Safer Decisions Against A Dynamic Opponent

Wojciech Jamroga

Parlevink Group, University of Twente, Netherlands
Institute of Mathematics, University of Gdansk, Poland

Abstract. A hierarchy of beliefs for an agent has been proposed in [3]. The aim of this paper is to investigate the performance of such 'multi-model' agents with some experiments in the simplest possible case. The experiments consist of the agent's interactions with simulated agents, acting as customers of an imaginary Internet banking service.

`Keywords:` autonomous agents, user modeling, machine learning, decision making, beliefs, game theory.

1 Introduction

A software agent may clearly benefit from having an up-to-date model of her environment of activity. The model may, for example, include actual users' profiles or some assumptions being accepted by default. However, the agent doesn't have to stick to one model only, she can possess a set of complementary beliefs, both learned and assumed. The agent may then switch to the most appropriate one when making her decisions, or even combine several models under certain assumptions [3]. The aim of this paper is to investigate the performance of such 'multi-model' agents in the simplest possible case, i.e. in the case of an agent using exactly two alternative models of reality. The output of such a hybrid agent can be then compared with the performance of both 'single-model' agents alone to see if (and when) a software agent can really benefit from using a more complicated belief structure and decision making scheme.

The controversy between normative models (like non-cooperative equilibria from Game Theory) and adaptive models (obtained through some kind of learning) has been another inspiration for this paper. The adaptive solutions are more useful when the domain is cooperative or neutral; they also allow the agent to exploit deficiencies of her adversaries. The 'best defense' assumptions are still tempting, though, in a situation when the agent risks real money. Even one opponent who plays his optimal strategy persistently can be dangerous then. This paper presents another attempt to integrate both approaches. The main model used by the agent in the experiments is a profile of the user; the other model is based on the maxmin equilibrium.

1.1 The Game and the Agent

The experiments were inspired by the following scenario: a software agent is designed to interact with users on behalf of an Internet banking service; she

can make an offer to a user, and the user's response determines her output at this step of interaction. In the actual experiments the agent has had 3 possible offers at hand: the 'risky offer', the 'normal offer' and the 'safe offer', and the customer could respond with: 'accept honestly', 'cheat' or 'skip'. The complete table of payoffs for the game is given below. The 'risky offer', for example, can prove very profitable when accepted honestly by the user, but the agent will lose much if the customer decides to cheat; as the user skips an offer, the bank still gains some profit from the advertisements etc.

	accept	cheat	skip
risky offer	30	-100	0.5
normal offer	10	-30	0.5
safe offer	0.5	0	0.5

The banking agent is a 1-level agent, i.e. an agent that models other agents as stochastic (0-level) agents. The user is simulated as a random 0-level agent – in other words, his behavior can be described with a random probabilistic policy. The agent estimates the user's policy p with a relative frequency distribution $\hat{p}$, counting the user's responses. At the same time the agent computes a confidence value C for the profile acquired so far. The default model is defined in the Game Theory fashion: the user is assumed an enemy who always cheats.

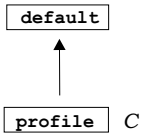

Fig. 1. The simplest hierarchy: two models of reality

The motivation behind the confidence C is the following: if a numerical evaluation can be computed for every decision with respect to a particular model (the expected payoff, for instance), then the agent's decision may be based on a linear combination of the evaluations, with the confidence providing weights. If the agent trusts the user's profile in, say, 70% – the final evaluation may depend on the profile in 70%, and the remaining 30% can be derived from the default model: $eval(a) = C\ eval_{profile}(a) + (1-C)\ eval_{default}(a)$. In consequence, the decision is based on both models at the same time, although in different proportions – weighting the partial evaluations with the confidence the agent has in them [3].

2 Confidence

In order to provide the agent with a way of computing her 'self-confidence', two measures are combined: the log-loss based confidence [4] and the datasize-related measure proposed by Wang [9].

Let $\hat{p}_i(\cdot|a)$ be the model of the user at his ith response to a, and b_i^* – his actual response at that step. To detect changes in the pattern of the user's behavior, a confidence measure based on the log-loss function [7] can be used. The one-step loss is defined as $l_i = -\log_2 \hat{p}_i(b_i^*|a)$, and the confidence is based on the average deviation of the actual loss from the expected optimal loss in n steps, re-scaled with respect to the minimal and maximal possible deviation value [4]. To implement a simple forgetting scheme, temporal decay with a decay rate of $\lambda \in [0,1]$ is introduced to make the recent loss values matter more than the old ones [5,6,4]:

$$
\begin{aligned}
C_{log}^{\lambda} &= 2^{-|\frac{\Delta_n^{\lambda}}{\Delta_n^{max,\lambda} - \Delta_n^{min,\lambda}}|} \\
\text{where:} \quad \Delta_n^{\lambda} &= M_{\lambda}\big(-\log_2 \hat{p}_i(b_i^*|a) + \sum_b \hat{p}_i(b|a) \log_2 \hat{p}_i(b|a) \big)_{i=1..n} \\
\Delta_n^{max,\lambda} - \Delta_n^{min,\lambda} &= \max_{(b_1^*..b_n^*)} \{-M_{\lambda}(\log_2 \hat{p}_i(b_i^*|a))\} - \min_{(b_1^*..b_n^*)} \{-M_{\lambda}(\log_2 \hat{p}_i(b_i^*|a))\} \\
&= M_{\lambda}\big(\log_2 \frac{\max_b \hat{p}_i(b|a)}{\min_b \hat{p}_i(b|a)} \big)_{i=1..n} \\
M_{\lambda}(X_{i=1,\ldots,n}) &= \frac{\sum_{i=1}^{n} \lambda^{n-i} X_i}{\sum_{i=1}^{n} \lambda^{n-i}}
\end{aligned}
$$

It is easy to prove that if $\hat{p}_i(\cdot|a)$ are frequency distributions and $\hat{p}_1(\cdot|a)$ is uniform, then for every $n > 1$: C_{log}^{λ} is defined and $0.5 \leq C_{log}^{\lambda} \leq 1$. C_{log}^{λ} helps to detect changes in the user's policy, but it's unreliable when the number observations is small. This disadvantage can be tackled, though, with a very simple (but efficient) measure: $C_{Wang} = n/(n+1)$ [9,2]. Now, the agent is confident in her knowledge if she has enough data *and* she detects no irregularities in the user's behavior: $\mathbf{C} = \min(\mathbf{C_{log}^{\lambda}}, \mathbf{C_{Wang}})$. The decay rate λ was set to 0.9 throughout the experiments.

3 The Simulations

To investigate performance of the hybrid agent, several series of experiments were run. The agent played with various kinds of simulated 'users', i.e. processes displaying different dynamics and randomness. Those included:

- static (or stationary) 0-level user with a random policy,
- 'linear' user: a dynamic 0-level agent with the initial and the final preferences p_0, p_{100} generated at random, and the rest evolving in a linear way: $p_i = p_0 + (p_{100} - p_0)/100$,
- 'stepping' user: same as the 'linear' one except that the preferences change after every 30 steps: $p_i(b) = p_0(b) + (i \text{ div } 30)(p_{100}(b) - p_0(b))/3$,
- 'cheater': a user that chooses the action 'cheat' with probability of 1.0,
- 'malicious': an adversary 0-level user with a stationary random policy for the first 30 rounds, then switching to the 'cheater' policy.

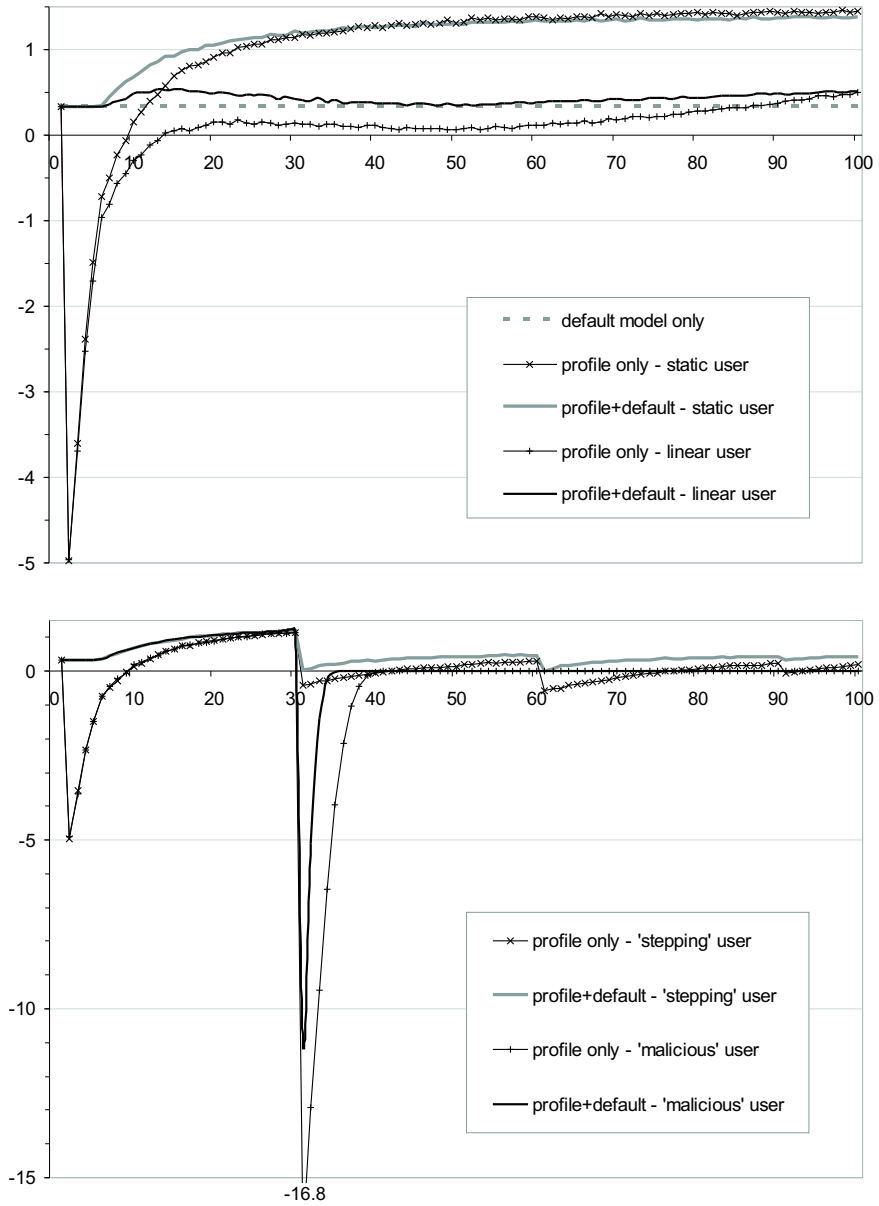

Fig. 2. Hybrid agents vs. single-model agents: the average payoffs for single-minded users

1000000 independent random interactions (a sequence of 100 rounds each) have been simulated for every particular setting; the average results are presented in the following subsections.

3.1 Playing Against Single-Minded Users

The user has been assumed rather simple-minded in the first series of experiments, in order to get rid of the exploration/exploitation tradeoff. Thus, it has been assumed that the user's response doesn't depend on the actual offer being made: $p(cheat)$, $p(accept)$ and $p(skip)$ are the same regardless of the offer (if he's dishonest, he cheats for a small reward as well as a big one, for instance). In consequence, no specific exploration strategy is necessary – every action the agent can choose will reveal exactly the same about the user's policy – so the agent can just maximize $eval(a)$ when making her decisions. The results for various types of users are presented on figure 2. The hybrid agent is almost never worse than the agent using only the user's profile (even for the static user), and in the most risky moments she plays much safer than the latter. At the same time, she has the potential to play positively better than the 'default model' agent. Some simulations were also run for a modified version of the banking game (representing a situation in which the agent's decisions involve less risk) with similar results (see figure 3).

The 'single-mindedness' assumption looks like a rough simplification. On the other hand, the preferences of a particular user (with respect to different offers) are hardly uncorrelated in the real world. For most human agents the situation seems to be somewhere between both extremes: if the user tends to cheat, he may cheat in many cases (although not all by any means); if the user is generally honest, he'll rather not cheat (although the temptation can be too strong if the reward for cheating is very high). Therefore the assumption that the user has the same policy for all the agent's offers may be also seen as the simplest way of collaborative modeling [10]. Section 3.3 gives some more rationale for this kind of assumption, while in the next section users with multi-dimensional (uncorrelated) policies are studied to complete the picture.

3.2 Experiments for Users with More Complex Policies

In this section users are simulated with no restriction on the relation between their conditional policies $p(\cdot|\text{safe})$, $p(\cdot|\text{normal})$ and $p(\cdot|\text{risky})$. Boltzmann exploration strategy is used to deal with the exploration-exploitation problem: the agent chooses action a with probability $P(a) = e^{eval(a)/T} / \sum_{a'} e^{eval(a')/T}$ [1]. As the possible rewards span a relatively large interval (we are using the first payoff table again), the initial temperature parameter is relatively high: $T_0 = 100$, and the decay factor is 0.8. Thus $T_i = T_0 * (0.8)^i$. The results on figure 4 show that the double-model agent has some problems with efficient exploration – in consequence, she plays *too* safe against a stationary user. On the other hand, she is much better protected from sudden changes in the user's behavior. Moreover, the double-model agent plays much better against a 'cheater': she loses 86.9 less than the profile-based agent in the first 15 steps (after that both agents fare almost the same).

	acpt.	cht.	skip
risky	30	-30	0.5
norm.	10	-9	0.5
safe	0.5	0	0.5

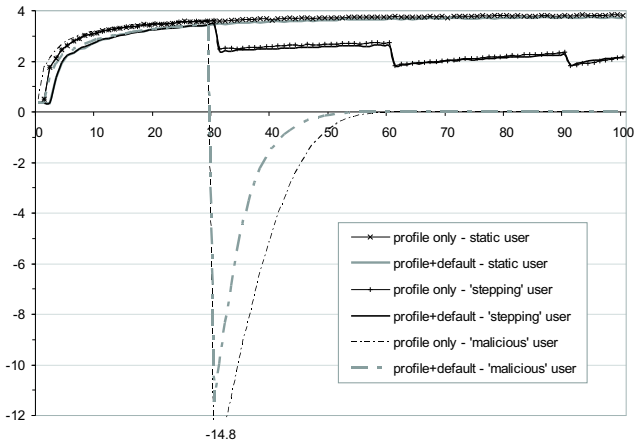

Fig. 3. Results for the modified game

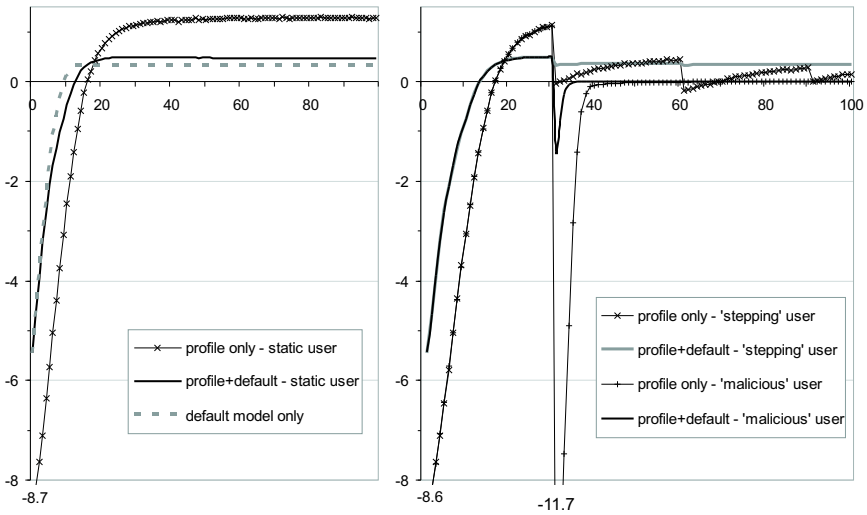

Fig. 4. Playing against non-singleminded users

3.3 Matrix Games with No Pure Equilibrium

Let us go back to section 3.1 and to the assumption that the user's response doesn't depend on the actual action from the agent. Note that the assumption makes perfect sense when the user simply cannot know the agent's action in advance. This is the case, for instance, when the negotiation process is longer and consists of multiple steps, or when some hidden policy of the bank is concerned (instead of particular 'offers'). The game is a matrix game then, and the 'default' strategy pair `(safe offer,cheat)` is the maxmin equilibrium [8].

The games from section 3.1 are somewhat special since they have their equilibria within the set of pure strategies (i.e. single decisions of the agents). For most matrix games that's not the case. However, every game has its maxmin equilibrium within the set of *mixed* strategies (i.e. probabilistic policies). The set is infinite, but in principle only a finite subset really matters: if the agent can guess the opponent's current (mixed) strategy approximately, then there is a pure strategy with the best expected payoff; otherwise, the agent should choose her maxmin. An example of such game is presented below. The agent's maxmin strategy for this game is $S_D = [0.4, 0.4, 0.2]$. If any of the players plays his/her maxmin, the expected output of the game is 0.

	b1	b2	b3
a1	-1	2	0
a2	0	-2	5
a3	2	0	-10

Note that in the case of mixed strategies, the strategies can be combined directly instead of combining the evaluations. Thus in the experiments the hybrid agent has been choosing the strategy $S = C\ S_{profile} + (1 - C)\ S_D$ where $S_{profile}$ is the strategy with the best estimated payoff. A different way of decision making calls for a modified confidence measure: the confidence is now $C' = C$ for $C \leq 0.4$, and $C' = \max(0.4, 3C - 1.9)$ otherwise. The results (figure 5) reveal that the hybrid agent is again too cautious when the user is random and stationary. However, the bottom line in the game is drawn by a user who can guess the agent's current strategy S somehow (it must a 2-level agent rather than 0-level, since the banking agent is a 1-level one). The 'malicious' user here is defined this way: he uses a random policy for the first 30 steps, and after that starts choosing the most dangerous action (the one with the minimal payoff), 'guessing' the agent's strategy in advance. Playing against a user who chooses the most dangerous action all the time, the hybrid agent was 93.6 better off than the profile-based agent after the first 50 steps.

4 Conclusions

The experiments presented in this paper suggest that a software agent can combine machine learning with Game Theory solutions to display more profitable (or at least safer) performance in many cases. The confidence measure used here is not perfect, and it shows in the results of the simulations. Further experiments should include also agents using more sophisticated learning methods.

References

1. B. Banerjee, R.Mukherjee, and S. Sen. Learning mutual trust. In *Working Notes of AGENTS-00 Workshop on Deception, Fraud and Trust in Agent Societies*, pages 9–14, 2000.

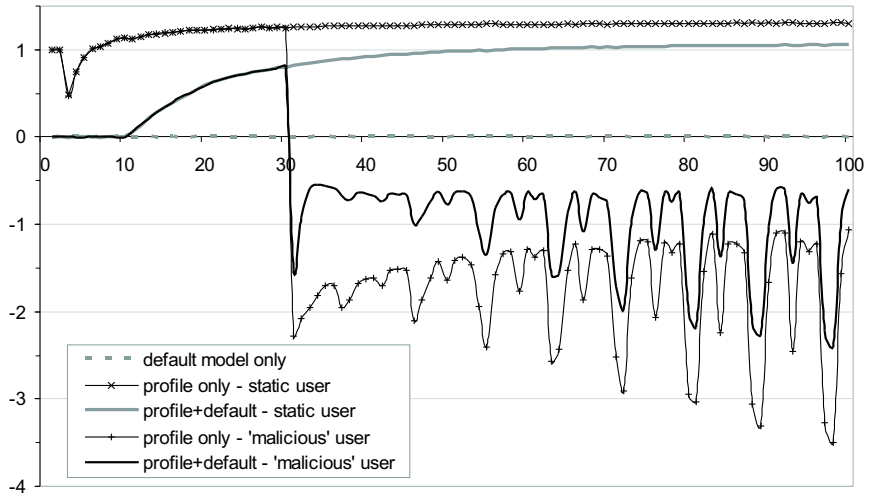

Fig. 5. Results for the matrix game: static user, 'malicious' user

2. W. Jamroga. Datasize-based confidence measure for a learning agent. In *Benelearn 2002*, 2002.
3. W. Jamroga. Multiple models of reality and how to use them. In H. Blockeel and M. Denecker, editors, *BNAIC 2002*, pages 155–162, 2002.
4. W. Jamroga. A confidence measure for learning probabilistic knowledge in a dynamic environment. In *Proceedings of International Conference on Computational Intelligence for Modelling Control and Automation*, 2003.
5. I. Koychev. Gradual forgetting for adaptation to concept drift. In *Proceedings of ECAI 2000 Workshop "Current Issues in Spatio-Temporal Reasoning"*, pages 101–106, 2000.
6. S. Kumar. Confidence based dual reinforcement q-routing: an on-line adaptive network routing algorithm. Master's thesis, Department of Computer Sciences, The University of Texas at Austin, 1998. Tech. Report AI98-267.
7. N. Merhav and M. Feder. Universal prediction. *IEEE Trans. Inform. Theory*, IT-44(6):2124–2147, 1998.
8. J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behaviour*. Princeton University Press: Princeton, NJ, 1944.
9. P. Wang. Confidence as higher-order uncertainty. In *Proceedings of the Second International Symposium on Imprecise Probabilities and Their Applications*, pages 352–361, 2001.
10. I. Zukerman and D.W. Albrecht. Predictive statistical models for user modeling. *User Modeling and User-Adapted Interaction*, 11:5–18, 2001.