A Confidence Measure for Learning Probabilistic Knowledge in a Dynamic Environment

Wojciech Jamroga
Parlevink Group, University of Twente, Netherlands
jamroga@cs.utwente.nl
`http://www.cs.utwente.nl/~jamroga`

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. Suggestions can be found in the literature from the field of probabilistic reasoning that such a measure can be based on the variance of a model. Thus, a confidence measure for models belonging to this class is proposed in this paper, based on some self-observation on the agent's part through computing the aggregate (decayed) variance of the learning process. As the variance-based measure turns out to have serious disadvantages, another one is proposed and studied as well. The second measure has been inspired by the research on universal prediction, and based on the self-information loss function. Both measures are verified through some simple experiments with simulated software agents.

Keywords: multiagent systems, confidence measure, machine learning, user modeling.

1 Introduction

An agent may benefit from keeping several alternative models of the reality in certain situations – the point has been advocated in [2]. 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 confidence measures are proposed for an agent, interacting with other agents (users) in a very simple environment. The agent is a 1-level agent, i.e. an agent that models other agents as stochastic agents [10]. The users are assumed to have probabilistic policies that may change dynamically. The reinforcement is known beforehand for every decision of the agent, given a response from the user, and the domain of action is stateless. The agent tries to estimate the actual policy of the user via calculating a frequency distribution, which can be further used to find the decision with the maximal expected reward. The agent uses 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 [9, 5]. Therefore the first measure being proposed is based on the aggregate variance of the model. The second measure is based on the self-information loss function (or log-loss function), used widely in the fields of information theory and universal prediction [8].

1.1 Related Research

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 [13], speech recognition ([7, 16] and many more) or in mobile user location [6] for instance. In most papers the term 'confidence' or 'confidence measure' refers to the probability that the agent's *decision* (e.g. a medical diagnosis, a classification of an image, a sequence of words assigned to a spoken text etc.) is right – i.e. it refers to the probability that a particular patient really suffers from pneumonia, that there is really a car in the middle of the picture, that the user really said "open sesame" etc. In this paper the term 'confidence' refers to a subjective property of the *beliefs* (probabilistic or not) themselves, i.e. this is our own *knowledge* in which we may be more or less confident. It may be seen in two ways: either the confidence constitute meta-knowledge or the objects on the object level are complex ones and refer to some lower-level objects as well (e.g. the possible decisions). [1] discuss several approaches to measuring the error in the context of unsupervised learning, but: (a) their 'error measures' refer to inconsistencies between the knowledge being acquired and the real state of the (supposedly static) environment, while our confidence measure should capture also the natural evolution of the dynamic reality; (b) neither the 'real state' is ever accessible for our agent nor has she multiple sample sets available, because observations made at different time points may refer to completely different states of reality. In general, however, it is a common idea that the confidence measure is a result of some posterior verification that follows the process of knowledge acquisition or decision making. Also, the confidence measure may be as well based on some factors that relate to the probability indirectly: for instance, the amount of data we have available [12] or the way the observed patterns evolve [9, 5].

The time flow and the resulting devaluation of the old data and/or knowledge have also been a focus of several papers. Kumar [4] 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 [3] to track the user's drifting interests effectively. There is no explicit confidence measure there, however; instead, a scheme for 'forgetting' old observations by an agent is proposed. Moreover – in contrast to Kumar's decaying *knowledge* – these are rather the *data* that become gradually forgotten.

Another perspective to the task of adapting to the user's dynamic behavior is offered by the research on time series prediction – especially the 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 [8]. 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 paper – and the results seem to be really promising.

2 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 by Wang [12]; it seems very ad hoc, but works surprisingly well – even in the experiments conducted within this study (section 3).

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.

2.1 Aggregate Variance with Temporal Decay and the Variance-Based Confidence

To detect the moments when the user's behavior changes considerably, some dispersion measure can be used that captures its variability. The larger the dispersion, the smaller the confidence – thus, if the dispersion estimation is $disp \in [0,1]$, the confidence measure C can be somehow proportional to $1 - disp$. If we assume that the agent's knowledge $\hat{p}_i$ approximates the user's policy p_i at various time points $i = 1..n$, we can use the variability of the knowledge (over time) to estimate the variability of the real policy. It reflects also the following intuition: if the agent has collected enough data to estimate a policy of a static user, and still she has to update the estimation considerably, the pattern must be changing somehow.

The aggregate sample variance (for all the actions $b \in ActB$ the user can choose to proceed with)

$$\sum_{b \in ActB} V(\hat{p}_{i=1..n}(b)) = \frac{1}{n} \sum_{b} \sum_{i=1}^{n} (\hat{p}_i(b) - \overline{\hat{p}_{i=1..n}(b)})^2$$

has some welcome properties: it captures the variability of the *whole process* (the last update size does not), it can be computed incrementally (aggregate sample mean deviation can not), and its value is always between 0 and 1 (aggregate sample standard deviation is not). Therefore it's the variance that will be used in this paper as the variability measure.

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:

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

for a series of n items and a temporal decay rate λ. M_λ has some standard properties of a mean (the proofs are straightforward):

$$\begin{aligned} M_\lambda(X+Y) &= M_\lambda(X) + M_\lambda(Y) && (2)\\ M_\lambda(aX) &= aM_\lambda(X) && (3) \end{aligned}$$

Moreover, the weights assigned to the data sequence satisfy the forgetting constraints by Koychev [3].

How can the decay be introduced into the confidence measure C_V? Essentially the variance is just the averaged square deviation from the mean, so the following definitions seem natural:

$$\begin{aligned} V_\lambda(\hat{p}_{i=1..n}(b)) &= M_\lambda[\hat{p}_i(b) - M_\lambda(\hat{p}_{j=1..n}(b))]^2_{i=1..n} && (4)\\ disp_\lambda &= \sum_b V_\lambda(\hat{p}_{i=1..n}(b)) && (5)\\ C_V &= (1 - disp_\lambda) && (6) \end{aligned}$$

By (2) and (3):

$$\begin{aligned} V_\lambda(\hat{p}_{i=1..n}(b)) &= M_\lambda[\hat{p}_i^2(b) - 2\hat{p}_i(b)M_\lambda(\hat{p}_{j=1..n}(b)) + M_\lambda^2(\hat{p}_{j=1..n}(b))]_{i=1..n} = \\ &= M_\lambda(\hat{p}_{i=1..n}^2(b)) - M_\lambda^2(\hat{p}_{i=1..n}(b)) \end{aligned}$$

and the decayed mean can also be computed incrementally:

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

Thus, it's sufficient for the agent to remember only the current values of $M_\lambda(\hat{p}_{i=1..n}^2(b))$, $M_\lambda(\hat{p}_{i=1..n}(b))$ and n to compute the dispersion.

Note that $disp_\lambda \geq 0$ since it's a sum of nonnegative elements. Moreover:

$$\begin{aligned} disp_\lambda &= \frac{1}{n}\sum_b \sum_{i=1}^{n} \hat{p}_i^2(b) - \frac{1}{n^2}\sum_b \big(\sum_{i=1}^{n} \hat{p}_i(b)\big)^2 = \\ &= 1 - \frac{1}{n} - \big(\frac{1}{n} - \frac{1}{n^2}\big)\sum_{i=1}^{n}\sum_{b_1 \neq b_2} \hat{p}_i(b_1)\hat{p}_i(b_2) - \frac{1}{n^2}\sum_b \sum_{i \neq j} \hat{p}_i(b)\hat{p}_j(b) \leq \\ &\leq 1 - \frac{(1+\lambda^n)(1-\lambda)}{(1+\lambda)(1-\lambda^n)} && (7) \end{aligned}$$

and the bound is tight for $n \leq |ActB|$.

Since the simulations show that the measure doesn't exactly hold to its promise (section 3), an alternative measure is proposed below.

2.2 Confidence Based on the Logarithmic Loss Function

Again, 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 [8]:

$$l_i = logloss(\hat{p}_i, b_i^*) = -\log_2 \hat{p}_i(b_i^*) \quad (8)$$

$$L_n = \frac{1}{n}\sum_{i=1}^{n} l_i = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \hat{p}_i(b_i^*) \quad (9)$$

According to [8], l_i is minimal (in the sense of expected value) when $\hat{p} = p$ (given p). Therefore it's the mean L_n that's being minimized in the universal prediction framework to find a model $\hat{p}$ that approximates the true probability distribution p. The most direct solution would be to 'translate' L_n into the required interval $[0, 1]$ using the exponential function (2^{-x}) – and we obtain a confidence measure. Note, however, that l_i is *not* minimal for $\hat{p} = p$, given $\hat{p}$! Thus, the agent would get higher confidence values if she guessed the policy incorrectly rather than correctly – which is unacceptable. Consider, for example, an agent who estimates the user's policy with the probability distribution $\hat{p} = P1$ at some moment (figure 1). If her guess is right (i.e. the user's policy is $p = P1$ indeed), the expected loss is $-0.5 \log 0.5 - 2 \cdot 0.25 \log 0.25 = 1.5$. Yet if the real policy is $p = P2$, then the expected loss is $-1 \log 0.5 = 1$, so the agent would be more confident than in the case when she guesses correctly. In consequence, the minimal loss is not the optimal loss in this case.

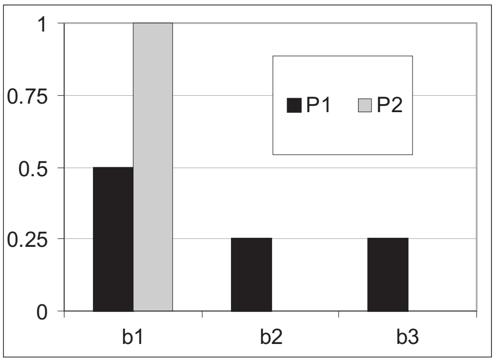

Figure 1: Minimal vs. optimal loss – an example

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*:

$$\begin{aligned} Opt_n &= EL_n = -\sum_{(b_1..b_n)} [p(b_1..b_n)\frac{1}{n}\sum_{i=1}^{n} \log_2 \hat{p}_i(b_i)] = \\ &= -\frac{1}{n}\sum_{i=1}^{n}\sum_{b_i} p_i(b_i)\ \log_2 \hat{p}_i(b_i) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{b} \hat{p}_i(b)\ \log_2 \hat{p}_i(b) \end{aligned} \quad (10)$$

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

$$\Delta_n = L_n - Opt_n = -\frac{1}{n}\sum_{i=1}^{n}\left[\log_2 \hat{p}_i(b_i^*) - \sum_b \hat{p}_i(b)\ \log_2 \hat{p}_i(b)\right] \tag{11}$$

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 $2^{-|\Delta_n|}$ will remain 0 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! Therefore the log-loss-based confidence measure proposed here includes a normalization factor:

$$C_{log} = 2^{-\left|\frac{\Delta_n}{\Delta_n^{max} - \Delta_n^{min}}\right|} \tag{12}$$

where

$$\Delta_n^{max} = \max_{(b_1^*..b_n^*)}\{\Delta_n\} = \max_{(b_1^*..b_n^*)}\{L_n - Opt_n\} = \max_{(b_1^*..b_n^*)}\{L_n\} - Opt_n \tag{13}$$

$$\Delta_n^{min} = \min_{(b_1^*..b_n^*)}\{\Delta_n\} = \min_{(b_1^*..b_n^*)}\{L_n - Opt_n\} = \min_{(b_1^*..b_n^*)}\{L_n\} - Opt_n \qquad \text{and} \tag{14}$$

$$\begin{aligned}
\Delta_n^{max} - \Delta_n^{min} &= \max_{(b_1^*..b_n^*)}\{L_n\} - \min_{(b_1^*..b_n^*)}\{L_n\} = \max_{(b_1..b_n)}\{-\frac{1}{n}\sum_{i=1}^{n}\log_2 \hat{p}_i(b_i)\} - \min_{(b_1..b_n)}\{-\frac{1}{n}\sum_{i=1}^{n}\log_2 \hat{p}_i(b_i)\} \\
&= -\frac{1}{n}\sum_{i=1}^{n}\min_{b_i}\log_2 \hat{p}_i(b_i) + \frac{1}{n}\sum_{i=1}^{n}\max_{b_i}\log_2 \hat{p}_i(b_i) = \frac{1}{n}\sum_{i=1}^{n}\left[\log_2 \max_b \hat{p}_i(b) - \log_2 \min_b \hat{p}_i(b)\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}\log_2 \frac{\max_b \hat{p}_i(b)}{\min_b \hat{p}_i(b)}
\end{aligned} \tag{15}$$

The measure has the following properties:

- $n\ \Delta_n$ 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 $0.5 \le C_{log} \le 1$;
- 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 $\hat{p}_i(b) = 0$. However, when $\hat{p}_i$ are frequency distributions, 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 frequencies will always be positive.

2.3 Log-loss Confidence with Temporal Decay

The temporal decay can be introduced into the log-loss confidence to make the recent loss values matter more than the old ones: $L_n^\lambda = M_\lambda(l_{i=1..n})$ rather than $\overline{l_{i=1..n}}$. Thus

$$L_n^\lambda = \frac{-\sum_{i=1}^n \lambda^{n-i} \log_2 \hat{p}_i(b_i^*)}{\sum_{i=1}^n \lambda^{n-i}} \tag{16}$$

Then:

$$\begin{aligned}
\Delta_n^\lambda &= L_n^\lambda - Opt_n^\lambda = \frac{\sum_{i=1}^n \lambda^{n-i}[-\log_2 \hat{p}_i(b_i^*) + \sum_b \hat{p}_i(b) \log_2 \hat{p}_i(b)]}{\sum_{i=1}^n \lambda^{n-i}} = \\
&= -M_\lambda(\log_2 \hat{p}_i(b_i^*))_{i=1..n} + M_\lambda(\sum_b \hat{p}_i(b) \log_2 \hat{p}_i(b))_{i=1..n} \qquad (17)\\
\Delta_n^{max,\lambda} - \Delta_n^{min,\lambda} &= \max_{(b_1^*..b_n^*)} \{-M_\lambda(\log_2 \hat{p}_i(b_i^*))\} - \min_{(b_1^*..b_n^*)} \{-M_\lambda(\log_2 \hat{p}_i(b_i^*))\} = \\
&= \frac{-\sum_{i=1}^n \lambda^{n-i} \log_2 \min_{b_i^*} \hat{p}_i(b_i^*)}{\sum_{i=1}^n \lambda^{n-i}} + \frac{\sum_{i=1}^n \lambda^{n-i} \log_2 \max_{b_i^*} \hat{p}_i(b_i^*)}{\sum_{i=1}^n \lambda^{n-i}} = \\
&= \frac{\sum_{i=1}^n \lambda^{n-i} \log_2[\max_b \hat{p}_i(b)/\min_b \hat{p}_i(b)]}{\sum_{i=1}^n \lambda^{n-i}} = M_\lambda(\log_2 \frac{\max_b \hat{p}_i(b)}{\min_b \hat{p}_i(b)}) \qquad (18)
\end{aligned}$$

Again,

$$C_{log}^\lambda = 2^{-|\frac{\Delta_n^\lambda}{\Delta_n^{max,\lambda} - \Delta_n^{min,\lambda}}|} \tag{19}$$

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

3 Experiments

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 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; the estimation can be then used in the decision-making process. The initial distribution $\hat{p}_0$ is either nil distribution (for C_V) or uniform distribution (for C_{log}). The agent can also use the maxmin equilibrium strategy to play things safe [11], or choose a more sophisticated behavior in the way described in section 3.1.

3.1 The Game

The motivation behind the measures proposed here is that an agent can use several alternative models to make her decisions. 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 values providing weights. For example, the agent can use two models: the user's profile (frequency distribution computed from available data) and some default user model. 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 she has in them.

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	1
normal offer	6	-20	1
safe offer	1.5	-1	1

The default model is defined in the Game Theory fashion: the user is assumed an enemy who always cheats. To get rid of the exploration/exploitation tradeoff we assume also that the user is rather simple-minded and his 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). The agent computes her expected payoffs with respect to each model: $e_p(a), e_d(a)$, and chooses the action with maximal value of $C \cdot e_p(a) + (1 - C) \cdot e_d(a)$.

The aim of the simulations was to compare the output of the hybrid agents (using C_V and C_{log} as the confidence respectively) to each other, to the hybrid agent using Wang's confidence $C_{Wang} = n/(n + 1)$, and to the outputs of both possible single-model agents. 1000000 independent random interactions (a sequence of 100 rounds each) have been simulated for every setting; the average results are presented in the following section.

3.2 Results

Figures 2 and 3 show how the confidence values evolve for a dynamic 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. $1 - expdev$. The dynamic users were simulated in the following manner: random initial and final policies p_0, p_{100} were generated for every simulation, and the user was changing his preferences in 3 big moves, after steps no. 30, 60 and 90: $p_i(b) = p_0(b) + (i \ \% \ 30)(p_{100}(b) - p_0(b))/3$.

Figures 4 and 5 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

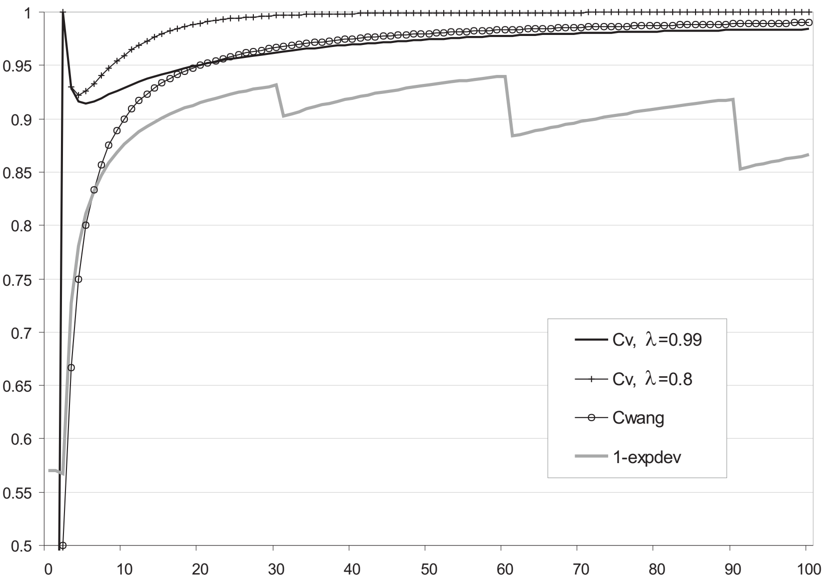

Figure 2: Confidence values: C_V vs. C_{Wang} vs. accurateness

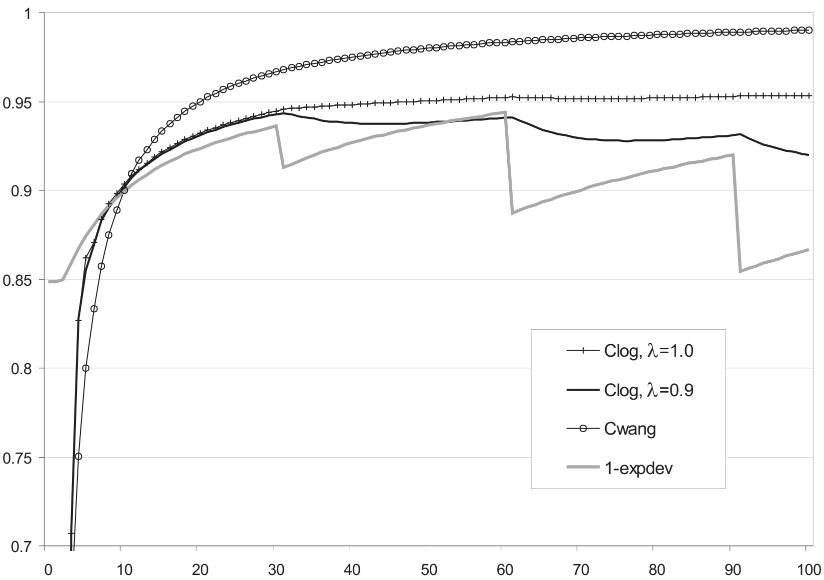

Figure 3: Confidence values: C_{log} vs. C_{Wang} vs. accurateness

the agent can start using the user profile successfully. The results of the simulations show that using the variance-based confidence from section 2.1 improves the agent's payoff, but it's never better than C_{Wang}, which is all the more surprising because the Wang's measure was designed to tackle a *static* and not dynamic environment. The problem is that the 'self-observation' approach proposed in the literature leans very heavily on the learning method being used by the agent. Usually the model gets stable quite fast regardless of the user's responses; any new item has a very small impact on the mean value after some, say, 20 observations have been collected. Therefore the measure doesn't really detect the

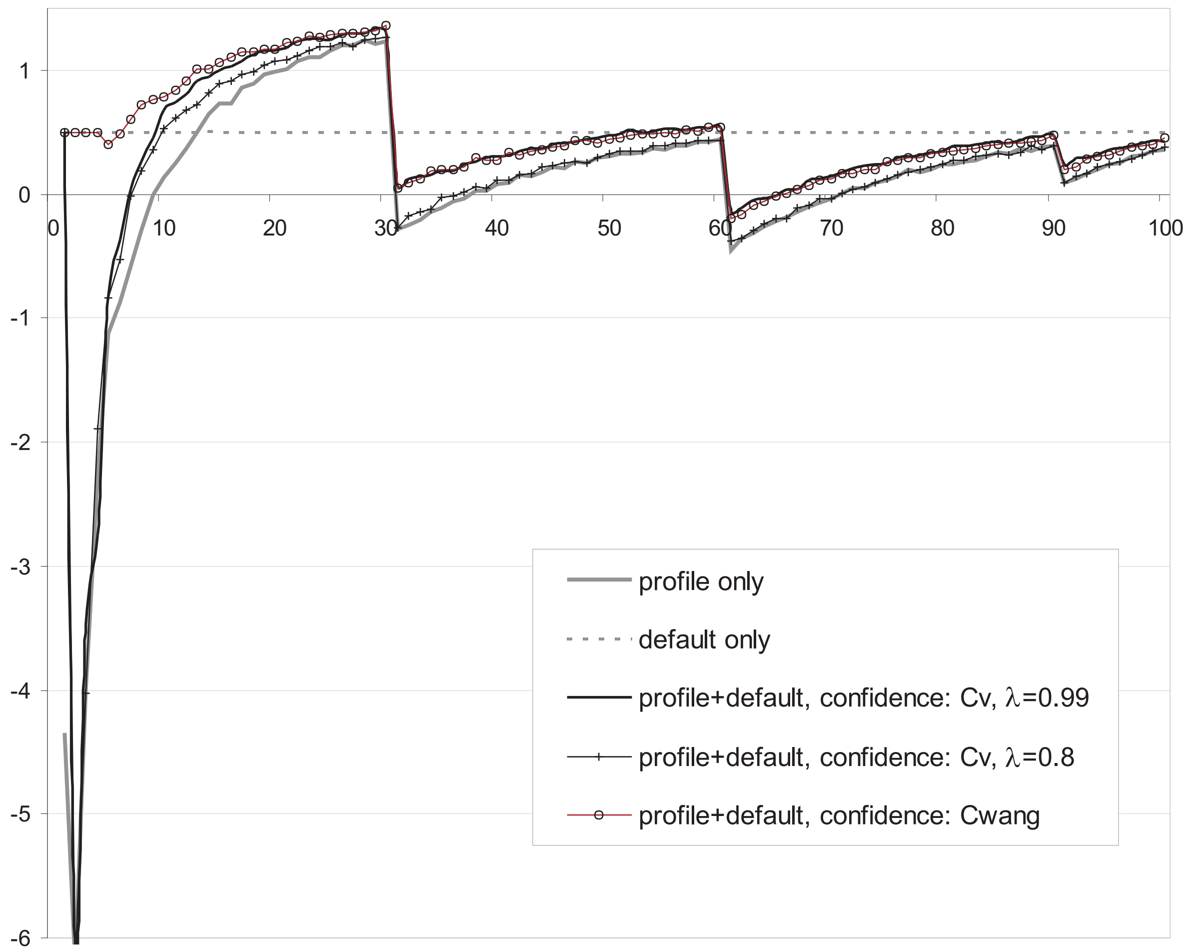

Figure 4: Hybrid agents vs. single-model agents: variance-based confidence

changes in the user's behavior. It's easy to see on figure 2 that C_V increases constantly almost all of the time, even when the model becomes less and less accurate, i.e. after the 30, 60 and 90th step. The agent may even focus on her most recent beliefs – it doesn't make things any better, because these are the beliefs that do not change.

The results of the experiments with an agent using the log-loss-based confidence are much more promising: first of all, the measure *does* detect the changes in the observed pattern, and it helps the agent to increase her payoff significantly. The self-information loss function provides a direct link between the model and the new observations. 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 90 steps of collecting and analyzing data.

4 Conclusions

The experiments showed that a confidence measure can be useful – at least in some settings – for instance, as a means for weighting alternative beliefs. The 'self-observation'

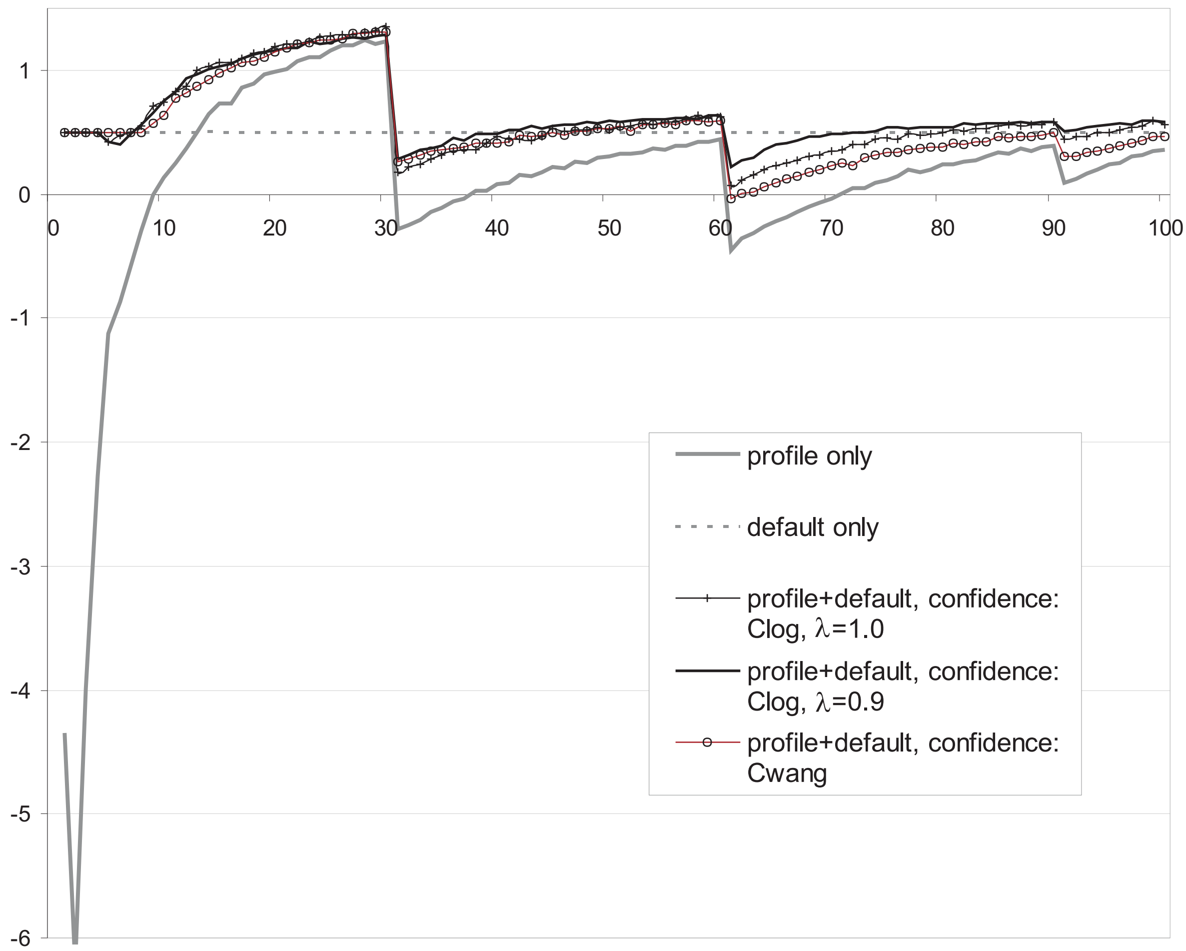

Figure 5: Hybrid agents vs. single-model agents: log-based confidence

approach, proposed e.g. in [9, 5], didn't prove very appropriate because the agent's knowledge can change quite slowly even when it's inaccurate. This led to investigation of a different confidence measure, based on a loss function that confronts the beliefs against the new observation directly – and with much better results.

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

References

[1] S. Hochreiter and M.C. Mozer. Beyond maximum likelihood and density estimation: A sample-based criterion for unsupervised learning of complex models. *Advances in Neural Information Processing Systems*, 13, 2001. Proceedings of NIPS'2000.

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

[3] 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.

[4] 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.

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

[6] 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.

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

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

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

[10] 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.

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

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

[13] 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.

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

[15] 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.

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