Confidence Measure for a Learning Agent

Wojciech Jamroga
Parlevink Group, University of Twente, Netherlands
Institute of Mathematics, University of Gdansk, Poland
jamroga@cs.utwente.nl

Abstract. This paper reports a research aimed to design a confidence measure for an agent who learns and uses probabilistic models of other agents' behavior. If the agent has several alternative models of a particular opponent, she can use the confidence values to combine the models or choose among them. The measure has been inspired by the research on universal prediction, and based on the self-information loss function. It was verified through some simple experiments with simulated software agents.

Keywords: multiagent systems, meta-uncertainty, confidence, machine learning, user modeling, self-information loss function.

1 Introduction

An agent may benefit from keeping several alternative models of the reality in certain situations – the point has been advocated in [1]. If a software agent is designed to interact with users, she can be obviously better off keeping the users' profiles to approximate the actual preferences of each user. However, when the identity of a user remains unknown or the user is completely new to the system, an average user model or a default model may be used instead. While a standard machine learning algorithm will assume some arbitrary initial model of such a user (via uniform or random distribution, for instance), it should be clear that such knowledge mustn't be trusted when it comes to decision making, since the model is not supported by any data so far. Moreover, users' preferences may evolve, and even worse: some users may assume someone else's identity (incidentally or on purpose). This calls for a kind of self-reflection on the agent's part: a confidence measure is needed to determine to which extent every piece of knowledge can be considered reliable. If we provide the agent with such a measure, she can base her decisions on the most reliable model, or use a linear combination of all the appropriate models.

In this paper a confidence measure is proposed for an agent, interacting with other agents (users) in a very simple environment. The agent is meant to employ a kind of meta-reasoning to determine the level of reliability of the resulting knowledge. The aim of the confidence measure is to represent meta-(un)certainty – thus the actual confidence values range from 0 (complete distrust) to 1 (full confidence). Some researchers from the probability theory community suggest that – to solve the problem – we should take the agent's probabilistic knowledge as a random quantity, and use its variance as a clue [11, 6]. The suggestion was followed in [2], with rather negative results. Another possibility explored in that paper was based on self-information loss function (or log-loss function), used widely in the fields of

information theory and universal prediction [9]. As the latter idea proved promising, we will explore it further in this paper.

Confidence has been recognized an important and useful notion within the Machine Learning community. Explicit representing and analyzing confidence in the knowledge or beliefs being constructed has been successfully used in the areas of movement recognition [15], speech recognition ([8, 18] and many more) or in mobile user location [7], for instance. The confidence measure proposed here comes perhaps closest to the measure proposed by Wang [14], based on the amount of data available to the agent. The time flow and the resulting devaluation of the old data and/or knowledge have also been focus of several papers. Kumar [5] uses a confidence measure to improve a Q-learning based algorithm for adaptive network routing. The measure is very simple – the confidence in every Q-value which hasn't been updated in the last step is subject to 'time decay': $C_{new}(x) = \lambda\, C_{old}(x)$, where $\lambda \in (0, 1)$ is the decay constant. A similar idea was introduced in [4] to track the user's drifting interests effectively.

2 Motivation: Multilevel User Modeling

The motivation behind the confidence was 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. 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 [1]. Another way is to combine strategies directly – we can do it if we treat them as mixed (probabilistic) ones [3].

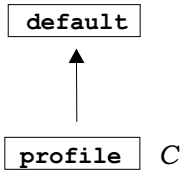

Figure 1: The simplest hierarchy: two models of reality

Here, the agent utilizes two models of the user: the profile (for which confidence C is being computed after every step of interaction), and the default model defined in Game Theory fashion: the user is assumed an enemy who always chooses the optimal response. The hybrid agent from Figure 1 can make her decisions two ways:

1. **combining evaluations:** evaluation of every action a is based on sub-evaluations derived from both models separately: $eval(a) = C\ eval_{\textit{profile}}(a) + (1 - C)\ eval_{\textit{default}}(a)$. The agent chooses a with maximal $eval(a)$;

2. **combining strategies:** if $S_{\textit{profile}}$ is the strategy that maximizes $eval_{\textit{profile}}(a)$, and S_{maxmin} is the agent's maxmin strategy (i.e. the safest possible decision), then $S = C\ S_{\textit{profile}} + (1 - C)\ S_{maxmin}$ is chosen (i.e. "choose the strategy based on the profile with probability C, and the maxmin strategy otherwise").

A confidence measure for such agents is proposed in Section 3. Some results of experiments with simulated agents can be found in Section 4.

3 Detecting Changes of the Pattern

There are roughly two possible sources of doubt. First, the agent may have too little data. This shows the need for a confidence measure in an obvious way: when the agent starts interaction with a completely new user, her knowledge about the user is virtually none. However, the model of the user is utilized in the same way by most algorithms – regardless of the number of learning steps that have been taken – although the model is more or less random when we have, say, two user's responses as the whole data set. A measure for tackling this kind of doubt was proposed in [14]: $C_{Wang} = n/(n+k)$, where n is the amount of data and k is an arbitrary fixed number. For instance, $C_{Wang} = n/(n+1)$ for $k = 1$. It seems simple and rather ad hoc, but works surprisingly well – even in the experiments conducted within this study (section 4).

Next, the environment might have changed considerably, so the data do not reflect its current shape. The agent can certainly benefit from detecting conspicuous changes of pattern in the user's behavior, and acting more cautiously in such situations. The latter kind of doubt is the focus of this paper.

3.1 Confidence Based on the Logarithmic Loss Function

Log-loss function is used in the research on machine learning and time series prediction – especially universal prediction, where a prediction doesn't necessarily have to be a simple estimation of the next observation, but it can be a complex structure (a probability assessment, a strategy etc.), and the real underlying structure (the 'source') generating the events is assumed to be unknown [9]. The universal prediction methods focus on finding a good predictor, not on assessing *how* good it is, though. A way of transforming the log-loss values into confidence values is proposed in this section.

Let $\hat{p}_i$ represent the agent's beliefs about the preferences of the user in response to a particular action from the agent or a particular state of the environment (at the ith step of interaction within this context). Let b_i^* be the user's actual response at that step. One-step loss l_i and the average loss in n steps L_n can be defined using the log-loss function:

$$\begin{aligned} l_i &= logloss(\hat{p}_i, b_i^*) = -\log_2 \hat{p}_i(b_i^*) \\ L_n &= \frac{1}{n}\sum_{i=1}^{n} l_i = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \hat{p}_i(b_i^*) \end{aligned}$$

Note that the expected value of l is a function of two probability distributions: the real distribution p (the 'source' distribution), and its model $\hat{p}$ built by the learning agent. More formally, $E\, l = -\sum_b p(b) \log_2 \hat{p}(b) = El(p, \hat{p})$. The loss is minimal (in the sense of expected value) when the agent has been guessing correctly, i.e. when the model she used was a true reflection of the reality: $\hat{p} = p$ [9]. However, this holds only if we assume that p is fixed, and not in general, as the following example demonstrates.

Example 1. *Consider an agent who estimates the user's policy with probability distribution* $\hat{p} = P1$ *at some moment (see Figure 2). If her guess is right (i.e. the user's policy is* $p = P1$

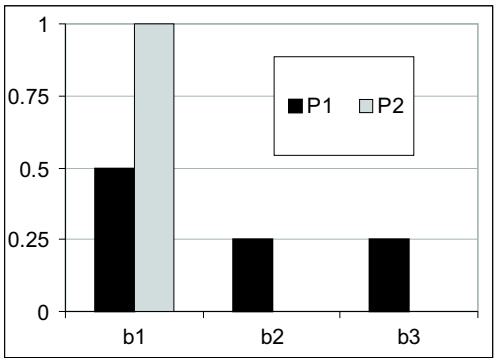

Figure 2: Minimal vs. optimal loss – an example

indeed), the expected loss is $E\ l = -0.5 \log_2 0.5 - 2 \cdot 0.25 \log_2 0.25 = 1.5$. *Yet if the real policy is* $p = P2$*, then* $E\ l = -1 \log_2 0.5 = 1$*: the agent's loss can be smaller when her guess is wrong! In other words,* $El(p, \hat{p})$ *has a global minimum of* $\hat{p} = p$ *for a fixed* p*, but not when we consider all the possible source distributions.*

Note that this is not a problem in time series prediction. The source distribution is presumably fixed in a single interaction (there is one *objective* source distribution), and hence $El(p, \hat{p})$ is *objectively* minimal for $\hat{p} = p$ (out of all the objectively possible values of $E\ l$). As long as the agent is not interested in the loss values themselves – only in finding the minimum point – minimizing the mean L_n is a valid strategy for her to find a model $\hat{p}$ that approximates the true probability distribution p. However, when the source distribution is unknown, some smaller loss values may be deemed possible from the agent's subjective point of view. Moreover, she may experience a smaller loss in a subsequent interaction in which her beliefs would be actually farther from the reality.

Example 2. *Consider the learning agent from Example 1 again. Suppose the agent computes her* disconfidence *in her own model of the reality as a value somehow proportional to* L_n*. Suppose that she interacts with two users, and in both cases she gets* $\hat{p} = P1$*. Moreover, let the real policy of the first user be* $p = P1$*, and the second:* $p' = P2$*. In a long run, our agent is going to obtain average loss of* 1.5 *in the first case, and* 1 *in the second. Thus, she is going to trust her beliefs more in the latter (where she actually guessed the policy incorrectly) – which is unacceptable. In consequence, the minimal loss is not the optimal loss in this case.*

What does 'optimal' mean then? Let us define the optimal series of models as a sequence of the true probability distributions: $\hat{p}_i = p_i$ for $i = 1..n$. Now the optimal expected loss is the expected value of the average loss we get *provided our actual sequence* $\hat{p}_1..\hat{p}_n$ *is optimal*:

$$\begin{aligned} Opt_n &= EL_n = -\sum_{(b_1..b_n)} \big[p(b_1..b_n)\frac{1}{n}\sum_{i=1}^{n} \log_2 \hat{p}_i(b_i)\big] = \\ &= -\frac{1}{n}\sum_{i=1}^{n}\sum_{b_i} p_i(b_i)\ \log_2 \hat{p}_i(b_i) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{b} \hat{p}_i(b)\ \log_2 \hat{p}_i(b) \end{aligned}$$

Now, the loss deviation (or rather its absolute value) seems a better basis for the confidence:

$$\Delta_n = L_n - Opt_n = -\frac{1}{n}\sum_{i=1}^{n} \big[\log_2 \hat{p}_i(b_i^*) - \sum_{b} \hat{p}_i(b)\ \log_2 \hat{p}_i(b)\big]$$

As different $\hat{p}$'s give different loss characteristics, they also define very different deviation intervals. For $\hat{p}_i = P2, i = 1..n$, for instance, the only possible values for Δ_n are 0 and ∞ – if the model has proved to be even slightly mistaken, then Δ_n will remain ∞ forever. On the other end of the scale it's easy to observe that if the agent stubbornly keeps the uniform distribution as the user's model (i.e. $\hat{p}(b) = \frac{1}{|ActB|}$ all the time), then the deviation Δ_n is always 0, regardless of the actual responses from the user! In both cases the value of Δ_n tells virtually nothing about the actual reliability of $\hat{p}$. It would be desirable that our confidence measure produced more appropriate values, or at least 'signal' such situations instead of giving unreliable output.

Between both extremes the range of possible Δ_n also vary: it is close to $(0, \infty)$ for very unbalanced models, and very narrow when $\hat{p}$ is close to the uniform distribution. It is proposed here that we can normalize the loss deviation with its range ($\Delta_n^{max} - \Delta_n^{min}$) to obtain disconfidence value that does not depend on the actual models $\hat{p}$ so much. Now the log-loss-based confidence measure can be defined as:

$$C_{log} = 1 - |\frac{\Delta_n}{\Delta_n^{max} - \Delta_n^{min}}|$$

where

$$\begin{aligned}
\Delta_n^{max} &= \max_{(b_1^*..b_n^*)} \{\Delta_n\} = \max_{(b_1^*..b_n^*)} \{L_n - Opt_n\} = \max_{(b_1^*..b_n^*)} \{L_n\} - Opt_n \\
\Delta_n^{min} &= \min_{(b_1^*..b_n^*)} \{\Delta_n\} = \min_{(b_1^*..b_n^*)} \{L_n - Opt_n\} = \min_{(b_1^*..b_n^*)} \{L_n\} - Opt_n \qquad \text{and} \\
\Delta_n^{max} - \Delta_n^{min} &= \max_{(b_1..b_n)} \{-\frac{1}{n}\sum_{i=1}^{n} \log_2 \hat{p}_i(b_i)\} - \min_{(b_1..b_n)} \{-\frac{1}{n}\sum_{i=1}^{n} \log_2 \hat{p}_i(b_i)\} \\
&= -\frac{1}{n}\sum_{i=1}^{n} \min_{b_i} \log_2 \hat{p}_i(b_i) + \frac{1}{n}\sum_{i=1}^{n} \max_{b_i} \log_2 \hat{p}_i(b_i) \\
&= \frac{1}{n}\sum_{i=1}^{n} \big[\log_2 \max_b \hat{p}_i(b) - \log_2 \min_b \hat{p}_i(b)\big] = \frac{1}{n}\sum_{i=1}^{n} \log_2 \frac{\max_b \hat{p}_i(b)}{\min_b \hat{p}_i(b)}
\end{aligned}$$

The measure has the following properties:

- n Δ_n and n $(\Delta_n^{max} - \Delta_n^{min})$ can be computed incrementally – the agent doesn't have to keep any additional information;
- if the value of C_{log} can be computed, then $0 \leq C_{log} \leq 1$;
- C_{log} is undefined exactly in the two cases where Δ_n is most dubious: when $\hat{p}_i$'s are uniform for all $i = 1..n$ or when there exist i and b such that $\hat{p}_i(b) = 0$. Note also that, when $\hat{p}_i$ are frequency distributions (more generally: probability distributions obtained through Bayesian updating), the first situation can happen only at the very beginning of the interaction, i.e. for $i = 1$. Moreover, the agent can be prevented from the latter situation by starting from an initial distribution such that $\hat{p}_1(b) > 0$ for every b (for instance, she may use the uniform rather than nil distribution as the starting point). Then we make sure that the probabilities will always be positive.

3.2 Log-loss Confidence with Temporal Decay

To implement a simple forgetting scheme, the idea of the decay rate $\lambda \in [0,1]$ is used in this paper: the older items in a time series are supposed to decay with every step.

Let the **sample mean with decay** be defined as follows:

$$M_\lambda(X_{i=1,...,n}) = \frac{X_1\lambda^{n-1} + X_2\lambda^{n-2} + ... + X_{n-1}\lambda + X_n}{\lambda^{n-1} + \lambda^{n-2} + ... + \lambda + 1} = \frac{\sum_{i=1}^n \lambda^{n-i} X_i}{\sum_{i=1}^n \lambda^{n-i}}$$

for a series of n items and a temporal decay rate λ. Note that M_λ can also be computed incrementally:

$$M_\lambda(X_{i=1..n}) = \frac{M_\lambda(X_{i=1..n-1}) \cdot (1-\lambda^{n-1})\lambda + (1-\lambda)X_n}{1-\lambda^n}$$

Temporal decay can be introduced into the log-loss confidence to make the recent loss values matter more than the old ones – we can redefine L_n to be a mean with decay:

$$L_n^\lambda = M_\lambda(l_{i=1..n}) = \frac{-\sum_{i=1}^n \lambda^{n-i} \log_2 \hat{p}_i(b_i^*)}{\sum_{i=1}^n \lambda^{n-i}}$$

Then:

$$\begin{aligned}
\Delta_n^\lambda &= L_n^\lambda - Opt_n^\lambda = \frac{\sum_{i=1}^n \lambda^{n-i}[-\log_2 \hat{p}_i(b_i^*) + \sum_b \hat{p}_i(b)\log_2 \hat{p}_i(b)]}{\sum_{i=1}^n \lambda^{n-i}} = \\
&= -M_\lambda\big(\log_2 \hat{p}_i(b_i^*)\big)_{i=1..n} + M_\lambda\big(\sum_b \hat{p}_i(b)\log_2 \hat{p}_i(b)\big)_{i=1..n} \\
\Delta_n^{max,\lambda} - \Delta_n^{min,\lambda} &= \max_{(b_1^*..b_n^*)}\{-M_\lambda(\log_2 \hat{p}_i(b_i^*))\} - \min_{(b_1^*..b_n^*)}\{-M_\lambda(\log_2 \hat{p}_i(b_i^*))\} = \\
&= \frac{-\sum_{i=1}^n \lambda^{n-i}\log_2 \min_{b_i^*} \hat{p}_i(b_i^*)}{\sum_{i=1}^n \lambda^{n-i}} + \frac{\sum_{i=1}^n \lambda^{n-i}\log_2 \max_{b_i^*} \hat{p}_i(b_i^*)}{\sum_{i=1}^n \lambda^{n-i}} = \\
&= \frac{\sum_{i=1}^n \lambda^{n-i}\log_2[\max_b \hat{p}_i(b)/\min_b \hat{p}_i(b)]}{\sum_{i=1}^n \lambda^{n-i}} = M_\lambda\big(\log_2 \frac{\max_b \hat{p}_i(b)}{\min_b \hat{p}_i(b)}\big)
\end{aligned}$$

Again,

$$C_{log}^\lambda = 1 - |\frac{\Delta_n^\lambda}{\Delta_n^{max,\lambda} - \Delta_n^{min,\lambda}}|$$

and C_{log}^λ retains the properties of C_{log}.

4 Experiments

A number of simulations were run in order to verify the confidence measure.

4.1 Online Banking Scenario

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. The banking agent

is an adaptive 1-level agent, i.e. an agent that models other agents as 0-level agents (agents whose behavior can be described with a probabilistic policy [12]) and uses the models in the decision-making process. The user is simulated as a 0-level agent. The agent estimates the user's policy p with a probability distribution $\hat{p}$, computed through simple Bayesian updating [10]:

$$\hat{p}(b) \leftarrow \begin{cases} \frac{\hat{p}(b)n+1}{n+1} & \text{if } b = b^* \\ \frac{\hat{p}(b)n}{n+1} & \text{if } b \neq b^* \end{cases} \quad ; \qquad n \leftarrow n+1$$

where b^* is the actual response from the user in the last round of interaction. Value $n_0 \geq 0$ is the number of "virtual" training examples. Initial distribution $\hat{p}_0$ is uniform in most experiments, although the "safe" distribution (corresponding to the maxmin strategy) has also been tried.

Adaptive user models are often useful when the domain is cooperative or neutral, or when the adversaries are generally weaker than 'our' agent. Classical Game Theory solutions [13] 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. In that case the agent can use her maxmin strategy to play things safe, or choose a more sophisticated behavior in the way described in section 2.

4.2 The Game

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 agent plays with various kinds of simulated 'users', i.e. processes displaying different dynamics and randomness. Those include:

- static (or stationary) 0-level user with a random policy: $p_0 = \cdots = p_{100}$ is generated at random at the beginning of each interaction,
- 'stepping' user: a dynamic 0-level agent with the initial and the final preferences p_0, p_{100} generated at random; his policy changes every 30 steps: $p_i(b) = p_0(b) + (i \text{ div } 30)(p_{100}(b) - p_0(b))/3$,
- 'malicious': an adversary 0-level user with a stationary random policy for the first 30 rounds, after which he cheats all the time.

1000000 independent random interactions (a sequence of 100 rounds each) have been simulated for every particular setting; the results (averaged for every round separately) are presented in section 4.3.

The user has been assumed rather simple-minded, in order to get rid of the exploration/exploitation tradeoff. Thus, it was 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. 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 [19].

4.3 Results

Figure 3 shows how the confidence values evolve against a 'stepping' user. The confidence values are compared against the expected absolute deviation of the learned profile from the real policy of the user: $expdev = \sum_b |\hat{p}(b) - p(b))| \cdot p(b)$, or rather the 'accurateness' of the profile, i.e. $1 - expdev$. It can be observed that the log-loss-based measure is able to detect changes in the user's behavior – especially when temporal decay is employed. Wang's measure, designed for static sources of data, increases always in the same manner regardless of the users' responses.

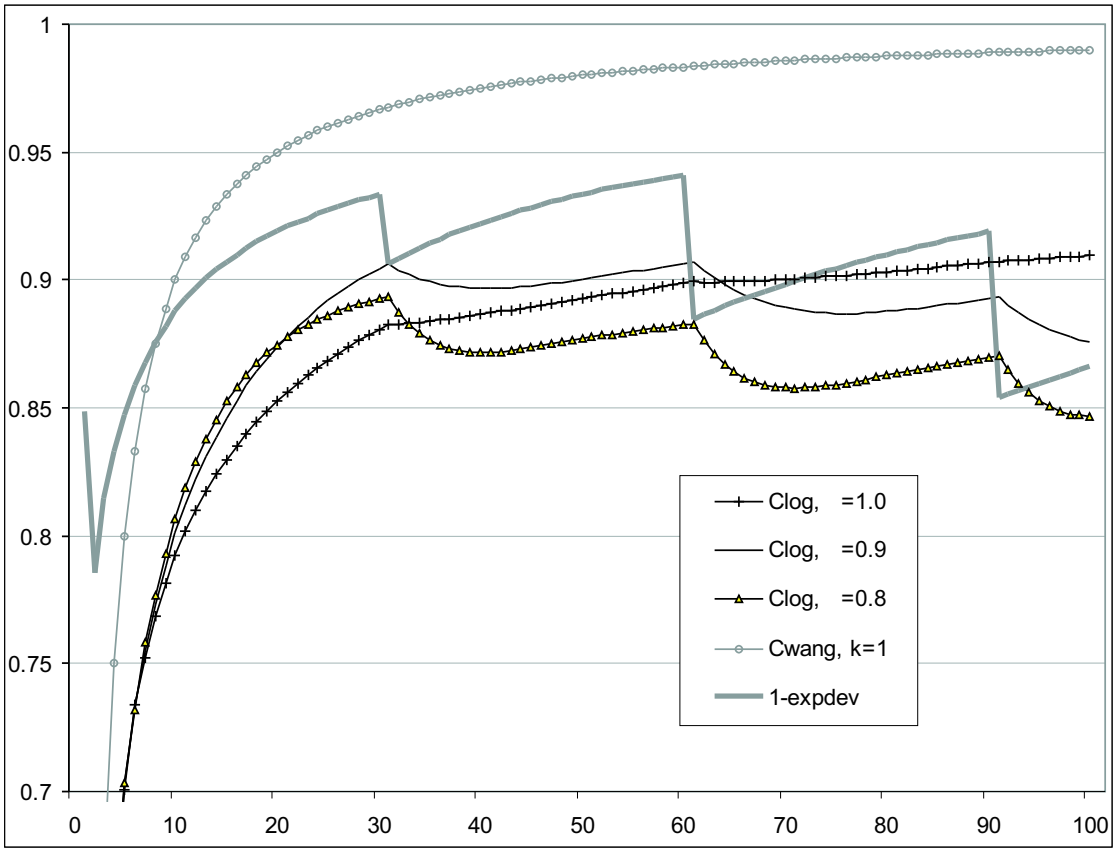

Figure 3: Confidence values: C_{log} vs. C_{Wang} vs. accurateness

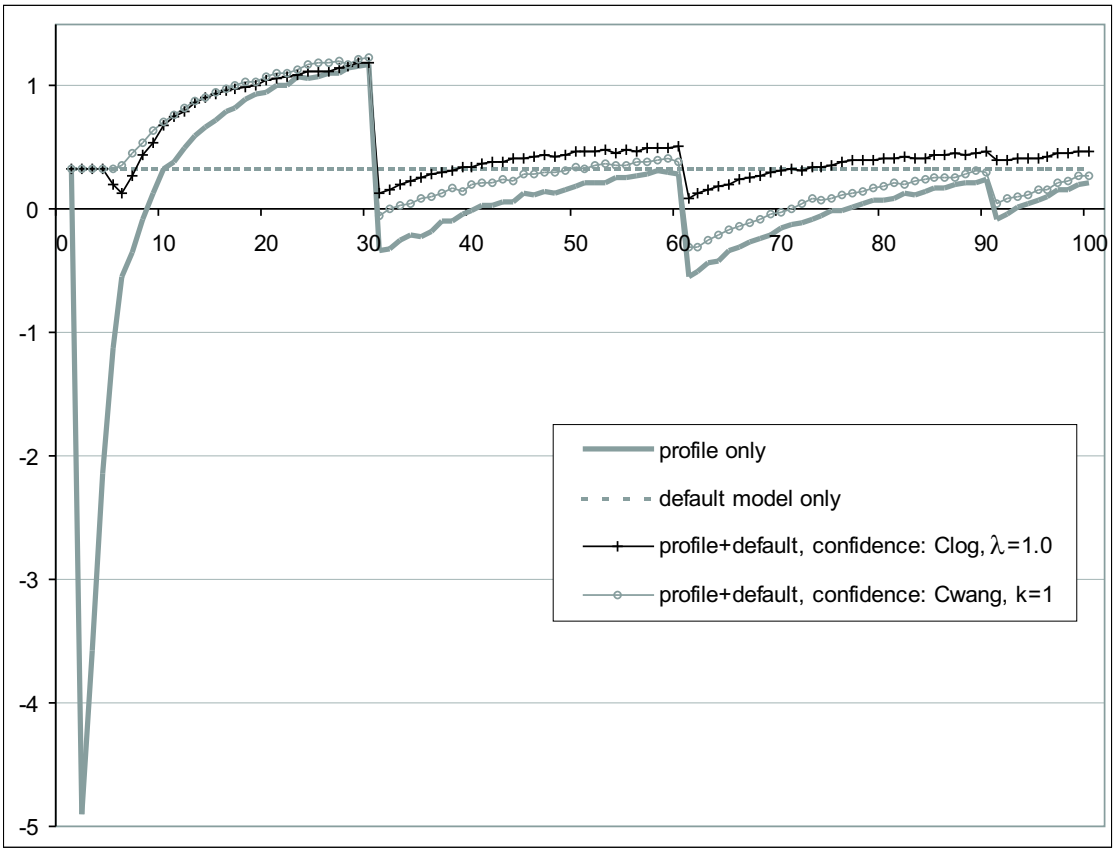

Figure 4: Hybrid agents vs. single-model agents: payoffs against 'stepping' users

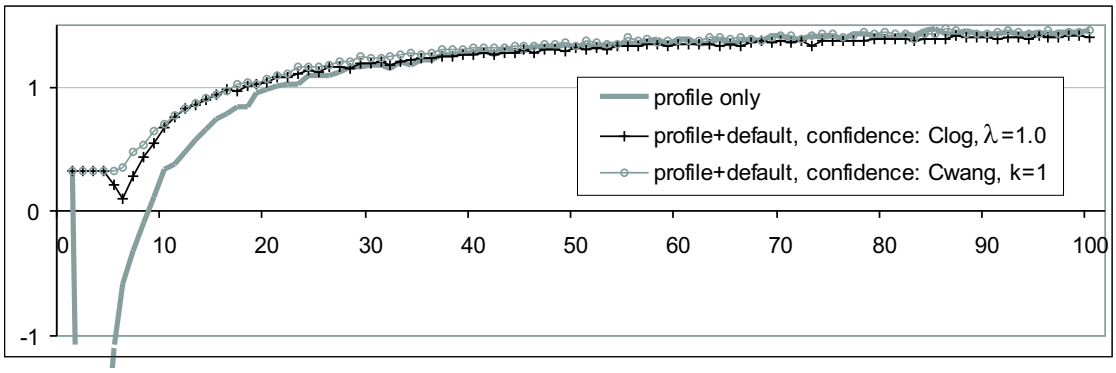

Figure 5: Hybrid agents vs. single-model agents: payoffs against stationary users

Figures 4, 5 and 6 show that an agent using such a hybrid model of the reality can be better off than an agent using either the profiles or the default user model alone. Such a 'multi-model' agent doesn't lose so much money at the beginning of an interaction (because the confidence is low and therefore she's using mostly the default model). On the other hand, the confidence is almost 1 by the time the acquired knowledge becomes more accurate so the agent can start using the user profile successfully. In most cases there were no significant differences in the output of the agents using C_{log} with or without temporal decay. Only against 'malicious' users smaller λ proves safer, since it makes the agent react faster.

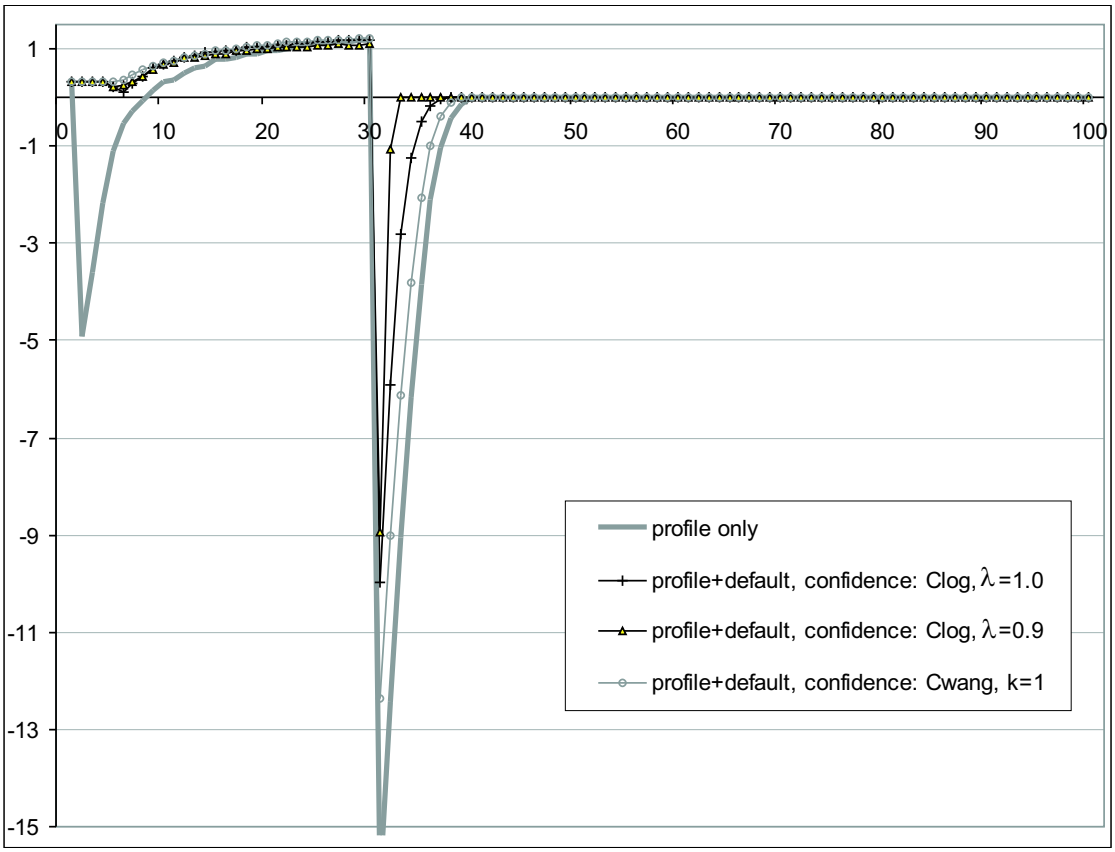

Figure 6: Hybrid agents vs. single-model agents: payoffs against 'malicious' users

All the above results were obtained for combining *evaluations*. Combining strategies directly, on the other hand, didn't prove successful in this setting – as Figure 7 shows. It seems that either the latter decision-making scheme isn't suitable for the 'banking game', or the confidence measure being used should be different in this case.

5 Conclusions

The experiments showed that a confidence measure can be useful – at least in some settings – for instance, to detect changes in a user's behavior, or as a means for weighting alternative beliefs. A confidence measure, based on logarithmic loss function was investigated with encouraging results. The loss function provides a direct link between the model and the new observations, and the temporal decay scheme lets the agent focus more on the results of recent predictions rather than all of them. In consequence, the measure is flexible enough to react appropriately even after many steps of collecting and analyzing data.

The author would like to thank Mannes Poel for the discussions and all his suggestions.

References

[1] W. Jamroga. Multiple models of reality and how to use them. In H. Blockeel and M. Denecker, editors, *BNAIC 2002*, pages 155–162, 2002.

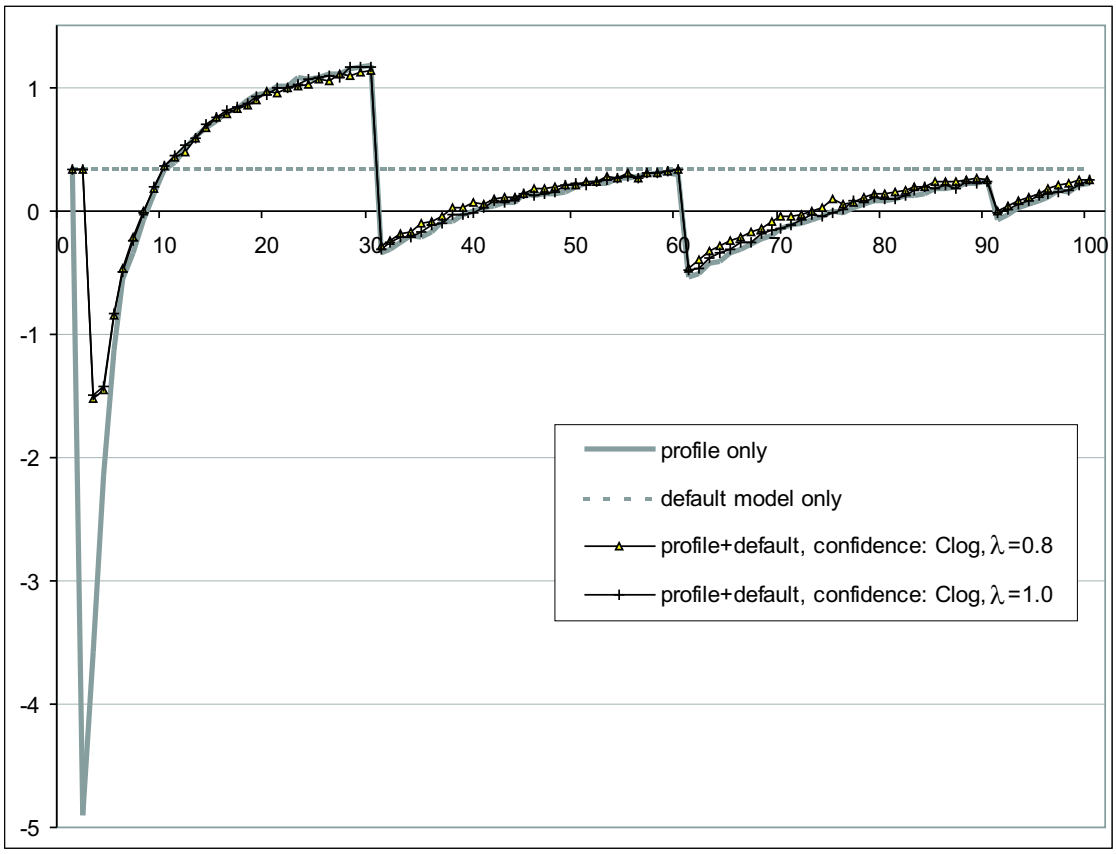

Figure 7: Hybrid agents vs. single-model agents: payoffs against 'stepping' users, combining strategies

[2] W. Jamroga. A confidence measure for learning probabilistic knowledge in a dynamic environment. In *Proceedings of International Conference on Computational Intelligence for Modeling Control and Automation*, 2003.

[3] W. Jamroga. Safer decisions against a dynamic opponent. In *Proceedings of IIPWM 2003*, Advances in Soft Computing. Springer, 2003. To appear.

[4] 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.

[5] 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.

[6] H.E. Kyburg. Higher order probabilities and intervals. *International Journal of Approximate Reasoning*, 2:195–209, 1988.

[7] Z. Lei, C.U. Saraydar, and N.B. Mandayam. Paging area optimization based on interval estimation in wireless personal communication networks. *Mobile Networks and Applications*, 5(1):85–99, 1999. Special Issue on Mobile Data Networks: Advanced Technologies and Services.

[8] E. Mengusoglu and C. Ris. Use of acoustic prior information for confidence measure in ASR applications. In *Eurospeech 2001*, 2001.

[9] N. Merhav and M. Feder. Universal prediction. *IEEE Trans. Inform. Theory*, IT-44(6):2124–2147, 1998.

[10] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[11] J. Pearl. Do we need higher-order probabilities and, if so, what do they mean? In *Uncertainty in Artificial Intelligence Workshop*, 1987.

[12] S. Sen and G. Weiss. Learning in multiagent systems. In Weiss G., editor, *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*, pages 259–298. MIT Press: Cambridge, Mass, 1999.

[13] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behaviour*. Princeton University Press: Princeton, NJ, 1944.

[14] P. Wang. Confidence as higher-order uncertainty. In *Proceedings of the Second International Symposium on Imprecise Probabilities and Their Applications*, pages 352–361, 2001.

[15] R. Wang, H.-J. Zhang, and Y.-Q. Zhang. A confidence measure based moving object extraction system built for compressed domain. Technical report, Microsoft Research, Beijing, 1999.

[16] G. Widmer. Tracking context changes through meta-learning. *Machine Learning*, 27:256–286, 1997.

[17] G. Williams. *Knowing What You Don't Know: Roles for Confidence Measures in Automatic Speech Recognition*. PhD thesis, University of Sheffield, Department of Computer Science, 1999.

[18] G. Williams and S. Renals. Confidence measures for hybrid HMM/ANN speech recognition. In *Eurospeech97*, pages 1955–1958, 1997.

[19] I. Zukerman and D.W. Albrecht. Predictive statistical models for user modeling. *User Modeling and User-Adapted Interaction*, 11:5–18, 2001.