Künstl Intell (2014) 28:21–27
DOI 10.1007/s13218-013-0284-x

TECHNICAL CONTRIBUTION

Transfer for Automated Negotiation

Siqi Chen · Haitham Bou Ammar · Karl Tuyls · Gerhard Weiss

Received: 15 July 2013 / Accepted: 25 October 2013 / Published online: 3 December 2013

Abstract Learning in automated negotiation is a difficult problem because the target function is hidden and the available experience for learning is rather limited. Transfer learning is a branch of machine learning research concerned with the reuse of previously acquired knowledge in new learning tasks, for example, in order to reduce the amount of learning experience required to attain a certain level of performance. This paper proposes a novel strategy based on a variation of TrAdaBoost—a classic instance transfer technique—that can be used in a multi-issue negotiation setting. The experimental results show that the proposed method is effective in a variety of application domains against the state-of-the-art negotiating agents.

Keywords Automated negotiation · Transfer learning · Opponent modeling

1 Introduction

In automated negotiation two (or more) agents try to come to a joint agreement in a consumer-provider or buyer-seller setup [10]. One of the biggest driving forces behind research into automated negotiation is the broad spectrum of potential applications. For example, Ponka et al. [14] propose the adoption of automated negotiation in electronic commerce and Lau et al. [11] make use of the framework to aid in electronic markets. Other application domains of automated negotiation include supply chain management [19] and pervasive computing [13].

Automated negotiations come in many forms such as sequential versus concurrent negotiations (i.e., multiple negotiations occur one after the other or at the same time), bilateral versus multilateral negotiations (i.e., an agent negotiates with a single other or multiple other opponents), and single issue versus multi-issue negotiations (i.e., a single or several issues are subject of negotiation among agents). In the work reported here, the widely used bilateral multi-issue negotiation setting is considered. In this negotiation setting, two agents negotiate with the goal to agree on a profitable contract for a product or a service. Such a contract consists of multiple issues that are of conflictive importance to the negotiators. We assume (1) that an agent is limited in that it has no prior knowledge about its opponents' utility models and/or strategies and (2) that the profit an agent gets for reaching an agreement is discounted over negotiation time (i.e., the longer it takes to reach an agreement the lower is the profit).

Given the pervasive nature of automated negotiation, negotiating agents are required to have a high level of self-determination, whereby they decide on their own what actions they should perform when and under what conditions so that they end up in satisfactory agreements. This kind of self-determination, however, is very challenging to achieve, mainly due to the lack of sufficient knowledge about the opponents. Modeling the opponent's behavior is one of the common approaches to predict the negotiation outcome. Different algorithms capable of modeling the

S. Chen (✉) · H. B. Ammar · G. Weiss
Department of Knowledge Engineering, Maastricht University, P.O. Box 616, 6200 MD Maastricht, The Netherlands
e-mail: siqi.chen09@gmail.com

H. B. Ammar
e-mail: haitham@bouammar.com

G. Weiss
e-mail: gerhard.weiss@maastrichtuniversity.nl

K. Tuyls
Department of Computer Science, University of Liverpool, Ashton Street, L69 3BX Liverpool, UK
e-mail: k.tuyls@liverpool.ac.uk

opponent's behavior in negotiation settings have been proposed. Chen and Weiss in [4] proposed the negotiation approach "OMAC" that learns the opponent's strategy to predict utilities of future counter-offers through discrete wavelet decomposition and cubic smoothing splines. The optimal concession is then made accordingly. Hao et al. [8] introduced a negotiation strategy named ABiNes (while the implementation is called CHUKAgent) to deal with negotiations in complex environments. To successfully perform negotiations, ABiNeS adjusts the time to stop exploiting the negotiating partner and also employs a reinforcement-learning approach to improve the acceptance probability of its proposals. In order to leverage the problem of a substantial amount of time required in learning, Chen et al. [3] proposed a sparse-Gaussian processes (SPGPs) framework. SPGPs are a form of nonparametric regression techniques capable of effectively modeling a latent relation between a set of dependent and independent data instances. Rather than using the overall available data, SPGPs make use of the so-called *pseudo-inputs*. These are typically smaller in number than the overall data set and are fitted such that their locations capture the main trends of variation in the latent function. Experiments in [3], have shown a reduction in the computational complexity required to learn an opponent model in automated negotiation.

Although successful, these methods have paid little attention to the problem of reusing of availability of data and knowledge. This problem can also be framed in terms of complexity as folows: howto reduce the amount of samples needed to learn a successful behavior? Transfer learning (TL) is a promising technique to tackle this problem [1, 12, 18]. The main idea in TL is to reuse knowledge attained in a previously encountered task to aid learning in a new task. In TL, there typically exists a source and a target task. The agent has already acquired a "good-enough" behavior in the source. This knowledge is available to be used in the target. When attempting to transfer, three main questions stand-out: what to transfer, when to transfer, and how to transfer. The first question addresses the type of knowledge to be transferred (e.g., instances, features, et. cetera); the second questions concerns the learning phase in which transfer is to be conducted (e.g., online, offline, et. cetera); and the third question is concerned with the "correct" mapping of source knowledge to the target task (e.g., weighting instances, mapping features, et. cetera).

This paper describes work that aims at at efficient opponent modeling in automated negotiation and contributes to this goal by:

1. Proposing a formalization for transfer in automated negotiation;
2. Proposing an instance transfer algorithm for automated negotiation based on TrAdaBoost;

Experiments performed on various state-of-the-art negotiation tasks show that transfer can indeed aid target agents in improving their behaviors once encountering new opponents varying in their preference profiles, bidding strategies, and/or utility models.

The rest of the paper is organized as follows. Sect. 2 provides the reader with necessary background knowledge needed to understand the remainder of the paper. Sect. 3 details the proposed algorithm. In Sect. 4 experimental results are shown and Sect. 5 discusses these results. Finally, Sect. 6 identifies promising future research directions.

2 Background

In this section background material is provided. Firstly, the overall automated negotiation setting is explained. Secondly, the regression framework (i.e., Gaussian Processes) is presented. TrAdaBoot, being the basis for one of the proposed transfer methods, is then detailed.

2.1 Negotiation Framework

As stated above, the automated negotiation framework adopted in this paper is a basic bilateral multi-issue negotiation model as it is widely used in the agents field (e.g., [4–6]). The negotiation protocol is based on a variant of the alternating offers protocol proposed in [15].

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. Formally, let J be the set of issues under negotiation where $j\,(j \in \{1, \ldots, n\})$ 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 [17], 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 (1) to accept or (2) 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 $j \in \{1, \ldots, n\}$. 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:

$$U^i(O) = \sum_{j=1}^{n} (w_j^i \cdot V_j^i(O_j)) \quad (1)$$

where w_j^i and O are as defined above and V_j^i 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 [15]. 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 [16], 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 Eq. 2.

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

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 [16] 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 Eq. 2 with respect to $\boldsymbol{\Theta}$ the vector of all hyperparameters. Typically, this maximization requires the computation of the derivatives of Eq. 2 with respect to $\boldsymbol{\Theta}$. These derivatives are then used in a gradient-based algorithm to perform the updates to the hyperparameters θ_j.

2.3 Boosting for Transfer Learning

TrAdaBoost [7] 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. In this work as shown in Sect. 3.3, an adaption of the algorithm to deal with regression is also presented.

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[1] and $\mathcal{Y} = \{0, 1\}$ is the set of labels[2]. A concept c is a function mapping from $\mathcal{X} \rightarrow \mathcal{Y}$ 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 $\mathcal{T}_D = \{(x_D^{(i)}, c(x_D^{(i)}))\}_{i=1}^{m}$ where $x_D^{(i)} \in \mathcal{X}_D$ and $\mathcal{T}_S$ represents the target task data set where $\mathcal{T}_S = \{(x_s^{(j)}, c(x_s^{(j)}))\}_{j=1}^{n}$. 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:

$$x^{(1)} = \begin{cases} x_d^{(i)} & \text{for } l=1,\ldots,m \\ x_s^{(i)} & \text{for } l=m+1,\ldots,m+n \end{cases} \quad (3)$$

Using these definitions the problem that TrAdaBoost tries to solve is: given a small number 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 $\hat{c} : \mathcal{X} \rightarrow \mathcal{Y}$ that minimizes the prediction error on the unlabeled data set $\mathcal{S}$.

The main idea behind TrAdaBoost is to weight source instances such that these relevant to the target task attain a high weighting factor, while the ones that hurt the target attain a lower weight. For a detailed description, the reader is referred to [7].

3 Transfer in Automated Negotiation

This section describes the proposed transfer techniques. First, transfer formalization in automated negotiation is

[1] 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".

[2] Extending TrAdaBoost to multi-class classification problems is fairly straight forward.

explained. Second, learning in the source task is detailed. Finally, the proposed method for instance transfer in automated negotiation is explained.

3.1 Transfer Formalization in Automated Negotiation

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 $\boldsymbol{\Pi} = \{\pi_1, \pi_2\}$, 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 $\epsilon^{\boldsymbol{\Pi}}$. In other words, the distance between the two strategy probability distributions – $\chi_{\Pi}^{(1)}$ and $\chi_{\Pi}^{(2)}$ is not larger than $\epsilon^{\boldsymbol{\Pi}}$, given by the following equation:

$$D_{KL}^{\Pi}(\chi_{\Pi}^{(1)} || \chi_{\Pi}^{(2)}) = \int_{-\infty}^{\infty} \chi_{\Pi}^{(1)}(\mathbf{x}) \ln \frac{\chi_{\Pi}^{(1)}(\mathbf{x})}{\chi_{\Pi}^{(2)}(\mathbf{x})} d\mathbf{x} \leq \epsilon^{\mathbf{\Pi}} \tag{4}$$

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.[3]. 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 $\epsilon^{\mathcal{U}}$. Moreover, we hypothesize that as the difference of strategy probability distributions or utility models grows, the transfer learning performance decreases, and vice versa.

3.2 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[4]. 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:

$$\begin{aligned} & u_1^{\star} | \boldsymbol{\Gamma_1}, \mathbf{u_1}, \boldsymbol{t_1^{\star}} \sim \mathcal{N}(\mathbf{u_1^{\star}}, \boldsymbol{cov}(\mathbf{u_1^{\star}})) \\ & \text{with} \\ & \bar{\mathbf{u}}_1^{\star} = \mathbf{K}_1(t_1^{\star}, \boldsymbol{\Gamma_1}) \left[\mathbf{K_1}(\boldsymbol{\Gamma_1}, \boldsymbol{\Gamma_1}) + \boldsymbol{\sigma_1^2}\mathbf{I}\right]^{-1} \mathbf{u_1} \\ & \text{cov}(\mathbf{u}_1^{\star}) = \mathbf{K}_1(t_1^{\star}, t_1^{\star}) - \mathbf{K}_1(t_1^{\star}, \boldsymbol{\Gamma_1}) \left[\mathbf{K_1}(\boldsymbol{\Gamma_1}, \boldsymbol{\Gamma_1}) + \boldsymbol{\sigma_1^2}\mathbf{I}\right]^{-1} \\ & \quad \cdot \mathbf{K}_1(\boldsymbol{\Gamma_1}, \boldsymbol{t_1^{\star}}) \end{aligned} \tag{5}$$

where $u_1^{\star}$ is the predicted value at $t_1^{\star}$, $\boldsymbol{\Gamma}_1$ is the matrix of all the history of inputs till $t_1^{\star}$, $\mathbf{u}_1$ presents the vector of utilities collected so far, $\mathbf{K}_1(t_1^{\star}, \boldsymbol{\Gamma}_1)$ is the covariance matrix formed between the new input and the history of all inputs, $\mathbf{K}_1(\boldsymbol{\Gamma}_1, \boldsymbol{\Gamma}_1)$ 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.

3.3 Instance Transfer in Automated Negotiation

Following the former formalization, it is now intuitive to extend the negotiation framework to a setting that is similar

[3] 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.

[4] In this work we split the negotiation session in intervals of 3 s.

to that of TrAdaBoost. To define the transfer problem we need to first define the data sets of both the source and the target tasks. The different distribution data set is defined as, $\mathcal{T}_D = \{t_1^{(i)}, \mathcal{GP}_1(t_1^{(i)})\}_{i=1}^{m}$. In the target task, the same distribution dataset is defined as, $\mathcal{T}_S = \{t_2^{(j)}, u_2^{(j)}\}_{j=1}^{n}$ with $n <<< m$. In other words, in the target task a handful of labels is available. The agent can gather these through the interaction with the target task opponent.

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 1. This function is detailed in the pseudo-code of Algorithm 2. The algorithm follows the same steps as in 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 Eq. 7 represents the source model prediction.

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 Eq. 6 in line 5 of Algorithm 1. $\mathbf{p}_1$ and $\mathbf{p}_2$ in Eq. 6 are the fitted distributions over the same and different distribution datasets. Moreover, $\mathcal{T}_S$. 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 1 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:

$$u_2^{(k)} = \mathbf{p}_1^{(k)} u_1^{(k)} + \mathbf{p}_2^{(k)} u_2^{(k)p} \tag{6}$$

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., $\boldsymbol{\Theta}_2$)

Algorithm 2 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 $\boldsymbol{\Gamma_2} = \mathcal{T}_D \cup \mathcal{T}_S$.
5: Compute the prediction error of $h^{(k)}$ using:

$$\epsilon^{(l)} = \sum_{i=t_{1_{\max}}+1}^{t_{1_{\max}}+n} \frac{w_i^{(l)} |h^{(k)}(i) - u_1^{(i)p}|}{Z^{(k)}} \tag{7}$$

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

4 Experiments and Results

The performance evaluation was done with general environment for negotiation with intelligent multipurpose usage simulation (GENIUS [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 [2]). 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.

4.1 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 s and the maximum negotiation time was 180 s. 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 of the transfer algorithm. The X-axis shows the average final utility in the four domains with the error bars representing the standard deviation. It is clear from the results that the transfer agent (i.e., red curve) outperformed the no transfer case shown in blue in all four domains. Please note that the no transfer case here refers to the normal negotiation setting.

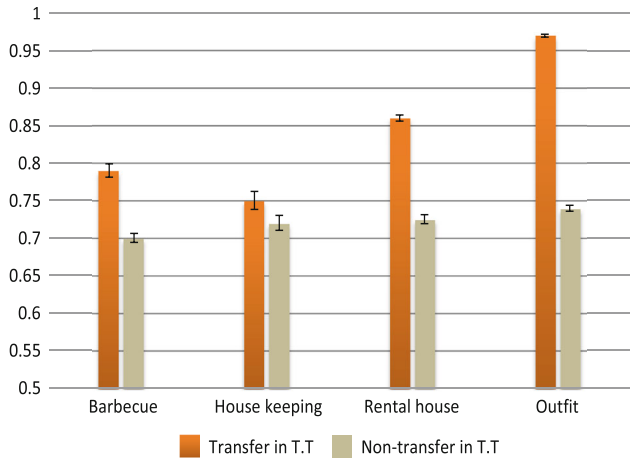

Fig. 1 Transfer vs. no transfer results in four test domains when the source and target tasks are similar

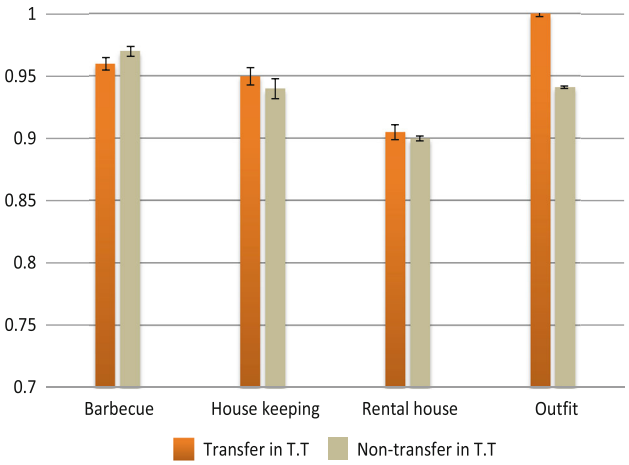

Fig. 2 Transfer vs. no transfer results in four test domains when the source and target tasks are dissimilar

This leads us to the conclusion that the agent using the proposed transfer algorithm is capable of significantly outperforming the non transfer negotiating agent in tasks against similar opponents.

4.2 Dissimilar Opponents

In this set of experiments the source and target opponents where 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 s intervals. The negotiation was repeated 200 times against each of the above agents in the four domains, where the mean and standard deviation of the final utility were calculated and used for quality assessment. The results are shown in Fig. 2. 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 we made in Sect. 3.1. It is also worth noting, that in the specific domain of Outfit the transfer agent was able to outperform the no transfer case. We speculate that it may be caused by a low level of competitiveness[5] of this domain (i.e., the lowest among the evaluation domains), which is likely to alleviate the problem of increasing difference of strategy probability distributions or utility models in these tasks with dissimilar opponents.

[5] Competitiveness refers to the minimum distance of possible outcomes in a domain to the point where both parties are both fully satisfied. To put it differently, agents tend to achieve better performance in a domain with lower competitiveness.

This leads us to the conclusion that as the source and the target task opponents become more dissimilar, the transfer performance decreases.

5 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) that can automatically avoid overfitting has been employed. It is worth noting that, although 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.

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 – 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 $\epsilon^{\Pi}, \epsilon^{\mathcal{U}}$ 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.

6 Conclusions and Future Work

This paper proposed a robust and efficient approach in transfer learning in negotiation tasks. The transfer technique makes use of adaptation of TrAdaBoost—a well known supervised transfer algorithm—to aid learning against a new negotiation opponent. Experimental results show the applicability of the learning scheme. More specifically, the results show that by using the proposed strategy the agent can significantly improve its performance in various negotiation domains. They further demonstrate that, as the source and target opponents become increasingly dissimilar, the transfer gain diminishes.

There are a lot of interesting future directions of this work. For instance, the quantification of negative transfer and the robustness of the transfer learning approach are important subjects of further research. Furthermore, the computational demands of Gaussian processes raises the question of what other variants of supervised learning are appropriate.

References

1. Ammar HB, Tuyls K, Taylor ME, Driessens K, Weiss G (2012) Reinforcement learning transfer via sparse coding. In: Proceedings of the 11th Int. Joint Conf. on Automomous Agents and Multi-Agent Systems. ACM, Valencia, p 383–390
2. ANAC' (2012) http://anac2012.ecs.soton.ac.uk/
3. Chen S, Ammar HB, Tuyls K, Weiss G (2013) Optimizing complex automated negotiation using sparse pseudo-input Gaussian processes. In: Proceedings of the 12th Int. Joint Conf. on Automomous Agents and Multi-Agent Systems. ACM, Saint Paul, p 707–714
4. Chen S, Weiss G (2012) An efficient and adaptive approach to negotiation in complex environments. In: Proceedings of the 20th European Conference on Artificial Intelligence. IOS Press, Montpellier, France, p 228–233
5. Chen S, Weiss G (2013) An efficient automated negotiation strategy for complex environments. Eng Appl Artif Intell 26(10):2613–2623
6. Coehoorn RM, Jennings NR (2004) 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. ACM, New York p 59–68
7. Dai W, Yang Q, Xue GR, Yu Y (2007) Boosting for transfer learning. In: Proceedings of the 24th international conference on Machine learning. ACM, New York, pages 193–200.
8. Hao J, Leung H (2012) ABiNeS: an adaptive bilateral negotiating strategy over multiple items. In: Proceedings of the 2012 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2012), Macau, China
9. Hindriks K, Jonker C, Kraus S, Lin R, Tykhonov D (2009) Genius: negotiation environment for heterogeneous agents. In: Proceedings of the 8th Int. Joint Conf. on Automomous Agents and Multi-Agent Systems, p 1397–1398
10. Jennings NR, Faratin P, Lomuscio AR, Parsons S, Sierra C, Wooldridge M (2001) Automated negotiation: prospects, methods and challenges. Int J Group Decis Negot 10(2):199–215
11. Lau RY, Li Y, Song D, Kwok RCW (2008) Knowledge discovery for adaptive negotiation agents in e-marketplaces. Decis Support Syst 45(2):310–323
12. Pan SJ, Yang Q (1010) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359
13. Park S, Yang S (2008) An efficient multilateral negotiation system for pervasive computing environments. Eng Appl Artif Intell 21(4):633–643
14. Ponka I (2009) Commitment models and concurrent bilateral negotiation strategies in dynamic service markets. PhD thesis, University of Southampton, School of Electronics and Computer Science
15. Raiffa H (1982) The art and science of negotiation. Harvard University Press Cambridge, Cambridge
16. Rasmussen CE (2006) Gaussian Processes for Machine Learning. MIT Press, Cambridge
17. Rubinstein A (1982) Perfect equilibrium in a bargaining model. Econometrica 50(1):97–109
18. Taylor ME, Stone P (2009) Transfer learning for reinforcement learning domains. A survey J Mach Learn Res 10:1633–1685
19. Wang M, Wang H, Vogel D, Kumar K, and Chiu DK (2009) Agent-based negotiation and decision making for dynamic supply chain formation. Eng Appl Artif Intell 22(7):1046–1055