoppage time reaches a maximum close to the beginning. This suggests the network, through gameplay, learns outcomes similar to backward-induction. Similarly in the centipede game, the network learns to wait, leveraging "interest" to accept near to the deadline, which also indicates *cooperative behavior*. Together, we conclude the neural agent learns to accept optimally and as time moves forward, shift its behavior from G1 (rational) to G3 (fair).

4.4.3 Multivariate Self-play

Now, we extend analysis to the continuous, multi-issue case. Fig. 4.19 shows the training dynamics of the Offer Net and Accept Net, whose final rewards are plotted per epoch, with the discount rate set to 0.95. Initially, the Offer Net (blue) has higher reward— as long as some reward is given to Accept Net, the Accept Net will accept it. However, around epoch 1600 the Accept Net learns to invoke the conflict deal. This is demarcated by the large drop in reward for both blue and orange to -1. After which, the Offer Net must concede some by offering a fairer amount.

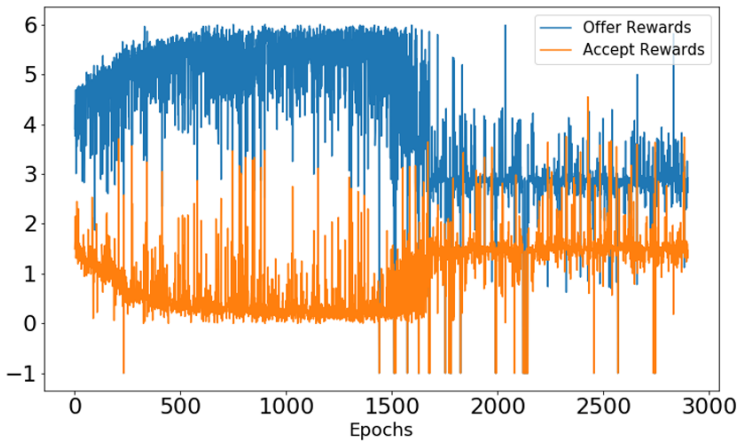

Figure 4.19: Multivariate self-play for negotiation. Offer Net concedes value after Accept Net adopts a mixed strategy with the conflict deal threat.

In sum, by including some probability of the conflict deal, neural agents force a counter-offer that is fairer. This departs from classical game theory, as an example of a *non-credible threat*. A non-credible threat describes actions that perfectly rational agents will not carry out, as it would also leave themselves worse off [54]. The adaption of non-credible threats is also observable in the uni-variate centipede game, with periods of dips in P1's

reward, following the conflict deal. For discounted bargaining, convergence to low playing time suggests that heavy discounting acts as a similar threat—if you do not offer a fair deal, I will drag on the negotiation. By keeping non-credible threats part of a mixed strategy, fair outcomes can evolve.

This is significant because it agrees with results from evolutionary game theory. We previously mentioned in Section 4.4.1, that reputation produces fairness in the repeated ultimatum game. Nowak et al. showed this through the same, exact simplified mini-game (bids restricted to low and fair offers), then full bidding space using population-based experiments [52]. Populations of G1, G2, and G3 played against each other, and "reproduced" based on their utility. After many rounds, results showed that the rational agent (G1) dominated. However, if the prior acceptance history was available, which showed agents rejecting below a certain threshold, then a population of fair agents (G3) who offered high and accepted high would emerge.

In other words, there is a strong similarity between non-credible threats in our neural agent and the rejection of low offers (using reputation) in evolutionary strategies. This is by far the most interesting result, and it's important to note evolutionary methods and RL are often framed as competing choices for agent design [67]. Similar results produced in these two fields may drive future research directions.

4.4.4 Against Tit-for-Tat Agents

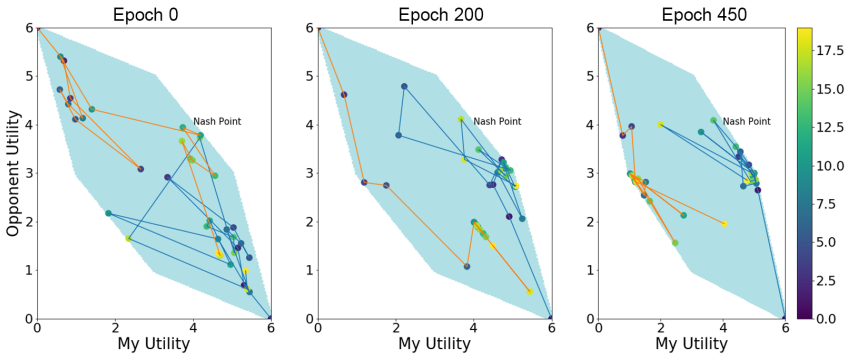

Figure 4.20: Evolution of play against relative tit-for-tat agents.

Finally, we present results against relative TFT and the Bayesian TFT agent. Since this investigation studies whether the acceptance and bidding strategy can adapt and induce promising counter-bids, each game is designed as follows: the TFT agent makes a bid, then the neural agent makes an acceptance decision and counter-bids. Thus, the game can only end on the neural agent's acceptance.

Against the relative TFT agent with no discounting (Fig. 4.20), the neural agent converges to the $(0, 1, 1)$ vertex. Yellow represents the ending

epoch and we observe the neural agent's utility is greater than 3. Notably, the TFT agent makes offers opposite of the Pareto Frontier. This arises as the relative TFT agent measures concession with respect to its own utility. As we've analyzed in the time-based opponents, DRL agents are prone to adjusting variables that yield the greatest rewards and lowest losses. To the TFT agent, this is reversed, prompting it to concede the issue it values most and propose away from the Pareto frontier. This demonstrates *adaptivity*.

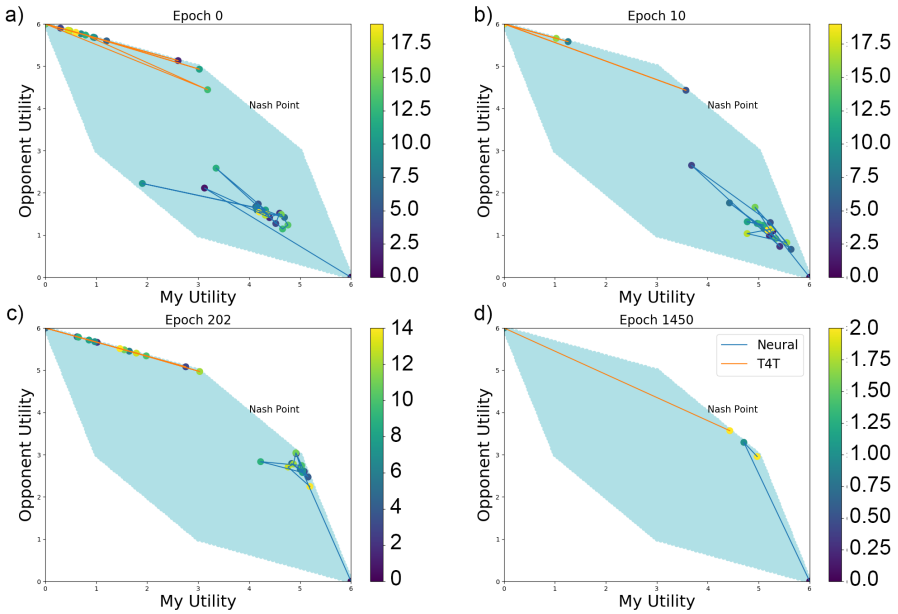

Figure 4.21: Evolution of play against Bayesian tit-for-tat agents. The agent learns to accept early and around the Nash point.

In contrast, the neural agent *cooperates* with the Bayesian TFT agent. In Fig. 4.21a (epoch 0) the neural agent performs randomly and suboptimally. Note, the color bar represents time step. However, the bid direction drifts towards the $(0, 1, 1)$ vertex (Figs 4.21(b) and (c)). By epoch 1450, the neural agent learns to induce results near the Nash point, in only four moves due to the discount rate.

With results from time-based agents and self-play, our analysis shows that when concession is necessary, the bidding strategy gravitates toward to $(0, 1, 1)$. This ensures near Pareto Optimal payoff if its offer is accepted. Exploitation occurs against time-based and relative TFT agents, while fairer outcomes arise with more complex agents.

Chapter 5

Conclusion

5.1 Summary of Discussion

Bilateral negotiation presents a unique domain that combines discrete and continuous control problems. Furthermore, the deadline produces a utility function analogous to cliff-walking. This paper is a fundamental evaluation of actor-critic models for negotiation, measuring its ability to exploit, adapt, and cooperate.

The neural agent shows clear **exploitative behavior** against time-based agents. For acceptance, the neural agent demonstrates precise logit switching behavior, in transitions between rejecting and accepting offers. The acceptance strategy resembles a conservative agent, accepting a little before the optimal time. For the bidding strategy, we evaluated the use of Normal, Cauchy, and beta distributions for continuous control. The Cauchy has the highest reward, but the Normal is more consistent. The neural agent learns ways to evaluate the opponent, such as maintaining high mean, high initial variance to ensure enough rejections, before lowering the variance to more deterministic outcomes. This also demonstrates adaptability to concession and discounting.

Time-based experiments reveal the **barriers** to optimal convergence. We discover the error in stoppage time can be explained by the change in marginal utility (second derivative) and cliff-walking: the agent waits for higher rewards, then is punished aggressively due to enacting the conflict deal. The primary factors that influence the bidding optimality is trade-offs in variance (i.e.the beta distribution suffers from slow convergence due to low variance sensitivity). High variance is required to seek out optimal strategies, but low variance helps avoid the conflict deal. The peakiness of the Cauchy and its heavy tails makes it a suitable candidate.

The neural agent was also shown to be **cooperative** and **adaptive**. When playing against time-based agents with preference-based concessions, offers are accepted along the Pareto Frontier and produce the highest ex-

pected reward. Self-play in the centipede game shows agents are willing to accrue interest, which demonstrates cooperation over rationality. Against simple Bayesian TFT agents, the neural agent learns to quickly arrive at the Nash Solution, resulting in win-win cooperation. Since all results arise from a single neural architecture, the neural agent shows significant adaptability. Most importantly, the neural agent forces fairer results by either 1) utilizing the conflict deal or 2) levying discounting to force fairer offers. There is a strong similarity between non-credible threats in our neural agent and the rejection of low offers (using reputation) in evolutionary strategies. It's important to note evolutionary methods and RL are often framed as competing choices for agent design [67]. Beyond theoretical interest in diverging from classical game theory, these results may guide the design of fairer negotiations, with EGT from a population perspective and DRL from the individual agent's perspective.

Before discussing future work, I'll note what didn't work. Initially, the use of LSTMs seemed promising due to its success in natural language negotiation generation. However, there the action domain is discrete and limited.

5.1.1 Evaluation and Future Work

One weakness of this study is it studies a specific preference ordering. For the **scenario**, promising avenues include variations in the utility functions, as there are six combinations of preference orderings for three issues. More importantly is the inclusion of more complicated behavior-based agents. One barrier to this is, unlike the iterated prisoner dilemma that has hundreds of established strategies, we lack a repository that collects these strategies, such as the Axelrod library for the IPD [1].

However, this is quickly changing. An annual negotiation competition that began in 2010 [8] collects strong bots into the Genius Environment [42], maintained by Tim Baarslag. I anticipate running the neural agent against these bots, to understand how DRL performs in a tournament setting and against more complicated strategies.

Another weakness of this study is experimentation with design choices (**learning methodology**), although this was not possible given the focus of this dissertation was behavioral analysis. A separate study with a deep learning focus could scope-out the impact of neural architecture (the type of non-linearity and number of layers) and hyper-parameters (learning rate, reward discounting and the use of schedulers). Additionally, increasing the complexity of the algorithm may improve performance, such as increasing the input space to include n-prior moves against trajectory-based opponents, or the use of Monte-Carlo Tree Search and rollout [71].

Bibliography

[1] The Axelrod project developers . Axelrod: <release title>, April 2016.

[2] Manish K Agrawal and Kaushal Chari. Learning negotiation support systems in competitive negotiations: A study of negotiation behaviors and system impacts. *International Journal of Intelligent Information Technologies*, 5(1):1–23, 2009.

[3] George A Akerlof. The market for "lemons": Quality uncertainty and the market mechanism. In *Uncertainty in economics*, pages 235–251. Elsevier, 1978.

[4] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

[5] Alexios Arvanitis and Antonis Karampatzos. Negotiation and aristotle's rhetoric: Truth over interests? *Philosophical Psychology*, 24(6):845–860, 2011.

[6] Tim Baarslag, Mark JC Hendrikx, Koen V Hindriks, and Catholijn M Jonker. Learning about the opponent in automated bilateral negotiation: a comprehensive survey of opponent modeling techniques. *Autonomous Agents and Multi-Agent Systems*, 30(5):849–898, 2016.

[7] Tim Baarslag, Koen Hindriks, and Catholijn Jonker. A tit for tat negotiation strategy for real-time bilateral negotiations. In *Complex Automated Negotiations: Theories, Models, and Software Competitions*, pages 229–233. Springer, 2013.

[8] Tim Baarslag, Koen Hindriks, Catholijn Jonker, Sarit Kraus, and Raz Lin. The first automated negotiating agents competition (anac 2010). In *New Trends in agent-based complex automated negotiations*, pages 113–135. Springer, 2012.

[9] Tibor Bosse and Catholijn M Jonker. Human vs. computer behavior in multi-issue negotiation. In *Rational, Robust, and Secure Negotiation Mechanisms in Multi-Agent Systems (RRS'05)*, pages 11–24. IEEE, 2005.

[10] Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

[11] Réal Carbonneau, Gregory E Kersten, and Rustam Vahidov. Predicting opponent's moves in electronic negotiations using neural networks. *Expert Systems with Applications*, 34(2):1266–1273, 2008.

[12] Réal A Carbonneau, Gregory E Kersten, and Rustam M Vahidov. Pairwise issue modeling for negotiation counteroffer prediction using neural networks. *Decision Support Systems*, 50(2):449–459, 2011.

[13] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[14] Robert M Coehoorn and Nicholas R Jennings. Learning on opponent's preferences to make effective multi-issue negotiation trade-offs. In *Proceedings of the 6th international conference on Electronic commerce*, pages 59–68. ACM, 2004.

[15] Heriberto Cuayáhuitl, Simon Keizer, and Oliver Lemon. Strategic dialogue management via deep reinforcement learning. *arXiv preprint arXiv:1511.08099*, 2015.

[16] George B Dantzig, Alex Orden, Philip Wolfe, et al. The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5(2):183–195, 1955.

[17] Carsten KW De Dreu, Bianca Beersma, Wolfgang Steinel, and Gerben A Van Kleef. The psychology of negotiation: Principles and basic processes. In A. W. Kruglanski and E. T. Higgins, editors, *Social psychology: Handbook of basic principles*. The Guilford Press, New York, NY, US, 2007.

[18] Ren-Jye Dzeng and Yu-Chun Lin. Searching for better negotiation agreement based on genetic algorithm. *Computer-Aided Civil and Infrastructure Engineering*, 20(4):280–293, 2005.

[19] M v Eeuwen. Mobile conversational commerce: messenger chatbots as the next interface between businesses and consumers. Master's thesis, University of Twente, 2017.

[20] Konrad Ehlich and Johannes Wagner. *The discourse of business negotiation*, volume 8. Walter de Gruyter, 2011.

[21] Fang Fang, Ye Xin, Xia Yun, and Xu Haitao. An opponent's negotiation behavior model to facilitate buyer-seller negotiations in supply

chain management. In *2008 International Symposium on Electronic Commerce and Security*, pages 582–587. IEEE, 2008.

[22] George M Farag, Samir El-Sayed AbdelRahman, Reem Bahgat, and Atef M A-Moneim. Towards kde mining approach for multi-agent negotiation. In *2010 The 7th International Conference on Informatics and Systems (INFOS)*, pages 1–7. IEEE, 2010.

[23] Shaheen Fatima, Sarit Kraus, and Michael Wooldridge. *Principles of automated negotiation*. Cambridge University Press, 2014.

[24] Shaheen Fatima and Iyad Rahwan. Negotiation and bargaining. *Multiagent Systems*, 143, 2013.

[25] Kallirroi Georgila and David Traum. Reinforcement learning of argumentation dialogue policies in negotiation. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[26] Enrico H Gerding, David Dominique Bernard van Bragt, and Johannes Antonius La Poutré. *Scientific approaches and techniques for negotiation: a game theoretic and artificial intelligence perspective*. Centrum voor Wiskunde en Informatica, 2000.

[27] Jeonghwan Gwak and Kwang Mong Sim. Bayesian learning based negotiation agents for supporting negotiation with incomplete information. In *World Congress on Engineering 2012. July 4-6, 2012. London, UK.*, volume 2188, pages 163–168. International Association of Engineers, 2010.

[28] Valeriia Haberland, Simon Miles, and Michael Luck. Adaptive negotiation for resource intensive tasks in grids. In *Stairs*, pages 125–136, 2012.

[29] Chongming Hou. Predicting agents tactics in automated negotiation. In *Proceedings. IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004.(IAT 2004).*, pages 127–133. IEEE, 2004.

[30] Chun-Che Huang, Wen-Yau Liang, Yu-Hsin Lai, and Yin-Chen Lin. The agent-based negotiation process for b2c e-commerce. *Expert Systems with Applications*, 37(1):348–359, 2010.

[31] Takayuki Ito, Mark Klein, and Hiromitsu Hattori. A multi-issue negotiation protocol among agents with nonlinear utility functions. *Multiagent and Grid Systems*, 4(1):67–83, 2008.

[32] Hamid Jazayeriy, Masrah Azmi-Murad, Md Nasir Sulaiman, and Nur Izura Udzir. A review on soft computing techniques in automated negotiation. *Scientific Research and Essays*, 6(24):5100–5106, 2011.

[33] Hamid Jazayeriy, Masrah Azmi-Murad, Nasir Sulaiman, and Nur Izura Udizir. The learning of an opponent's approximate preferences in bilateral automated negotiation. *Journal of theoretical and applied electronic commerce research*, 6(3):65–84, 2011.

[34] Nicholas R Jennings, Peyman Faratin, Alessio R Lomuscio, Simon Parsons, Michael J Wooldridge, and Carles Sierra. Automated negotiation: prospects, methods and challenges. *Group Decision and Negotiation*, 10(2):199–215, 2001.

[35] Sarit Kraus. Negotiation and cooperation in multi-agent environments. *Artificial intelligence*, 94(1-2):79–97, 1997.

[36] Alex Krizhevsky and Geoff Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7):1–9, 2010.

[37] Chun Ching Lee and Chao Ou-Yang. A neural networks approach for forecasting the supplier's bid prices in supplier selection negotiation process. *Expert Systems with Applications*, 36(2):2961–2970, 2009.

[38] Roy J Lewicki, Bruce Barry, and David M Saunders. *Essentials of negotiation*. McGraw-Hill Education, 2016.

[39] Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra. Deal or no deal? end-to-end learning for negotiation dialogues. *arXiv preprint arXiv:1706.05125*, 2017.

[40] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[41] Raz Lin and Sarit Kraus. Can automated agents proficiently negotiate with humans? *Communications of the ACM*, 53(1):78–88, 2010.

[42] Raz Lin, Sarit Kraus, Tim Baarslag, Dmytro Tykhonov, Koen Hindriks, and Catholijn M Jonker. Genius: An integrated environment for supporting the design of generic automated negotiators. *Computational Intelligence*, 30(1):48–70, 2014.

[43] Yossi Maaravi, Yoav Ganzach, and Asya Pazy. Negotiation as a form of persuasion: Arguments in first offers. *Journal of personality and social psychology*, 101(2):245, 2011.

[44] Marisa Masvoula, Constantine Halatsis, and Drakoulis Martakos. Predictive automated negotiators employing risk-seeking and risk-averse strategies. In *Engineering Applications of Neural Networks*, pages 325–334. Springer, 2011.

[45] Marisa Masvoula, George Kontolemakis, Panagiotis Kanellis, and Drakoulis Martakos. Design and development of an anthropocentric negotiation model. In *Seventh IEEE International Conference on E-Commerce Technology (CEC'05)*, pages 383–386. IEEE, 2005.

[46] Stan Matwin, Tomasz Szapiro, and Karen Haigh. Genetic algorithms approach to a negotiation support system. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(1):102–114, 1991.

[47] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[48] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[49] Dirk C Moosmayer, Alain Yee-Loong Chong, Martin J Liu, and Bjoern Schuppar. A neural network approach to predicting price negotiation outcomes in business-to-business contexts. *Expert Systems with Applications*, 40(8):3028–3035, 2013.

[50] Graham Neubig. Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint arXiv:1703.01619*, 2017.

[51] Christoph Niemann and Florian Lang. Assess your opponent: A bayesian process for preference observation in multi-attribute negotiations. In *Advances in agent-based complex automated negotiations*, pages 119–137. Springer, 2009.

[52] Martin A Nowak, Karen M Page, and Karl Sigmund. Fairness versus reason in the ultimatum game. *Science*, 289(5485):1773–1775, 2000.

[53] Mihaela Oprea. An adaptive negotiation model for agent-based electronic commerce. *Studies in informatics and Control*, 11(3):271–279, 2002.

[54] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994.

[55] Yinon Oshrat, Raz Lin, and Sarit Kraus. Facing the challenge of human-agent negotiations via effective general opponent modeling. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 377–384. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[56] Ioannis Papaioannou, Ioanna Roussaki, and Miltiades Anagnostou. Multi-modal opponent behaviour prognosis in e-negotiations. In *International Work-Conference on Artificial Neural Networks*, pages 113–123. Springer, 2011.

[57] Ioannis V Papaioannou, Ioanna G Roussaki, and Miltiades E Anagnostou. Neural networks against genetic algorithms for negotiating agent behaviour prediction. *Web Intelligence and Agent Systems: An International Journal*, 6(2):217–233, 2008.

[58] Howard Raiffa. *The art and science of negotiation*. Harvard University Press, 1982.

[59] David G Rand, Corina E Tarnita, Hisashi Ohtsuki, and Martin A Nowak. Evolution of fairness in the one-shot anonymous ultimatum game. *Proceedings of the National Academy of Sciences*, 110(7):2581–2586, 2013.

[60] Amnon Rapoport, Terry E Daniel, and Darryl A Seale. Reinforcement-based adaptive learning in asymmetric two-person bargaining with incomplete information. *Experimental Economics*, 1(3):221–253, 1998.

[61] Hsin Rau, Mou-Hsing Tsai, Chao-Wen Chen, and Wei-Jung Shiang. Learning-based automated negotiation between shipper and forwarder. *Computers & industrial engineering*, 51(3):464–481, 2006.

[62] Zhaomin Ren and Chimay J Anumba. Learning in multi-agent systems: a case study of construction claims negotiation. *Advanced engineering informatics*, 16(4):265–275, 2002.

[63] Jeffrey S Rosenschein and Gilad Zlotkin. *Rules of encounter: designing conventions for automated negotiation among computers*. MIT press, 1994.

[64] Jeffrey Z Rubin and Bert R Brown. *The social psychology of bargaining and negotiation*. Elsevier, 2013.

[65] Ariel Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica: Journal of the Econometric Society*, pages 97–109, 1982.

[66] Sabyasachi Saha, Anish Biswas, and Sandip Sen. Modeling opponent decision in repeated one-shot negotiations. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 397–403. ACM, 2005.

[67] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[68] Tuomas Sandholm, Nir Vulkan, et al. Bargaining with deadlines. In *AAAI/IAAI*, pages 44–51, 1999.

[69] Sandip Sen, Mahendra Sekaran, John Hale, et al. Learning to coordinate without sharing information. In *AAAI*, volume 94, pages 426–431, 1994.

[70] Yoav Shoham, Rob Powers, and Trond Grenager. Multi-agent reinforcement learning: a critical survey. Technical report, Technical report, Stanford University, 2003.

[71] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[72] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[73] Kwang Mong Sim, Yuanyuan Guo, and Benyun Shi. Adaptive bargaining agents that negotiate optimally and rapidly. In *2007 IEEE Congress on Evolutionary Computation*, pages 1007–1014. IEEE, 2007.

[74] Kwang Mong Sim, Yuanyuan Guo, and Benyun Shi. Blgan: Bayesian learning and genetic algorithm for supporting negotiation with incomplete information. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(1):198–211, 2008.

[75] Reid G Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on computers*, (12):1104–1113, 1980.

[76] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[77] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[78] Katia Sycara, Daniel Zeng, et al. Benefits of learning in negotiation. In *Proceedings of the AAAI National Conference on Artificial Intelligence. Menlo Park, California*, pages 36–41, 1997.

[79] Rustam Vahidov, Gregory Kersten, and Raafat Saade. An experimental study of software agent negotiations with humans. *Decision Support Systems*, 66:135–145, 2014.

[80] Bruno Verbeek. Game theory and ethics. In *Stanford Encyclopedia of Philosophy*, pages 1–19. Stanford University Press, 2008.

[81] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior (commemorative edition)*. Princeton university press, 2007.

[82] William E Walsh and Michael P Wellman. Modeling supply chain formation in multiagent systems. In *International Workshop on Agent-Mediated Electronic Commerce*, pages 94–101. Springer, 1999.

[83] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[84] RJ Willianms. Toward a theory of reinforcement-learning connectionist systems. *Technical Report NU-CCS-88-3, Northeastern University*, 1988.

[85] Kelvin Kian Loong Wong. Bridging game theory and the knapsack problem: a theoretical formulation. *Journal of Engineering Mathematics*, 91(1):177–192, 2015.

[86] Chao Yu, Fenghui Ren, and Minjie Zhang. An adaptive bilateral negotiation model based on bayesian learning. In *Complex automated negotiations: Theories, models, and software competitions*, pages 75–93. Springer, 2013.

[87] Dajun Zeng and Katia Sycara. Bayesian learning in negotiation. *International Journal of Human-Computer Studies*, 48(1):125–141, 1998.

Appendix A

Appendix

A.1 Policy-Gradient Theorem

The policy gradient theorem states the change in scalar is proportional to change in policy weights. More specifically, this is given as:

$$\nabla J(\theta) \propto \Sigma_s \mu(s) \Sigma q_\pi(s,a) \nabla \pi(a|s,\theta)$$

Here, $\mu(s)$ denotes the on-policy distribution of a state under policy π. We can think of this as the frequency a state has occurred. $q_\pi(s,a)$ is the value of the state-action pair and $\nabla\pi(a|s,\theta)$ is the change in the policy distribution. What this says is a positive change in $J(\theta)$ can be produced a proportional shift in the policy. A full proof can be found in Chapter 13 of [76].

A.2 Point to Line Calculation

The general form of the closest distance from a point to line is

$$d(ax + by + c = 0, (x_0, y_0)) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}. \tag{A.1}$$

The Pareto Frontier s given by Eq. 4.8. Hence, the distance of a point $P = (x_p, y_p)$ to the Pareto Frontier is:

$$\min \left\{ \frac{|x_p + \frac{1}{3}y_p - 6|}{\sqrt{\frac{10}{9}}}, \frac{|x_p + y_p - 8|}{\sqrt{2}}, \frac{|x_p + 3y_p - 18|}{\sqrt{10}} \right\}$$

A.3 Actor-Critic Playout Implementation

for *e in epochs* **do**
  while *Not accepted and t <deadline* **do**
    P1 offers;
    if *P2 Accepts* **then**
      Collect States for both players ;
      Collect Acceptance Actions and Rewards for both players;
    else
      P2 Offers;
      Swap Places (P1 receives then counter offers);
      $t+ = 1$;
    end
  end
  Calculate Critic and Actor Loss for both Accept Net and Offer Net;
  Backprop on both networks;
end

Algorithm 3: Implementation of negotiation playout and training pipeline

A.4 Change in Univariate Mean Estimation

The y-axis shows the decision utility at a given time, which is inversely related to the concession factor. The higher the decision utility, the more Boulware the agent is. The concession value can be converted with $c = \frac{\ln \frac{t}{T}}{\ln U_d}$.

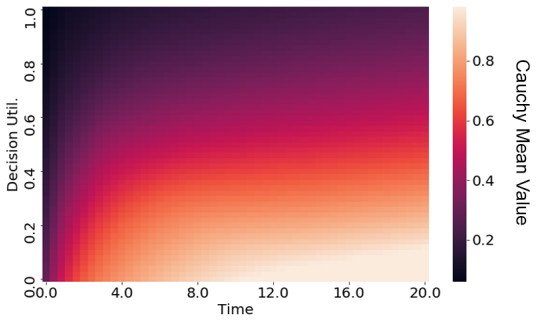

Figure A.1: Cauchy means based on opponent decision utility and time. The decision utility serves as a proxy for concession, as the higher the decision utility, the more Boulware the opponent.