Automatic Transfer Between Negotiation Tasks

Siqi Chen, Haitham Bou Ammar, Kurt Driessens, Karl Tuyls, Gerhard Weiss
Department of Knowledge Engineering
P.O. Box 616, 6200 MD Maastricht
Maastricht, The Netherland
{first.last}@maastrichtuniversity.nl

ABSTRACT

Learning in automated negotiations, while useful, is hard because of the indirect way the target function can be observed and the limited amount of experience available to learn from. Transfer learning is a branch of machine learning research concerned with the reuse of previously acquired knowledge in new learning tasks to, for example, reduce the amount of learning experience required to attain a certain level of performance. This paper proposes two new variations of TrAdaBoost — a well known instance transfer technique — that can be used in a multi-issue negotiation setting. Theoretical bounds on the performance increase caused by this transfer scheme are derived and the suggested approach is evaluated in a couple of negotiation scenarios.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distribute Artificial intelligence, intelligence agents, multi-agents systems

General Terms

Algorithms, Performance, Economics, Experimentation

Keywords

Automated Multi-issue Negotiation, Transfer Learning, Gaussian Processes

1. INTRODUCTION

In automated negotiation two (or more) agents try to come to a joint agreement in a consumer-provider or buyer-seller set-up. One of the biggest driving forces behind research into automated negotiation is the broad spectrum of potential applications. For example, Dang et al. in [7] use automated negotiation for service allocation, Ponka et al. [15] propose the adoption of automated negotiation in electronic commerce and Lau et al. [12] make use of the framework to aid in electronic markets. Other applications of automated negotiation include their deployment to information markets [19], in business process management [11], etc.

Various negotiation settings differentiate themselves through e.g. the number of participants, i.e. bilateral or multilateral, or the number of issues being negotiated upon, i.e. whether a single or multiple issues are the subject of each placed bid. While the contributions from this paper are (at least in principle) applicable to multilateral negotiations, the paper focusses on bilateral, multi-issue negotiation mainly because this makes the proposed techniques easier to explain. The interaction framework enforced in automated negotiations (see Section 2.1) lends itself to the use of machine learning techniques for opponent modeling. The driving force of an (opposing) agent is governed by its hidden utility function through its also hidden bidding strategy. Given the utility function (and bidding strategy)[1] of an opposing agent, an agent can reach better final (or cumulative) agreement terms by exploiting this knowledge. However, learning an opposing agent's utility function is hard because (i) the utility function can only be observed indirectly through offers refused and counter offers made by the opposing agent, (ii) learning needs to be very fast as the number of these interactions in a single negotiation is limited.

Transfer learning (Section 2.3) is a branch of machine learning that enables the use of knowledge learned in a so called *source* task to aid learning in a different, related *target* task. One of the main goals of transfer is to accelerate learning, i.e. to reach the same or better performance with less learning examples, which makes it ideally suited for learning settings like that of automated negotiation. This paper proposes a novel principled learning scheme which applies TrAdaBoost to automated negotiation. TrAdaBoost is an example transfer technique that reuses and weights learning examples from the source task according to their usefulness for the target task. The contributions of the paper also include an empirical illustration of the technique on a couple of automated negotiations and the derivation of theoretical bounds on the performance improvement resulting from transfer that opens up a few new directions of research.

The remainder of this paper is structured as follows. Section 2 overviews important related work. Section 3 underlines the background in the our research. Section 4 propose the novel learning scheme. Section 5 presents the mathematical derivations of the error bounds for each of the update rules. Section 6 offers the experimental analysis of the proposed learning approach. Section 7 discusses on the applicability of the proposed method. Section 8 identifies some important research lines induced by the described work and concludes the paper.

2. PRELIMINARIES

This section provides the reader with the necessary background knowledge for the remainder of this paper. In order, it sketches the framework of bilateral negotiations, presents the needed preliminaries and notations for Gaussian Processes as a regression technique and discusses the instance transfer technique of TrAdaBoost.

2.1 Bilateral Negotiation

As stated before, the automated negotiation framework adopted in this paper is a basic bilateral multi-issue negotiation model as it is

[1]Because both an agents utility function and bidding strategy are hidden, we will often use the term utility function to encompass both as the force governing the agents bidding behavior.

widely used in the agents field (e.g., [5]). The negotiation protocol is based on a variant of the alternating offers protocol proposed in [16].

Let $I = \{a, b\}$ be a pair of negotiating agents, where i ($i \in I$) is used to represent any of the two agents. The goal of a and b is to establish a contract for a product or service, where a contract consists of a vector of values, each assigned to a particular issue such as price, quality or delivery time.

Inherent to the negotiation process is that agents a and b act in conflictive roles. To make this precise, let J be the set of issues under negotiation where j ($j \in \{1, ..., n\}$) is used to represent a particular issue. Contracts are tuples $O = (O_1, \ldots, O_n)$ that assign a value O_j to each issue j. A contract is said to be established if both agents agree on it. Following Rubinstein's alternating bargaining model [18], each agent makes, in turn, an offer in form of a contract proposal.

An agent receiving an offer at time t needs to decide whether (i) to accept or (ii) to reject and propose a counter-offer at time $t + 1$. Counter-offers can be made until one agent's deadline t_{max} is reached and it has to withdraw from the negotiation. Negotiation continues until one of the negotiating agents accepts or withdraws due to timeout.

Each agent i decides to accept or reject a contract based on a weight vector $w^i = (w^i_1, \ldots, w^i_n)$ (also called importance vector or preference vector) that represents the relative importance of each issue $j \in \{1, ..., n\}$. These weights are usually normalized (i.e., $\sum_{j=1}^{n}(w^i_j) = 1$ for each agent i).

The utility of an offer for agent i is obtained by the utility function, defined as:

$$U^i(O) = \sum_{j=1}^{n}(w^i_j \cdot V^i_j(O_j)) \tag{1}$$

where w^i_j and O are as defined above and V^i_j is the evaluation function for i, mapping every possible value of issue j (i.e., O_j) to a real number.

After receiving an offer from the opponent, O_{opp} at time t, an agent decides on acceptance or rejection according to its interpretation $I(t, O_{opp})$ of the current negotiation situation. For instance, this decision can be made depending on a certain threshold or can be based on utility differences. Agents usually have a lowest expectation for the outcome of a negotiation below which the agent will never accept an offer; this expectation is called reserved utility u_{res}. If the agents know each other's utility functions, they can compute the Pareto-optimal contract [16]. However, in general, a negotiator will not make this information available to its opponent.

2.2 Gaussian Processes

Gaussian Processes (GPs) are a form of non-parametric regression techniques. Following the notation of [17], given a data set $\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{m}$ where $\mathbf{x} \in \mathbb{R}^d$ is the input vector, $y \in \mathbb{R}$ the output vector and m is the number of available data points when a function is sampled according to a GP, we write, $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, where $m(\mathbf{x})$ is the mean function and $k(\mathbf{x}, \mathbf{x}')$ the covariance function, together fully specifying the GP. Learning in a GP setting involves maximizing the marginal likelihood of Equation 2.

$$\log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}^T\left(\mathbf{K} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K} + \sigma_n^2\mathbf{I}| - \frac{n}{2}\log 2\pi, \tag{2}$$

where $\mathbf{y} \in \mathbb{R}^{m\times 1}$ is the vector of all collected outputs, $\mathbf{X} \in \mathbb{R}^{m\times d}$ is the matrix of the data set inputs, and $\mathbf{K} \in \mathbb{R}^{m\times m}$ is the covariance matrix with $|.|$ representing the determinant. The interested reader should refer to [17] for a more thorough discussion of the topic. To fit the hyperparameters that best suit the available data set we need to maximize the marginal likelihood function of Equation 2 with respect to $\mathbf{\Theta}$ the vector of all hyperparameters. Typically, this maximization requires the computation of the derivatives of Equation 2 with respect to $\mathbf{\Theta}$. These derivatives are then used in a gradient-based algorithm to perform the updates to the hyperparameters θ_j using the following set of equations:

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\log p(\mathbf{y}|\mathbf{X}, \mathbf{\Theta}) &= \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\frac{\partial\mathbf{K}}{\partial\theta_j}\mathbf{K}^{-1}\mathbf{y} - \frac{1}{2}\text{tr}\left(\mathbf{K}^{-1}\frac{\partial\mathbf{K}}{\partial\theta_j}\right)\\ &= \frac{1}{2}\text{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^T - \mathbf{K}^{-1})\frac{\partial\mathbf{K}}{\partial\theta_j}\right) \text{ with } \boldsymbol{\alpha} = \mathbf{K}^{-1}\mathbf{y}\end{aligned} \tag{3}$$

2.3 Transfer Learning

Transfer Learning (TL) is a technique used to counter the tabula rasa approach used often in machine learning. The idea of TL is to reuse knowledge obtained through previous learning experiences when dealing with a new learning task. In its most basic form, there exists a source and a target task where learning in the source task was accomplished at some earlier time and the knowledge acquired (in whatever form) is available for use in the target task. The main assumption made in any TL scheme [6, 14, 20] is that the source and target task can be found similar in some sense. Finding out wether a given set of tasks is similar or not is still an open question in TL and while it is out of the scope of this paper to actually answer this question, we provide some intuition and possible ideas in Section 7 that could be useful in the specific task of determining if two negotiation tasks are similar or not.

TL can be used to bias learning in a new learning task with the intention to reduce learning time, or more generally, the amount of experience needed to learn an equally successful model. This makes TL ideally suited to strengthen any learning done within a restricted scenario like those of automated negotiation, where the amount of experience to learn from is limited. In this paper, previously encountered offers are used to bias the learning agent's offer proposition scheme. More precisely, source task samples are transferred to the target task to aid in learning the opponent's utility model in a bilateral multi-issue negotiation task. This is a supervised learning problem for which we resort to a well understood and widely used transfer learning algorithm called *TrAdaBoost* and detailed next.

2.3.1 Boosting for Transfer Learning

TrAdaBoost [6] is an algorithm used to transfer learning instances between a source and a target task. The idea is to use source task samples in the target task to increase the amount of learning data available. The transferred examples are weighted so that they get a low weight if they hurt performance in the target task and a high weight if they increase performance in the target task. TrAdaBoost was proposed for classification tasks. First follows a description the original TrAdaBoost algorithm and later, in Section 4.3, two variants of TrAdaBoost for negotiation tasks are proposed.

TrAdaBoost is formalized in terms of the following: $\mathcal{X}_s$ is the example space in which a new learning task needs to be solved, i.e. the example space of the target task, $\mathcal{X}_D$ is the example space from the source task[2] and $\mathcal{Y} = \{0, 1\}$ is the set of labels[3]. A concept c

[2]Adopting the same notation as in the original TrAdaBoost paper, the index s stands for "*same* distribution instance space" and the index D for "*different* distribution instance space"

[3]Extending TrAdaBoost to multi-class classification problems is

Algorithm 1 TrAdaBoost Framework

1: **Require:** two labeled data sets $\mathcal{T}_D$ and $\mathcal{T}_s$, the unlabeled data set $\mathcal{S}$, a base learning algorithm Ξ, and the maximum number of iterations N.
2: **Initialize:** the weight vector $\mathbf{w}^{(1)} = [w_1^1, \ldots, w_{n+m}^{(1)}]^T$
3: **for** $t = 1$ to N **do**
4: Set $\mathbf{p}^{(t)} = \mathbf{w}^{(t)} / \left(\sum_{i=1}^{n+m} w_i^{(t)}\right)$
5: Learn a hypothesis $h^{(t)} : \mathcal{X} \to \mathcal{Y}$ by calling Ξ and passing the distribution $\mathbf{p}^{(t)}$ over the combined data set $\mathcal{T}$.
6: Compute the prediction error of $h^{(t)}$ on $\mathcal{T}_s$ using:

$$\epsilon^{(t)} = \sum_{i=n+1}^{n+m} \frac{w_i^{(t)} |h^{(t)}(x^{(i)}) - c(x^{(i)})|}{\sum_{i=n+1}^{n+m} w_i^{(t)}}$$

7: **end for**
8: Set $\beta^{(t)} = \frac{\epsilon^{(t)}}{1-\epsilon^{(t)}}$ and $\beta = \frac{1}{1+\sqrt{2\ln(n/N)}}$
9: Update the weight vector according to:

$$w_i^{(t+1)} = \begin{cases} w_i^{(t)} \beta^{|h^{(t)}(x^{(i)}) - c(x^{(i)})|} & \text{for } i = 1, \ldots, n \\ w_i^{(t)} \beta^{-|h^{(t)}(x^{(i)}) - c(x^{(i)})|} & \text{for } i = n+1, \ldots, n+m \end{cases} \quad (5)$$

10: **Output:**

$$h^{(f)}(x) = \begin{cases} 1 & \text{if } \prod_{t=\frac{N}{2}}^{N} \beta_t^{-h^{(t)}(x)} \geq \prod_{t=\frac{N}{2}}^{N} \beta_t^{-\frac{1}{2}} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

is a function mapping from $\mathcal{X} \to \mathcal{Y}$ with $\mathcal{X} = \mathcal{X}_s \cup \mathcal{X}_d$. The test set is denoted by $\mathcal{S} = \{x_t^{(i)}\}_{i=1}^k \in \mathcal{X}_s$. The training data set $\mathcal{T} \subseteq \{\mathcal{X} \times \mathcal{Y}\}$ is partitioned into two labeled data sets $\mathcal{T}_D$ and $\mathcal{T}_s$. $\mathcal{T}_D$ represents the source task data set with $\mathcal{T}_D = \{(x_D^{(i)}, c(x_D^{(i)}))\}_{i=1}^n$ where $x_D^{(i)} \in \mathcal{X}_D$ and $\mathcal{T}_s$ represents the target task data set where $\mathcal{T}_s = \{(x_s^{(j)}, c(x_s^{(j)}))\}_{j=1}^m$. The superscripts n and m are the sizes of the two data sets $\mathcal{T}_D$ and $\mathcal{T}_s$. Therefore, the combined training set $\mathcal{T} = \{(x^{(l), c(x^{(l)})})\}_{l=1}^{n+m}$ now consists of:

$$x^{(l)} = \begin{cases} x_d^{(i)} & \text{for } l = 1, \ldots, n \\ x_s^{(i)} & \text{for } l = n+1, \ldots, n+m \end{cases} \quad (4)$$

Using these definitions the problem that TrAdaBoost tries to solve is: given a small number of of labeled target task training data $\mathcal{T}_s$ and a large number of source task instances $\mathcal{T}_D$ and an unlabeled data set $\mathcal{S}$, learn a classifier $\hat{c} : \mathcal{X} \to \mathcal{Y}$ that minimizes the prediction error on the unlabeled data set $\mathcal{S}$.

The main idea behind TrAdaBoost is described in Algorithm 1. The algorithm starts by initializing, for example randomly, the instance weighting vector (line 2). In each iteration, the weight vector is translated into a probability distribution over all learning examples. This distribution of the learning examples is used by a learner Ξ to produce a concept hypothesis $h^{(t)}$. Line 6 computes the weighted error of $h^{(t)}$ on the target task data which in turn is used in lines 8-9 to update the weighting vector. The process is repeated until the weight vector stays at a constant value and the convergence is achieved or when the maximum number of iterations is reached. Finally, the hypothesis $h^{(f)}(x)$ is returned. It is worth noting that TrAdaBoost does not make any assumptions on the type of learner Ξ to be used. In theory any learner capable performing classification with an error rate $\epsilon < 0.5$ could be applied.

fairly straight forward

3. RELATED WORK

Opponent modeling is assumed to be of key importance to performing well in automated negotiations [18]. Learning in the negotiation setting is however hampered by the limited information that can be gained about an opponent during a single negotiation session. To enable learning in this setting, simplifying assumptions are often made. For example, Zeng et al. [22] proposed a Bayesian learning approach to opponent modeling in single-issue and one-shot negotiation, which enables an agent to learn the reserved utility of its opponent. Also founded in Bayesian learning, Lin et al. [13] introduce a reasoning model based on a decision making and beliefs updating mechanism which allows the identification of the opponent profile from a set of publicly available profiles. Brzostowski et al. [2] investigate online prediction of future counter-offers by using differentials, thereby assuming that the opponent strategy is defined using a combination of time- and behavior-dependent tactics[4]. Hou [10] employed non-linear regression to learn the opponent's decision function in a single-issue negotiation setting with the assumption that the opponent uses a tactic from the three tactic families introduced in [8]. Carbonneau et al. [3] use a three-layer artificial neural network to predict future counter-offers in a specific domain, but the training process requires a large amount of previous encounters.

Only recently work has started to focus on learning opponent's strategy in complex negotiation without simplifying assumption. William et al. [21] applied Gaussian processes to predict the opponent's future concession. The resulting information can be used profitably by the agent to set the concession rate accordingly. In [4] Chen and Weiss proposed a negotiation approach called OMAC which learns the opponent's strategy to adjust its negotiation tactic through wavelets and cubic smoothing splines. Although these methods are very successful, the restriction caused by the negotiation setting, makes the learning process fairly slow.

4. INSTANCE TRANSFER FOR NEGOTIATION TASKS

This section details the proposed method for automatically transferring between two negotiation tasks within the same domain. The negotiation setting in which the algorithms operate is first described. Second, the details of the learning in each of the source and the target tasks are presented.

4.1 Transfer Learning Setting

In the proposed transfer learning setting, there exists a source and a target negotiation task. The source task agent $\mathcal{A}_1$ has already finished negotiating against a source opponent $\mathcal{O}_1$ and already approximated the latter's model using gaussian processes. Then, $\mathcal{A}_1$ is faced with a new opponent, $\mathcal{O}_2$, in a target negotiation task. The target opponent, $\mathcal{O}_2$, potentially differs from the one of the source task in its bidding strategy and/or utility function. Still, using the source instances straightforwardly in the target task might be beneficial, although this can be expected to be beneficial only in a small fraction of closely related negotiation tasks. In other words, as the behavioral differences between the source and target opponents grow, the transferred instances will be of less use for modeling the target opponent. In the worst case, it might even diminish the target agent's performance. To counter this, the source instances are weighted in the target task. Determining the weights is done iteratively according to the TrAdaBoost algorithm, which was detailed in Section 4.3.

[4]The concepts of time-dependent and behavior-dependent tactics were introduced in [8].

During the negotiation, the agent collects data and iteratively builds the opponent's model. In the conventional case, the prediction of the opponent's next step utility is solely based on the approximated model. In this work, two variants of the latter are proposed. In the first, the agent predicts the utility based on the target's utility model combined with a weighted version of the source task opponent's model. This is reasonable, since if both the source and the target task opponent's are similar in their strategies and utilities, adding the weighted knowledge of the source task aids in predicting the target's opponent future decisions. In the second, the agent predicts the target opponent's utility based on weighting both the source and the target task's models. The motivation behind weighting the source task's model is the same as for the previous case. For the target task, weighting the instances can focus the attention of the learner to drastic changes in the behavior of the opponent as needed.

4.1.1 Transfer Setting Formalization

The source and target task knowledge are attained from different opponents. This means that either the utility models of the source and the target opponent and/or the strategies between them are different. We define the strategies set $\mathbf{\Pi} = \{\pi_1, \pi_2\}$, with π_1 and π_2 being the strategies of each of the source and the target opponent, respectively. The source task's opponent strategy π_1 is sampled from $\sim \chi_{\Pi}^{(1)}(.)$, while π_2 is sampled from $\sim \chi_{\Pi}^{(2)}(.)$ with $(.)$ being the suitable domain. We assume the strategies of the two opponents are similar to a certain tolerance $\epsilon^{\mathbf{\Pi}}$. In other words, the distance between the two strategy probability distributions, $\chi_{\Pi}^{(1)}$ and $\chi_{\Pi}^{(2)}$, $D_{KL}^{\Pi}(\chi_{\Pi}^{(1)}||\chi_{\Pi}^{(2)}) = \int_{-\infty}^{\infty} \chi_{\Pi}^{(1)}(\mathbf{x}) \ln \frac{\chi_{\Pi}^{(1)}(\mathbf{x})}{\chi_{\Pi}^{(2)}(\mathbf{x})} d\mathbf{x} \leq \epsilon^{\mathbf{\Pi}}$ with $\mathbf{x}$ being an instance of the valid probability distribution domain. Further, we define a utility set $\mathcal{U} = \{u_1, u_2\}$, where u_1 is the utility function of the source opponent and u_2 is the one of the target. The utility functions of each of the two opponents are also distributed according to their own probability density functions, $u_1 \sim \mathcal{U}_{\mathcal{U}}^{(1)}$, and $u_2 \sim \mathcal{U}_{\mathcal{U}}^{(2)}$. Similar to before, we assume, $D_{KL}^{\mathcal{U}}(\mathcal{U}_{\mathcal{U}}^{(1)}||\mathcal{U}_{\mathcal{U}}^{(1)}) = \int_{-\infty}^{\infty} \mathcal{U}_{\mathcal{U}}^{(1)} \ln \frac{\mathcal{U}_{\mathcal{U}}^{(1)}}{\mathcal{U}_{\mathcal{U}}^{(2)}} d\mathbf{y} \leq \epsilon^{\mathcal{U}}$, where $\epsilon^{\mathcal{U}}$ is an acceptable utility difference and $\mathbf{y}$ represents an instance of the valid domain.[5]. According to previous formalization, for transfer to be successful the difference between the two utility models of each of the source and the target opponent should be within a certain range $\epsilon^{\mathcal{U}}$.

4.2 Learning in the Source Task

The source negotiation task starts by the opponent agent presenting an offer describing values for the different negotiation issues. Utility is calculated according to the proposed opponent's offer, which is either accepted or rejected. If the offer is accepted the negotiation session ends. On the other hand, if the offer is rejected the agent proposes a counter-offer. Then, the opponent can decide, according to his own utility function, whether to accept or reject this counter-offer.

While the opponent's utility function is unknown, it can be learned over time. The opponent utility is indirectly observed from the utilities of the opponent's counter-offers: every time the opponent proposes a counter-offer, the utility of this offer is computed and added to the data set $\mathcal{D}_1 = \{t_1^{(i)}, u_1^{(i)}\}_{i=0}^{t_{1_{\max}}}$, with $t_1^{(i)}$ representing the source task time steps running to a maximum of $t_{1_{\max}}$. The data set grows dynamically as the negotiation session continues. Every time a new instance is obtained, the model — in this case a Gaussian process — is trained anew to discover a new latent function best describing the new data set[6]. The new model is then used to propose a new offer to the opponent. This is achieved through the prediction probability distribution of the trained Gaussian processes. Formally, the predicted utility at a new time step $t_*^{\star}$ is calculated according to the following:

$$u_1^{\star}|\mathbf{\Gamma}_1, \mathbf{u}_1, t_1^{\star} \sim \mathcal{N}(\bar{\mathbf{u}}_1^{\star}, \text{cov}(\mathbf{u}_1^{\star})) \tag{7}$$

with

$$\bar{\mathbf{u}}_1^{\star} = \mathbf{K}_1(t_1^{\star}, \mathbf{\Gamma}_1) \left[\mathbf{K}_1(\mathbf{\Gamma}_1, \mathbf{\Gamma}_1) + \sigma_1^2 \mathbf{I}\right]^{-1} \mathbf{u}_1$$

$$\text{cov}(\mathbf{u}_1^{\star}) = \mathbf{K}_1(t_1^{\star}, t_1^{\star}) - \mathbf{K}_1(t_1^{\star}, \mathbf{\Gamma}_1) \left[\mathbf{K}_1(\mathbf{\Gamma}_1, \mathbf{\Gamma}_1) + \sigma_1^2 \mathbf{I}\right]^{-1} \mathbf{K}_1(\mathbf{\Gamma}_1, t_1^{\star})$$

where $u_1^{\star}$ is the predicted value at $t_1^{\star}$, $\mathbf{\Gamma}_1$ is the matrix of all the history of inputs till $t_1^{\star}$, $\mathbf{u}_1$ presents the vector of utilities collected so far, $\mathbf{K}_1(t_1^{\star}, \mathbf{\Gamma}_1)$ is the covariance matrix formed between the new input and the history of all inputs, $\mathbf{K}_1(\mathbf{\Gamma}_1, \mathbf{\Gamma}_1)$ describes the covariance formed between all the history of inputs, and $\sigma_1^2\mathbf{I}$ signifies the noise in each of the problem's dimensions.

The negotiation session ends when either an agreement is reached or the available time steps are exhausted. Finally, the opponent's utility model described by the hyperparameters of the Gaussian process is returned for later use.

4.3 Knowledge Transfer and Target Task Learning

A modification of the TrAdaBoost algorithm of Section 2.3.1 is used to transfer knowledge from the source task. From the definitions of Section 4.1.1 the needed data sets for the proposed algorithm can be intuitively defined. The approach is specified in Algorithm 2 which requires two data sets as input. The first is $\mathcal{T}_D$ which represents the different distribution data set. Since the source and the target opponent's attain their utilities from different distributions, defined in Section 4.1.1, then, the different distribution data is that of the source. Namely, $\mathcal{T}_D = \{t_1^{(i)}, u_1^{p(i)}\}_{i=1}^{t_{1_{\max}}}$, where $u_1^{p(i)}$ is the predicted source task's utility determined according to Equation 7, and i is the time index running to $t_{1_{\max}}$. The second data set required by the transfer algorithm is the same distribution data set $\mathcal{T}_S$. The same distribution data set is the one from the target task having target time steps as inputs and target utilities as outputs. In the TrAdaBoost setting the number of points in the same distribution data set is allowed to be much less than that of the source.

To define $\mathcal{T}_S$ we propose one of the following two options. In the first, some instances of $\mathcal{T}_S$ are set manually. This is a valid option if the designer has some background knowledge about the negotiation task. In the other case, the instances of $\mathcal{T}_S$ are attained automatically from the initial negotiation steps in the target task, where the offer proposition step depends only on the source task's knowledge (i.e., u_1^p). In this case, the behavior of the opponent is monitored similar to the learning in the source task and the utilities of the counter-offers are computed and added to $\mathcal{T}_S$. Please note, that the number of samples n of the same distribution data set, $\mathcal{T}_S$, need not be large. In fact, it suffices for n to be much less than the number of instance in $\mathcal{T}_D$ for the algorithm to perform well.

Having the above data sets, the weights of each of the samples are fitted according to a modified version of TrAdaBoost as shown in line 4 of Algorithm 2. This function is detailed in the pseudocode of Algorithm 3. The algorithm follows the same steps as in

[5]Please note, that the formalization using the KL measure assesses that the two distributions should be having the same domain. This is reasonable in our framework as we operate within the same negotiation domain. If the two distributions are structurally different both could be approximated using one bigger distribution such as Gaussian mixture models.

[6]In this work we split the negotiation session in intervals of 3 sec.

the normal TrAdaBoost with the slight modification that it uses the Gaussian process and the normalization constant $Z^{(k)}$ to compute the normalized prediction error (line 5). $u_1^{(i)p}$ in Equation 10 represents the source model prediction determined according to Equation 7 with the source task covariance and input variables.

The outputs of this function are the learned model parameters as well as the two data distributions corresponding to $\mathcal{T}_D$ and $\mathcal{T}_S$. Once the TrAdaBoost algorithm fits the weights, the agent proposes an offer according to one of the equations (8 or 9) in line 5 of Algorithm 2. Here $u_2^{(k)p}$ is the predicted output of the target task function approximator. In case of a counter-offer, the utility of this offer is determined and added to $\mathcal{T}_S$ so to be used in the next run of TrAdaBoost. After the total time is exhausted the algorithm returns the target opponent utility model.

Algorithm 2 The overall framework of the transfer scheme is described. The idea is to re-weight instances from the source task such that it helps the target task agent in learning the target task's opponent utility model.

1: **Require:** different distribution (i.e., source task) labeled data sets $\mathcal{T}_D$, same distribution data set $\mathcal{T}_s$, a base learning algorithm $\mathcal{GP}_2$, the maximum number of iterations N, and maximum time allowed $t_{2_{\max}}$
2: Set n=size($\mathcal{T}_S$)
3: **while** $t_2^{(k)} < t_{2_{\max}}$ **do**
4: TrAdaBoost ($\mathcal{T}_D$, $\mathcal{T}_S$, N, n)
5: Propose new offer according to one of the following:

$$u_2^{(k)} = \mathbf{p}_1^{(k)} u_1^{(k)} + u_2^{(k)p} \tag{8}$$

$$u_2^{(k)} = \mathbf{p}_1^{(k)} u_1^{(k)} + \mathbf{p}_2^{(k)} u_2^{(k)p} \tag{9}$$

6: **if** Opponent Accept **then**
7: agreement reached
8: **else**
9: Opponent proposes counteroffer
10: Collect time and utility and add to $\mathcal{T}_S$
11: **end if**
12: Increment time index
13: **end while**
14: **Output:**
The hyperparameters of the target task model $\mathcal{GP}_2$ (i.e., $\mathbf{\Theta}_2$)

Algorithm 3 TrAdaBoost ($\mathcal{T}_D$, $\mathcal{T}_S$, N, n)

1: **Initialize:** weight vector $\mathbf{w}^{(l)} = [w_l, \ldots, w_{t_{1_{\max}}+n}]$
2: **for** $i = l : N$ **do**
3: Set $\mathbf{p}^{(l)} = \mathbf{w}^{(l)} / \left(\sum_{i=1}^{t_{1_{\max}}+n} w_i^{(l)} \right)$
4: Learn a hypothesis $h^{(l)}$ by calling $\mathcal{GP}_2$ and passing the distribution $\mathbf{p}^{(l)}$ over the combined data set $\mathbf{\Gamma_2} = \mathcal{T}_D \cup \mathcal{T}_s$.
5: Compute the prediction error of $h^{(k)}$ using:

$$\epsilon^{(l)} = \sum_{i=t_{1_{\max}}+1}^{t_{1_{\max}}+n} \frac{w_i^{(l)} |h^{(k)}(i) - u_1^{(i)p}|}{Z^{(k)}} \tag{10}$$

6: Perform weight updates
7: **end for**
8: **Output:**
Model hyperparamters $\mathbf{\Theta}_2$, the two probability distributions of $\mathcal{T}_D$ and $\mathcal{T}_S$, $\mathbf{p}_1$ and $\mathbf{p}_2$, respectively

5. MATHEMATICAL DERIVATION AND ERROR BOUNDS

Next the mathematical derivations of the error bounds for each of the proposal rules is presented.

5.1 First Bidding Strategy

This section derives the error bounds of the first proposing rule (Equation 8). In this case the utility prediction in the target task at any instant in time[7] is given by the following:

$$u_2 = \mathbf{p}_1 u_1^p + \mathcal{N}_2(\boldsymbol{\mu}_2, \mathbf{\Sigma}_2)$$

where, $\mathbf{p}_1$ is the normalized fitted weight vector, u_1^p is the source task utility model, and $\mathcal{N}_2(\boldsymbol{\mu}_2, \mathbf{\Sigma}_2)$ is the prediction made by the model of the target task with average $\boldsymbol{\mu}_2$ and a covariance matrix $\mathbf{\Sigma}_2$. The following theorem is presented:

THEOREM 5.1 (BOUNDS PARTIAL UPDATE). *Given a source and a target task negotiation agent, a different distribution data set $\mathcal{T}_D$, a same distribution data set $\mathcal{T}_s$, a model of the source task $\mathcal{N}(\boldsymbol{\mu}_1, \mathbf{\Sigma}_1)$, and $\epsilon^{\mathcal{U}}$. The agent opponent's utility model u_2 in the target task has a lower bound given by:*

$$u_2 \geq \mathcal{N}_2(\boldsymbol{\mu}_2, \mathbf{\Sigma}_2)$$

with $\mathcal{N}_2(\boldsymbol{\mu}_2, \mathbf{\Sigma}_2)$ being the posterior of the learned model in the target task so far.

PROOF. Since $\mathbf{p}_1$ is a valid probability distribution, then it should belong to the $[0, 1]$ simplex and therefore $0 \leq \mathbf{p}_1 \leq 1$. Manipulating the equations we reach to:

$$\begin{aligned} & 0 \leq \mathbf{p}_1 \leq 1 \\ & 0 \leq \frac{u_2 - \mathcal{N}_2(\boldsymbol{\mu}_2, \mathbf{\Sigma}_2)}{u_1^p} \leq 1 \\ & \mathcal{N}_2(\boldsymbol{\mu}_2, \mathbf{\Sigma}_2) \leq u_2 \leq u_1^p + \mathcal{N}_2(\boldsymbol{\mu}_2, \mathbf{\Sigma}_2) \end{aligned} \tag{11}$$

□

The above theorem is asserting that if the agent used transfer it will be performing at least as good as the non transfer case. Please note that this is valid as long as the similarity between the utility distributions of the source and the target task is within a certain threshold $\epsilon^{\mathcal{U}}$ which restricts the norm between the target model, $\mathcal{N}_2(\boldsymbol{\mu}_2, \mathbf{\Sigma}_2)$, and that of the source u_1^p.

5.2 Second Bidding Strategy

In the second case (Equation 9), the algorithm employs the following model:

$$u_2 = \mathbf{p}_1 u_1^p + \mathbf{p}_2 \mathcal{N}_2(\boldsymbol{\mu}_2, \mathbf{\Sigma}_2)$$

THEOREM 5.2 (BOUNDS FULL UPDATE). *Given a source and a target task negotiation agent, a different distribution data set $\mathcal{T}_D$, a same distribution data set $\mathcal{T}_s$, a model of the source task $\mathcal{N}(\boldsymbol{\mu}_1, \mathbf{\Sigma}_1)$, a positive real number β^l, and $\epsilon^{\mathcal{U}}$. The agent opponent's utility model u_2 in the target task has a lower bound given by:*

$$u_2 > \boldsymbol{p}_2 u_1^p \beta^l + \boldsymbol{p}_2 \mathcal{N}_2(\boldsymbol{\mu}_2, \mathbf{\Sigma}_2)$$

with $\mathcal{N}_2(\boldsymbol{\mu}_2, \mathbf{\Sigma}_2)$ being the posterior of the learned model in the target task so far. In the equality case of $\boldsymbol{p}_2 u_1^p \beta^l$ this utility has an additional bound given by:

$$u_2 \leq u_1^p \beta^l + \mathcal{N}_2(\boldsymbol{\mu}_2, \mathbf{\Sigma}_2)$$

[7]Please note that the time notation is omitted.

and thus,

$$u_1^p\beta^l + \mathcal{N}_2(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) < u_2 < \boldsymbol{p}_2 u_1^p \beta^l + \boldsymbol{p}_2 \mathcal{N}_2(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$

The equality case of this theorem is applicable when considering the bounds of $\mathbf{p}_2 u_1^p \beta^l$, which are detailed in Equation 16 of the proof.

PROOF. Since $\mathbf{p}_1$ and $\mathbf{p}_2$ are two probability distributions they belong to the $[0, 1]$ simplex. In other words, $0 \leq \mathbf{p}_1 \leq 1$ and $0 \leq \mathbf{p}_2 \leq 1$. The values of $\mathbf{p}_1$ and $\mathbf{p}_2$ are updated according to:

$$\begin{aligned} \mathbf{p}_1 &= \mathbf{w}\beta^{|\mathcal{N}_2(\boldsymbol{\mu}_2,\boldsymbol{\Sigma}_2)-\mathcal{N}_1(\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1)|} \\ \mathbf{p}_2 &= \mathbf{w}\beta_2^{-|\mathcal{N}_2(\boldsymbol{\mu}_2,\boldsymbol{\Sigma}_2)-\mathcal{N}_1(\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1)|} \end{aligned} \tag{12}$$

where $\beta = \frac{1}{\sqrt{1+\ln n/N}}$ which is constant for each iteration, and $\beta_2 = \frac{\epsilon}{1-\epsilon}$ with $\epsilon < \frac{1}{2}$.
Dividing $\mathbf{p}_1$ by $\mathbf{p}_2$ leads to:

$$\frac{\mathbf{p}_1}{\mathbf{p}_2} = \left(\frac{\beta}{\beta_2}\right)^l \tag{13}$$

where $l = |\mathcal{N}_2(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) - \mathcal{N}_1(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)|$.
Next:

$$\begin{aligned} &\epsilon < \frac{1}{2} \text{ then } \frac{\epsilon}{1-\epsilon} < 1 \\ &\frac{1}{\beta_2} > 1 \text{ then } \frac{\beta}{\beta_2} > \beta \\ &\text{and } \left(\frac{\beta}{\beta_2}\right)^l > \beta^l \end{aligned} \tag{14}$$

Therefore, from Equation 13:

$$\frac{\mathbf{p}_1}{\mathbf{p}_2} > \beta^l \tag{15}$$

Back to the update rule of Equation 9:

$$\begin{aligned} &\frac{u_2}{\mathbf{p}_2} = \frac{\mathbf{p}_1}{\mathbf{p}_2} u_1^p + \mathcal{N}_2(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \\ &\frac{u_2}{u_1^p \mathbf{p}_2} - \frac{\mathcal{N}_2(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)}{u_1^p} > \beta^l \text{ from Equation 15} \\ &u_2 > \mathbf{p}_2 u_1^p \beta^l + \mathbf{p}_2 \mathcal{N}_2(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \\ &\text{and } 0 \leq \mathbf{p}_2 u_1^p \beta^l \leq u_1^p \beta^l \\ &\text{and } 0 \leq \mathbf{p}_2 \mathcal{N}_2(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \leq \mathcal{N}_2(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \\ &0 \leq \mathbf{p}_2 u_1^p \beta^l + \mathbf{p}_2 \mathcal{N}_2(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \leq u_1^p \beta^l + \mathcal{N}_2(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \\ &u_2 \leq u_1^p \beta^l + \mathcal{N}_2(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \end{aligned} \tag{16}$$

□

6. EXPERIMENTS AND RESULTS

The performance evaluation was done with GENIUS (General Environment for Negotiation with Intelligent multipurpose Usage Simulation [9]) which is also used as a competition platform for the international Automated Negotiating Agents Competition (ANAC). The four domains used were: (1) Barbecue, (2) House keeping, (3) Rental house, and (4) Outfit (for the description of each domain, refer to [1]). Two sets of experiments where conducted. In the first the source and the target task's opponents were similar, while in the second the target opponent differed significantly from the source one.

6.1 Similar Source and Target Opponents

In the source task the agent faced the second ranked ANAC 2012 agent, the *AgentLG*. The target opponents had similar negotiation power. These were the following agents: (1) *CUHKAgent* (1*st* place of the 2012 ANAC), (2) *OMACagent* (joint 3rd place of the 2012 ANAC), and (3) *TheNegotiator Reloaded* (joint 3rd place of the 2012 ANAC). The negotiation intervals were set to 3 seconds and the maximum negotiation time was 180 seconds. After approximating a model of the source task's opponent, the agent uses the proposed transfer algorithm to negotiate against different target opponents in each of the previous domains. For the results to be statistically significant, the negotiation session against each opponent in each domain was repeated 200 times. Figure 1, shows the performance using the first bid proposition rule of the transfer algorithm. The dots and triangles show the averaged final utility in the four domains. The error bars represent the deviation negotiating against the different target opponents. It is clear from the results that the transfer agent (i.e., red curve) outperformed the no transfer case[8] shown in blue in all four domains.

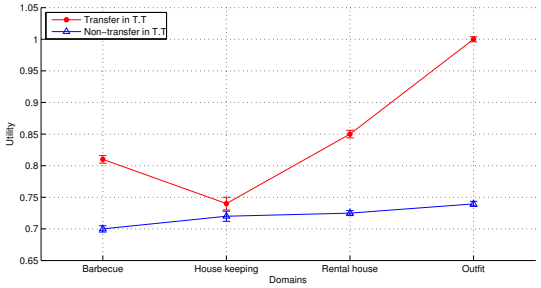

Figure 1: Transfer v.s. no transfer results. The graph shows the results using the first proposed rule in the four domains. The dots (triangles) present the means of the final utility, while the error bars show the corresponding deviation.

In the second set of experiments, the transfer agent negotiated in the same previous domains against the same opponents, but with the second bidding strategy of the transfer algorithm. The assessment of the transfer benefit was obtained as in the previous case. The results reported in Figure 2, clearly show that also here the transfer agent outperformed the no transfer one.
This leads us to the following conclusion:

Conclusion I: The proposed transfer algorithm is capable of significantly outperforming the non transfer negotiation agents in tasks with similar opponents using either of the two proposal strategies.

6.2 Dissimilar Opponents

In this set of experiments the source and target opponents where tougher and significantly dissimilar. First, in the source task, the agent negotiates against the *OMACagent*. Then, it commences to negotiate in the target task using the transfer algorithm against: (1) *IAMHaggler2011* (3rd place of the 2011 ANAC), (2) *BRAMAgent* (4th place of the 2011 ANAC), and (3) *Agent_K2* (5th place of the 2011 ANAC). The maximum negotiation steps were set to 100 and split into 3 seconds intervals. The negotiation was repeated 200 times against each of the above agents in the four domains,

[8]Please note that the no transfer case here refers to the normal negotiation setting.

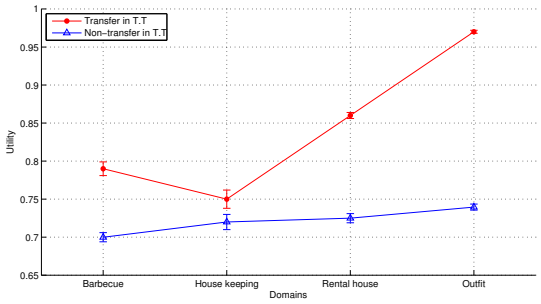

Figure 2: Transfer v.s. no transfer results. The graph shows the results using the second proposed rule in the four domains. The dots (triangles) present the means of the attained final utility, while the error bars show the corresponding deviation.

where the mean and variance of the final utility were calculated and used for quality assessment. The results using the first proposition rule are shown in Figure 3. It is interesting to see that once the source and target tasks became more dissimilar the transfer agent performed similar to the no transfer case. This confirms our hypothesis. It is also worth noting, that in the specific domain of Outfit the transfer agent was able to outperform the no transfer case.

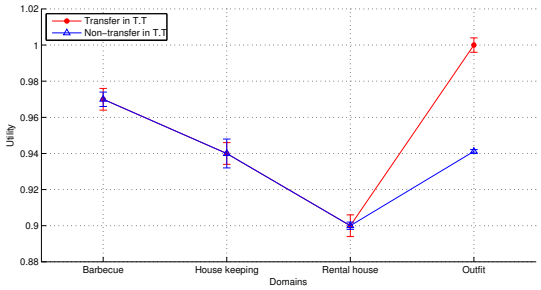

Figure 3: Transfer v.s. no transfer results in dissimilar source and target tasks. The graph shows the results using the first proposed rule in the four domains. The dots (triangles) present the means of the attained final utility, while the error bars show the corresponding deviation.

The results for the second bidding rule are shown in Figure 4. These results are very similar to those of the first bidding rule, although a comparison of Figure 4 and Figure 3 seems to indicate that for pairs of dissimilar opponents, the second bidding strategy could outperform the first.

This leads us to the following conclusions:

Conclusion II: As the source and the target task opponents become more dissimilar, the transfer performance decreases.

Conclusion III: Between pairs significantly dissimilar source and target opponents, the second bidding rule is more likely to perform best.

7. DISCUSSION

One important point is that the proposed method is function approximation independent. Since the opponent's model might be complex, a nonparametric functional prior (i.e., Gaussian processes)

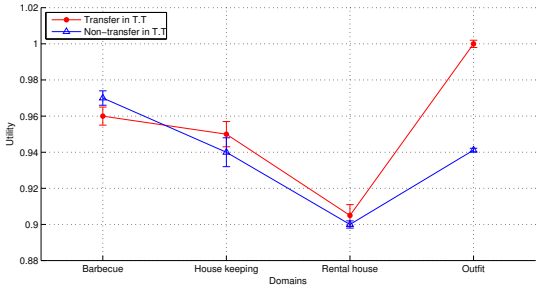

Figure 4: Transfer v.s. no transfer results in dissimilar source and target tasks. The graph shows the results using the second proposed rule in each of the four domains. The dots (triangles) present the means of the attained final utility, while the error bars show the corresponding deviation

that can automatically avoid overfitting has been employed. It is worth noting, although that GPs are considered to be one of the most powerful function approximation techniques, they suffer from computational complexity problems when dealing with large data sets. The solution for this problem is out of the scope of this paper and will be dealt with in future work. Moreover, any other function approximation scheme is equally applicable.

The presented results clearly demonstrate the applicability and efficacy of transfer learning for negotiation tasks. The proposed transfer technique operates within the same domain of multi-issue negotiations. Operating in such a setting allows the agents to avoid much of the computational complexity encountered otherwise. In other words, if the transfer had to operate in different negotiation domains an inter-task mapping that relates the source and target dimensions would have been required.

Negative Transfer

Although the proposed method has been shown to operate well in specific negotiation domains, transfer learning always carries the possibility of negative transfer. Negative transfer is a well known and a vital unanswered question in transfer learning. In most cases, this problem is ill-defined. Namely, it is hard to generally define negative transfer covering all possible performance measures. In the negotiation task, negative transfer occurs when transferring from the source to the target task actually hurts the learning of the target agent using one specific quality measure, such as the final utility. Answering the question of negative transfer in negotiation settings is out of the scope of this paper and is left for future work as it requires a quantification of the differences between the source and the potential target opponents. However, we do present some ideas that are helpful to avoid this problem. The performance in the target task depends on the difference measure between the source and target task utility and strategy distributions. In other words, as ϵ^{Π} and $\epsilon^{\mathcal{U}}$ increase, the performance in the target task will diminish. One idea to solve this problem is to use a certain performance measure γ — such as the final attained utility — to quantify the relation between ϵ^{Π}, $\epsilon^{\mathcal{U}}$ and γ. More specifically, a set of source and target negotiation tasks can be generated by varying the strategy and utility distributions. The proposed transfer algorithms are then applied and after the negotiation session terminated the performance measure γ is calculated. This gives rise to a data set $\mathcal{D} = \{(\langle \epsilon_i^{\Pi}, \epsilon_i^{\mathcal{U}} \rangle, \gamma_i)\}_{i=1}^{o}$, where o is the index of the different tasks that can be used to determine a data driven negative transfer measure. Specifically, a regression problem could be formulated

in which a mapping from the distribution difference to the performance measure is learned. Using this function any new negotiation task could be assessed according to this measure to determine whether negative transfer is likely to occur.

8. CONCLUSIONS AND FUTURE WORK

This paper proposes the first robust and efficient transfer learning in negotiation tasks algorithm. The transfer technique makes use of adaptation of TrAdaBoost — a well known supervised transfer algorithm — to aid learning against a new negotiation opponent. Two bidding strategy variants were proposed. In the first the target task agent proposes offers based only on weighting the source task model, while in the second, the agent bids depending on a weighting of both the source and the target models. Experimental results show the applicability of both approaches. More specifically, the results show that the proposed agent outperforms state-of-the-art negotiation agents in various negotiation domains with a significant margin. They further demonstrate that, as the source and target opponents become increasingly dissimilar, the transfer gain diminishes to a minimum. This minimum value relates to the lower bounds derived as second major contribution of the paper. It is also worth noting that, through the derivation of the lower bounds for the first bidding strategy, it becomes clear that the worst the transfer agent performs is equal in performance to the no transfer case.

There are a lot of interesting future directions of this work. For instance, the quantification of negative transfer, as well as the robustness of the algorithm are some of the essential steps that need to be studied. Further, to deal with the computational problem of Gaussian processes various other variants of supervised learning algorithms can be considered.

9. REFERENCES

[1] ANAC2012. http://anac2012.ecs.soton.ac.uk/.

[2] J. Brzostowski and R. Kowalczyk. Predicting partner's behaviour in agent negotiation. In *Proceedings of the Fifth Int. Joint Conf. on Autonomous Agents and Multiagent Systems*, AAMAS '06, pages 355–361, New York, NY, USA, 2006. ACM.

[3] R. Carbonneau, G. E. Kersten, and R. Vahidov. Predicting opponent's moves in electronic negotiations using neural networks. *Expert Syst. Appl.*, 34:1266–1273, February 2008.

[4] S. Chen and G. Weiss. An efficient and adaptive approach to negotiation in complex environments. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012)*, Montpellier, France, 2012. IOS Press.

[5] R. M. Coehoorn and N. R. Jennings. Learning on opponent's preferences to make effective multi-issue negotiation trade-offs. In *Proceedings of the 6th Int. conf. on Electronic commerce*, ICEC '04, pages 59–68, New York, NY, USA, 2004. ACM.

[6] W. Dai, Q. Yang, G. rong Xue, and Y. Yu. Boosting for transfer learning. In *In ICML*, 2007.

[7] J. Dang and M. N. Huhns. Concurrent multiple-issue negotiation for internet-based services. *IEEE Internet Computing*, 10:42–49, 2006.

[8] P. Faratin, C. Sierra, and N. R. Jennings. Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems*, 24(4):159–182, 1998.

[9] K. Hindriks, C. Jonker, S. Kraus, R. Lin, and D. Tykhonov. Genius: negotiation environment for heterogeneous agents. In *Proceedings of AAMAS 2009*, pages 1397–1398, 2009.

[10] C. Hou. Predicting agents tactics in automated negotiation. In *Intelligent Agent Technology, IEEE / WIC / ACM International Conference on*, volume 0, pages 127–133, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[11] J. B. Kim and A. Segev. A web services-enabled marketplace architecture for negotiation process management. *Decision Support Systems*, 40(1):71 – 87, 2005.

[12] R. Y. Lau, Y. Li, D. Song, and R. C. W. Kwok. Knowledge discovery for adaptive negotiation agents in e-marketplaces. *Decision Support Systems*, 45(2):310 – 323, 2008.

[13] R. Lin, S. Kraus, J. Wilkenfeld, and J. Barry. Negotiating with bounded rational agents in environments with incomplete information using an automated agent. *Artificial Intelligence*, 172:823–851, April 2008.

[14] S. J. Pan and Q. Yang. A survey on transfer learning.

[15] I. Ponka. *Commitment models and concurrent bilateral negotiation strategies in dynamic service markets*. PhD thesis, University of Southampton, School of Electronics and Computer Science, 2009.

[16] H. Raiffa. *The art and science of negotiation*. Harvard University Press Cambridge, Mass, 1982.

[17] C. E. Rasmussen. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[18] A. Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica*, 50(1):97–109, 1982.

[19] D. Somefun, E. Gerding, S. Bohte, and J. L. Poutre. Automated negotiation and bundling of information goods. In P. Faratin, D. Parkes, J. Rodriguez-Aguilar, and W. Walsh, editors, *Agent-Mediated Electronic Commerce V: Designing Mechanisms and Systems, Springer Lecture Notes in AI, vol. 3048*, pages 1–17. Springer-Verlag, 2004.

[20] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.*, 10:1633–1685, December 2009.

[21] C. Williams, V. Robu, E. Gerding, and N. Jennings. Using gaussian processes to optimise concession in complex negotiations against unknown opponents. In *In Proceedings of the 22nd Internatioanl Joint Conference on Artificial Intelligence.* AAAI Press, 2011.

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