Opponent Modeling in a PGM Framework

(Extended Abstract)

Nicolaj Søndberg–Jeppesen
Dept. of Computer Science
Aalborg University
DK–9220 Aalborg, Denmark
nicolaj@cs.aau.dk

Finn V. Jensen
Dept. of Computer Science
Aalborg University
DK–9220 Aalborg, Denmark
fvj@cs.aau.dk

Yifeng Zeng
School of Computing
Teesside University
Middlesbrough TS1 3BA, UK
y.zeng@tees.ac.uk

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Distributed Artificial Intelligence—*Intelligent agents, Multiagent systems*

Keywords

Opponent modeling; Influence diagrams; Adaptation

1. GAME DESCRIPTION AND NOTATION

We consider games with the following characteristics. The *scene* is visible to all players. Each player has a set of *actions*, which have an effect on the state of the scene. The players act concurrently, and the joint effect of the two actions may be non-deterministic. Each player is assigned a *score*, which is a function of the state of the scene, and it does not change during the game. This assignment is known only to the player. We assume everything to be finite, and we consider games with only two players. We shall call this kind of games a *Simple Sequential Bayesian Game (SSBG)*.

Formally, an SSBG consists of two players ♡, ♠ and a *world* W with a finite set of states. We assume ♡ to be female and ♠ to be male. The players have finite sets of *moves*, $M^{\heartsuit}$ and $M^{\spadesuit}$, which affect W. The transition between world states at time t to time $t+1$ is determined by a probabilistic function τ, $\tau : W \times M^{\heartsuit} \times M^{\spadesuit} \times W \to \mathbf{R}$, where $\tau(w^1, m^{\heartsuit}, m^{\spadesuit}, w^2)$ is the probability of W_{t+1} being in state w^2 given that W_t was in state w^1 and the two players make the moves $m^{\heartsuit}$ and $m^{\spadesuit}$, respectively. Furthermore, each player draws an assignment from a finite set $\mathcal{A}$. The assignment is a particular score function reflected in utility numbers over states of the world. We shall assume that ♡ has received the assignment a_1. The structure of the game and the world state is always known by both players, but the actual assignment of the other player remains hidden.

When both players have decided their moves, the game continues with the next time step. There is no prefixed limit on the number of moves, but the time for playing the game is so short that discounting is not relevant. That is, the players aim for maximizing the sum of the utilities gained during the game.

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

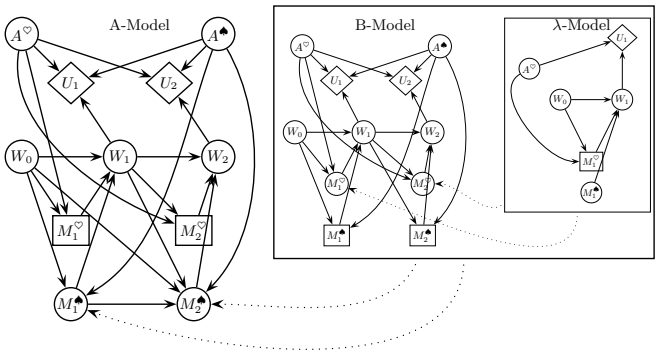

Figure 1: An ID framework for opponent modeling which contains a set of nested models. This one has nesting level 3 and time horizon 2

2. RECURSIVE INFLUENCE DIAGRAMS

An RID ([6]) is a dynamic influence diagram [7] that models the agent's subjective decision-theoretic reasoning and its reasoning about other agent's reasoning. Figure 1 shows an RID modeling an SSBG as seen in the eyes of ♡. The model consists of 3 influence diagrams, namely the A-Model, B-Model and the λ-Model, representing ♡'s model, ♡'s model of ♠ and ♡'s model of ♠'s model of ♡ respectively.

RIDs follow the same notations and conventions as influence diagrams [2]. The nodes labeled $A^{\heartsuit}$ and $A^{\spadesuit}$ represent the assignments; the nodes labeled with M represent moves; the U-labeled nodes represent score functions, and the Ws represent the world states. The transition function is represented by the conditional probability $P(W_1|W_0, M^{\heartsuit}, M^{\spadesuit})$.

The connection between the A-Model, the B-Model and the λ-Model is as follows: in the A-Model, ♡'s own decisions are represented as decision nodes while she represents ♠ decisions as chance nodes. That means that the policy for ♠ in each of his decisions must be represented as a conditional probability distribution. To find these, ♡ consults the B-Model - which in this case is another RID but this time representing the game in the eyes of ♠. In the B-Model, ♠'s decisions are represented as decision nodes and ♡'s decisions as chance nodes. The models can be solved using standard algorithms for solving IDs (see for instance [3, 4, 5]).

Probabilistic Graphical Models (PGMs) have previously been applied in opponent modeling frameworks [1, 8]. As opposed to those, RIDs have proven particularly effective in solving SSBGs [6].

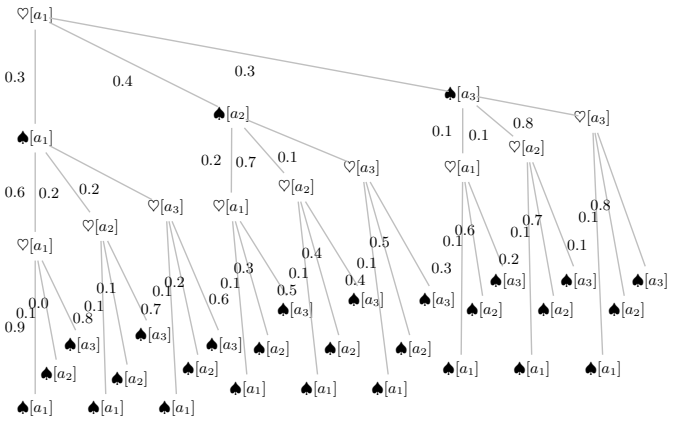

Figure 2: The tree representing the level 3 initial type for ♡. ♡ knows her own assignment a_1. From the root there is a branch to a sub tree for each possible ♠ assignment. The edges are labeled with the probability assigned to each subtree.

2.1 The Player Representation

RIDs have inherent assumptions about how many future time steps ♡ is considering, how many future time steps ♡ thinks ♠ is considering and finally how many future time steps ♡thinks♠ thinks ♡is considering. Furthermore, RIDs have inherent assumptions about how deep ♡'s nesting level is. In Figure 1 the A-Model, the B-Model and the λ-Model takes 2, 2 and 1 future time steps into account respectively, and the nesting level of the A-Model is 3.

3. TYPE TREES

To model the uncertainty of the opponent's assigment we use *type trees*. A type tree is a data structure that can be used to represent ♡'s probability distribution over ♠'s assignment. It can also represent ♡'s beliefs of ♠'s belief of ♡'s assignment etc (see Figure 2).

Each internal node in a type tree represents an acting agent, and we have to find the policy for each of them. So, we attach an influence diagram to each internal node.

3.1 Solving the type tree with IDs

The tree is solved from the leaves and up to the root. Taking the type tree in Figure 2, we first calculate the nine optimal policies in the leaves. They are used to determine the conditional probability table for $M^{\heartsuit}$ in the parents.

Next, the 3 IDs under the root are solved to provide the appropriate ♠-policies, which finally are used for solving the ID in the root. When the tree is solved, ♡ uses the recommendation from the model to decide for a move. ♠ also makes a move and both observe a new state of W. ♡ uses the new information to update the parameters of the model, and the updated model is used to determine the next move. That is, when deciding the next move, she uses the same time horizon and nesting depth as the first move.

3.2 Updating the type tree

When both players have taken a move and the resulting world state is observed, they shall use the new information to update their beliefs in order to determine their next move. This means that ♡ has to update her type tree. If the moves are public, standard Bayesian network (BN) algorithms are used for the updating. In the case of private moves, the private information grows during play, and the type tree grows exponentially. We address this problem in a separate paper.

4. MIXTURE MODELS FOR ADAPTATION

Formally, let $\Gamma_1, \ldots, \Gamma_k$ be models. Then a mixture model can be denoted as

$$\Gamma = \bigoplus_i \mu_i \Gamma_i, \tag{1}$$

where μ_i are positive reals for which $\sum_i \mu_i = 1$. We treat them as probabilities reflecting ♡'s belief in the various models. When calculating the policies in Γ, you combine the appropriate policies from the Γ_i-models as the sum weighted by the beliefs. Now, when information e has been collected, the probabilities for the various models change. Let $P(e|\Gamma_i)$ denote the probability of the evidence e if ♠ plays in accordance with Γ_i. Bayesian conversion yields $P(\Gamma_i|e) \simeq P(e|\Gamma_i)P(\Gamma_i)$, and standard BN algorithms provide $P(e|\Gamma_i)$ for all i, and we can use the collected information about ♠'s moves to adapt the mixture to his actual reasoning.

Over time inconsistency may emerge in the models. At time step t, ♡ receives observations and needs to update her type tree T_t. If the observations are inconsistent with T_t (i.e. the joint observations have probability 0 according to her model), Bayesian updating becomes invalid and ♡ will have no way of assessing the probabilities of T_{t+1}. This is a general problem in opponent modeling techniques. When ♡ has a model of ♠, it will inevitably contain a wrong model of ♡ (otherwise the model would be infinite) and inconsistent observations will eventually emerge. We have resolved the conflict by adding a baseline model, $\Gamma_0 = NIL$, that prescribes random behavior. As Γ_0 hypothesizes all possible actions, updating is always possible.

5. REFERENCES

[1] P. J. Gmytrasiewicz, S. Noh, and T. Kellogg. Bayesian update of recursive agent models. *User Modeling and User-Adapted Interaction*, 8(1-2):49–69, Jan. 1998.

[2] R. A. Howard and J. E. Matheson. Influence diagrams. In *Readings on the Principles and Applications of Decision Analysis*, pages 721–762, 1984.

[3] F. Jensen, F. V. Jensen, and S. L. Dittmer. From influence diagrams to junction trees. In *UAI*, pages 367–374, 1994.

[4] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34(6):597–609, 1986.

[5] P. P. Shenoy. Valuation-based systems for Bayesian decision analysis. *Operations Research*, 40(3):463–484, 1992.

[6] N. Søndberg-Jeppesen and F. V. Jensen. A pgm framework for recursive modeling of players in simple sequential bayesian games. *IJAR*, 5(51):587–599, 2010.

[7] J. A. Tatman and R. D. Shachter. Dynamic programming and influence diagrams. *IEEE Trans. on Systems, Man, and Cybernetics*, 20(2):365–379, 1990.

[8] Y. Zeng and P. Doshi. Exploiting model equivalences for solving interactive dynamic influence diagrams. *JAIR*, (43):211–255, 2012.