Datasize-Based Confidence Measure for a Learning Agent

Wojciech Jamroga
Parlevink Group
University of Twente, Netherlands
jamroga@cs.utwente.nl

Abstract

In this paper a confidence measure is considered for an agent who tries to keep a probabilistic model of her environment of action. The measure is meant to capture only one factor of the agent's doubt – namely, the issue whether the agent has been able to collect a sufficient number of observations. In this case stability of the agent's current knowledge may give some clue about the trust she can put in the model – indeed, some researchers from the field of probability theory suggest that such confidence should be based on the variance of the model (over time).

In this paper two different measures are proposed, both based on aggregate variance of the estimator provided by the learning process. The way the measures work is investigated through some simple experiments with simulated software agents. It turns out that an agent can benefit from using such measures as means for 'self-reflection'. The simulations suggest that the agent's confidence should reflect the deviation of her knowledge from the reality. They also show that it can be sometimes captured using very simple methods: a measure proposed by Wang is tested in this context, and it works seldom worse than the variance-based measures, although it seems completely ad hoc and not well suited for this particular setting of experiments at the first sight.

`Keywords:` multiagent systems, confidence measure, uncertainty, machine learning, user modeling

1 Introduction

There are roughly two possible sources of doubt for a learning agent. First, the agent may have collected too little data. For instance, when the agent starts interaction with a completely new user, her knowledge about the user is virtually none. However, the knowledge is utilized in the same way by most algorithms, regardless of the number of learning steps that have been taken so far.

Next, the environment might have changed considerably, so the data do not reflect its current shape.

The knowledge produced by a learning algorithm is often no more than a working hypothesis. It's necessary for the agent that she can make her decisions; however, trusting the knowledge blindly implies some additional assumptions which are not true in most real-life situations. It's good for the agent to have some measure of uncertainty in her own knowledge – to minimize the risk of a decision, especially in the case when she has several alternative models to choose among or combine.

This paper is focused on the first source of the agent's uncertainty: how much confidence can she have in her knowledge when there is not enough data to support it? The problem is analyzed in a very simple setting: the agent is assumed to be a 1-level agent – i.e. an agent that models other agents as stochastic agents (Sen and Weiss, 1999) – and the users are 0-level agents with probabilistic policies. The reinforcement is known beforehand for every decision of the agent, given a response from the user, and the domain of action is stateless (or at least the agent's perception doesn't let her distinguish between different states of the environment). The agent tries to estimate the actual policy of the user calculating a frequency distribution, which can be further used to find the decision with the maximal expected reward. The aim of the confidence is to represent meta-(un)certainty about the agent's knowledge, so when she has several alternative models available she can choose among them or combine their output. Thus, the actual confidence values should range from 0 (complete distrust) to 1 (full confidence).

2 The Learning Method: Counting with Decay

Assume an autonomous software agent A who interacts with some other agent B (for example, A may be an agent representing a bank and B may be a potential customer – a user of an Internet banking service). The interaction with the 'user' is sequential and it consists of subsequent turns: first A chooses to proceed with an action $a*$ from a finite set $ActA$, then B replies with some $b* \in ActB$, then A does $a' \in ActA$ and so on. Let $p_B(b|a) \equiv p_B(a,b)$ denote the current probability of agent B choosing action b as a response to A's action a. A tries to estimate the policy with a relative frequency distribution $\hat{p}_B$:

$$\hat{p}(b|a) \leftarrow \begin{cases} \frac{\hat{p}(b|a)N(a)\cdot\lambda+1}{N(a)\cdot\lambda+1} & a=a^*, b=b^* \\ \frac{\hat{p}(b|a)N(a)\cdot\lambda}{N(a)\cdot\lambda+1} & a=a^*, b\neq b^* \\ \hat{p}(b|a) & \text{else} \end{cases} \quad (1)$$

$$N(a) \leftarrow \begin{cases} N(a)\cdot\lambda+1 & a=a^* \\ N(a) & \text{else} \end{cases} \quad (2)$$

where $\lambda \in [0,1]$ is the decay rate implementing the way A 'forgets' older observations in favor of the more

recent ones to model users that may change their preferences dynamically (Kumar, 1998) (Koychev, 2000) (Koychev, 2001). $N(a)$ represents the data size after collecting n observations. Since the older observations are used only partially (the first one with weight λ^{n-1}, the second: λ^{n-2} etc.), the real quantity of data we use is

$$N(a) = \sum_{i=1}^{n} \lambda^{n-i} = \begin{cases} \frac{1-\lambda^n}{1-\lambda} & \text{for } 0 < \lambda < 1 \\ n & \text{for } \lambda = 1 \end{cases}$$

The nil distribution $\mathbf{0}(b|a) = 0$ is used as the initial one. If the decay rate is allowed to vary then $N(a) = \sum_{i=1}^{n} \prod_{j=i+1}^{n} \lambda_j$, where $\lambda_1, ..., \lambda_n$ denote the actual decay rates at the moments when the subsequent observations and updates were made.

Note that $\hat{p}(b|a)$ is basically a sample mean of a Bernoulli variable $Resp(b|a)$, although it's a *mean with decay*:

$$\hat{p}_n(b|a) = M_{\lambda_{1..n}}(Resp_{i=1,...,n}(b|a)), \quad (3)$$

$$\text{where} \quad Resp(b|a) = \begin{cases} 1 & \text{if } b \text{ is the user's response to } a \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

$$M_{\lambda_{1..n}}(X_{i=1,...,n}) = \frac{\sum_{i=1}^{n}(\prod_{j=i+1}^{n} \lambda_j) X_i}{\sum_{i=1}^{n} \prod_{j=i+1}^{n} \lambda_j} \quad (5)$$

Note also that for $\lambda = 1$ we obtain an ordinary frequency distribution with no temporal decay.
M_λ has some standard properties of a mean (the proofs are straightforward):

$$M_{\lambda_{1..n}}(X + Y) = M_{\lambda_{1..n}}(X) + M_{\lambda=1..n}(Y) \quad (6)$$

$$M_{\lambda_{1..n}}(aX) = aM_{\lambda_{1..n}}(X) \quad (7)$$

$$\sum_b M_{\lambda_{1..n}}(p_{i=1..n}(b)) = 1 \text{ if } p_i \text{ are probability functions} \quad (8)$$

3 Self-Confidence with Insufficient Data

It is often assumed that the (un)certainty an agent can have about her knowledge is nothing but a meta-probability or meta-likelihood – cf. (Draper, 1995) for instance. On the other hand, there are researchers who argue against it (Kyburg, 1988; Wang, 2001). This seems to reflect the assumption that the meta-uncertainty should refer to the usability of the model. Indeed, meta-probability isn't very useful in this case: even if we know for sure that the model is slightly different from the reality (in consequence, its meta-probability is exactly 0), it *does* matter whether it's close to the real situation or not (Wang, 2001). This is also the perspective adopted in this paper. In this respect, some authors propose approaches based on some notion of error or fitting obtained through a posterior verification of the model (Hochreiter and Mozer, 2001; Spiegelhalter et al., 1998; Marshall and Spiegelhalter, 1999). However, the disconfidence studied here is *a priori* not *a posteriori* by definition – therefore any posterior reasoning can do no good here. In consequence, purely practical solutions may be very useful and work surprisingly well in particular situations (Kumar, 1998; Wang, 2001).

It has been suggested that, when the model is a probability distribution, the agent's self-confidence may be defined using the variance of the distribution treated as a random quantity itself (Pearl, 1987; Kyburg, 1988). Thus, the confidence measures being proposed and studied in this paper are based on the notion of aggregate variance of the estimator provided by the learning process.

3.1 Binding the Variance of Sampling

Handbooks on statistics like (Berry and Lindgren, 1996) suggest a way to determine whether an amount of data is enough to estimate the population mean EX with a sample mean $\bar{X}$: we assume some acceptable error level ε and as soon as the sampling deviation (standard deviation, for instance) gets below this value: $\sigma(\bar{X}) \leq \varepsilon$, we feel satisfied with the estimation itself. Since the real deviation value is usually hard to obtain, an upper bound or an estimation can be used instead.

If we want the 'satisfaction measure' to be continuous, it seems natural that the satisfaction is full 1 when the condition holds for $\varepsilon = 0$, and it decreases towards 0 as the dispersion grows. It is proposed here that the confidence for a frequency distribution $\hat{p}(\cdot|a)$ can be somehow proportional to $1 - \sum_b disp(b|a)$, and the variance $var(\hat{p}(b|a))$ is used to express the dispersion $disp(b|a)$. The reason for choosing the variance is that $0 \leq \sum_b var(\hat{p}(b|a)) \leq 1$ in our case, while the same is not true for the standard deviation σ as well as the mean deviation $m.a.d$.

We assume that the old observations are appropriate only partially with respect to the (cumulative) data decay encountered so far. Let $n \geq 1$ be an arbitrary number. By the properties of the variance and given that $Resp_1(b|a), ..., Resp_n(b|a)$ represent a random sampling of the user's responses:

$$\begin{aligned} var(\hat{p}_n(b|a)) &= var\Big(M_\lambda(Resp_{i=1..n}(b|a))\Big) = \\ &= var\Big(\frac{\sum_{i=1}^{n} Resp_i(b|a)\lambda^{n-i}}{\sum_{i=1}^{n} \lambda^{n-i}}\Big) = \\ &= \frac{\sum_{i=1}^{n} var(Resp_i(b|a))\lambda^{2(n-i)}}{(\sum_{i=1}^{n} \lambda^{n-i})^2} \end{aligned}$$

$var(Resp_i(b|a))$ is a population variance at the moment when the ith observation was made. If $p_i(b|a)$ was the real probability of user responding with action b at that particular moment, then:

$$\begin{aligned} var(Resp_i(b|a)) &= p_i(b|a) - p_i^2(b|a) \\ \sum_b var(\hat{p}_n(b|a)) &= \frac{\sum_{i=1}^{n} \lambda^{2(n-i)} \Big(\sum_b p_i(b|a) - \sum_b p_i^2(b|a)\Big)}{(\sum_{i=1}^{n} \lambda^{n-i})^2} \end{aligned}$$

$\sum_b p_i^2(b|a)$ is minimal for the uniform distribution $p_i(b|a) = \frac{1}{|ActB|}$, so:

$$\sum_b var(\hat{p}_n(b|a)) \leq \frac{\sum_{i=1}^{n} \lambda^{2(n-i)}}{(\sum_{i=1}^{n} \lambda^{n-i})^2}(1 - \frac{1}{|ActB|}) \quad (9)$$

Let

$$dispb(a) = \frac{\sum_{i=1}^{n}\lambda^{2(n-i)}}{(\sum_{i=1}^{n}\lambda^{n-i})^2}(1-\frac{1}{|ActB|}) \quad (10)$$

$$Cbound(a) = 1 - dispb(a) \quad (11)$$

Now the confidence is never higher than it *can* be – the agent is playing it safe:

$$Cbound(a) \le 1 - \sum_b var(\hat{p}_n(b|a)) \quad (12)$$

Note also that $dispb(a)$ is a decreasing function of λ for $\lambda \in (0,1]$, so its value is always between $(1-\frac{1}{|ActB|})/n$ (the value for $\lambda = 1$), and $1-\frac{1}{|ActB|}$ (which is $\lim_{\lambda\to 0} dispb(a)$). Thus also

$$0 \le \frac{1}{|ActB|} \le \quad Cbound(a) \quad \le \frac{n-1}{n} + \frac{1}{n|ActB|} \le 1 \quad (13)$$

In the more general case when λ is variable:

$$\begin{aligned}\sum_b var(\hat{p}_n(b|a)) &= var\Big(M_{\lambda=1..n}(Resp_{i=1..n}(b|a))\Big) = \\ &= \frac{\sum_b\sum_{i=1}^{n}(\prod_{j=i+1}^{n}\lambda_j)^2 var(Resp_i(b|a))}{(\sum_{i=1}^{n}\prod_{j=i+1}^{n}\lambda_j)^2} \\ &\le \frac{\sum_{i=1}^{n}(\prod_{j=i+1}^{n}\lambda_j)^2}{(\sum_{i=1}^{n}\prod_{j=i+1}^{n}\lambda_j)^2}(1-\frac{1}{|ActB|})\end{aligned}$$

It is possible to compute

$$Lsqr_n = \sum_{i=1}^{n}(\prod_{j=i+1}^{n}\lambda_j)^2 = \lambda_n^2 Lsqr_{n-1} + 1$$

$$\text{and} \quad L_n = \sum_{i=1}^{n}\prod_{j=i+1}^{n}\lambda_j = \lambda_n L_{n-1} + 1$$

in an incremental way. Then the confidence can be defined as

$$Cbound(a) = 1 - \frac{Lsqr_n}{(L_n)^2}(1-\frac{1}{|ActB|}) \quad (14)$$

which is never greater than $1 - \sum_b var(\hat{p}_n(b|a))$.

3.2 Adjusting the Confidence Value

The value of $dispb(a)$ proposed above can give some idea of the uncertainty the agent should have in $\hat{p}(\cdot|a)$. The most straightforward solution: $Cbound(a) = 1 - dispb(a)$ may not always work well for practical reasons, though. The agent can use a 'magnifying glass' parameter $\mathfrak{m}$ to sharpen her judgment:

$$Cbound(a) = (1 - dispb(a))^{\mathfrak{m}} \quad (15)$$

Since different learning methods show different dynamics of knowledge evolution, $\mathfrak{m}$ offers the agent an opportunity to 'tune' her confidence measure to the actual learning algorithm.

3.3 Forward-Oriented Distrust

In a perfect case we would be interested in the *real* variation of the sampling made so far – to have some clue about the expected (real) deviation from the estimation $\hat{p}_n$ obtained through the sampling. This value can be approached through its upper bound – as proposed in section 3.1. Alternatively we can try to approximate the variability we may expect from our estimator in the future (possibly with temporal discount).

It is worth noting that insufficient data can be seen as generating 'future oriented distrust': even if the agent's knowledge doesn't change much during the first few steps (e.g. the corresponding user's responses are identical) it may change fast in the very next moment. When the evidence is larger, the model of the reality being produced gets more stable and it can hardly be changed by a single observation. If we assume that the learning algorithm is correct – i.e. the model converges to the true user characteristics as the number of input data increases – then the agent can base her self-assessment on the possible future-oriented dispersion (possibly with a temporal discount Λ – to make the closer entries matter more than the farther ones):

$$Csize_\Lambda(a) = (1 - fdisp_\Lambda(a))^{\mathfrak{m}} \quad (16)$$

$$\begin{aligned}fdisp_\Lambda(a) &= \lim_{k\to\infty} E\, fdisp_\Lambda^k(a) = \\ &= \lim_{k\to\infty} E\Big(\sum_b V_\Lambda(\hat{p}_{n+k}(b|a), ..., \hat{p}_n(b|a))\Big)\end{aligned} \quad (17)$$

where $\hat{p}$ is the agent's current model of the user, every $\hat{p}_{n+i}(\cdot|a), i = 1..k$ is obtained from $\hat{p}_{n+i-1}(\cdot|a)$ through response b_i^*, and the mean is taken over all the response sequences $(b_1^*, ..., b_k^*)$. The sample variance with discount/decay can be defined in a natural way as:

$$V_\Lambda(X) = M_\Lambda(X - M_\Lambda X)^2 \quad (18)$$

By properties (6), (7): $V_\Lambda(X) = M_\Lambda(X^2) - M_\Lambda^2(X)$. Assuming uniform a priori likelihood for all the possible sequences, the expected value can be approximated with simple averaging:

$$avg_{(b^*_{i=1..k})}\, fdisp_\Lambda^k(a) = \frac{1}{|ActB|^k}\sum_{(b^*_{i=1..k})} fdisp_\Lambda^k(a) \quad (19)$$

The limit in (17) can be then approximated iteratively for the generalized frequency counting presented in section 2. Let:

$$Mpsqr^k(a) \equiv avg_{(b^*_{i=1..k})}\sum_b M_\Lambda(\hat{p}^2_{n+k}(b|a), ..., \hat{p}^2_n(b|a))$$

$$Msqr^k(a) \equiv avg_{(b^*_{i=1..k})}\sum_b M^2_\Lambda(\hat{p}_{n+k}(b|a), ..., \hat{p}_n(b|a))$$

$$MP^k(a) \equiv avg_{(b^*_{i=1..k})}\sum_b \hat{p}_{n+k}(b|a) M_\Lambda(\hat{p}_{n+k}(b|a), ..., \hat{p}_n(b|a))$$

$$Psqr^k(a) \equiv avg_{(b^*_{i=1..k})}\sum_b \hat{p}^2_{n+k}(b|a)$$

Then for $0 < \Lambda < 1$

$$avg_{(b^*_{i=1..k})}\, fdisp_\Lambda^k(a) = Mpsqr^k(a) - Msqr^k(a)$$

$$Mpsqr^k(a) = \frac{1-\Lambda^k}{1-\Lambda^{k+1}}Mpsqr^{k-1}(a) + \frac{(1-\Lambda)\Lambda^k}{1-\Lambda^{k+1}}Psqr^k(a)$$

$$Msqr^k(a) = \frac{(1-\Lambda^k)^2}{(1-\Lambda^{k+1})^2}Msqr^{k-1}(a) + \frac{2(1-\Lambda)\Lambda^k}{1-\Lambda^{k+1}}MP^k(a) - \frac{\Lambda^{2k}(1-\Lambda)^2}{(1-\Lambda^{k+1})^2}Psqr^k(a)$$

fdisp ($\hat{p}$, a, Λ, λ, N, precision);

Iterative approximation of $fdisp_\Lambda(a)$. Returns the approximate value of the average future-oriented temporally-discounted dispersion (with temporal discount $0 < \Lambda < 1$). λ is the current temporal decay rate on the observation level, N represents the number of observations collected so far that takes into account the cumulative time decay; $\hat{p}$ is the current model of the user.

$Mpsqr, Msqr, MP, Psqr \leftarrow \sum_b \hat{p}^2(b|a)$; /* initial values */
$k \leftarrow 0$;
$V \leftarrow 0$;
repeat
  $V_{old} \leftarrow V$;
  $k \leftarrow k + 1$;
  $N \leftarrow N\lambda + 1$;
  $Psqr \leftarrow \left(\frac{N-1}{N}\right)^2 Psqr + \frac{2(N-1)}{|ActB|N^2} + \frac{1}{N^2}$;
  $MP \leftarrow \frac{(1-\Lambda^k)(N-1)}{(1-\Lambda^{k+1})N} MP + \frac{(1-\Lambda)\Lambda^k}{1-\Lambda^{k+1}} Psqr + \frac{1-\Lambda^k}{|ActB|(1-\Lambda^{k+1})N}$;
  $Msqr \leftarrow \frac{(1-\Lambda^k)^2}{(1-\Lambda^{k+1})^2} Msqr + \frac{2(1-\Lambda)\Lambda^k}{1-\Lambda^{k+1}} MP - \frac{\Lambda^{2k}(1-\Lambda)^2}{(1-\Lambda^{k+1})^2} Psqr$;
  $Mpsqr \leftarrow \frac{1-\Lambda^k}{1-\Lambda^{k+1}} Mpsqr + \frac{(1-\Lambda)\Lambda^k}{1-\Lambda^{k+1}} Psqr$;
  $V \leftarrow Mpsqr - Msqr$;
until $|V - V_{old}| \leq$ precision;
return(V);

Figure 1: The algorithm for iterative approximation of $fdisp(a)$.

$$MP^k(a) = \frac{(1-\Lambda^k)(N_k-1)}{(1-\Lambda^{k+1})N_k} MP^{k-1}(a) + \frac{(1-\Lambda)\Lambda^k}{1-\Lambda^{k+1}} Psqr^k(a) + \frac{1-\Lambda^k}{|ActB|(1-\Lambda^{k+1})N_k}$$

$$Psqr^k(a) = \left(\frac{N_k-1}{N_k}\right)^2 Psqr^{k-1}(a) + \frac{2(N_k-1)}{|ActB|N_k^2} + \frac{1}{N_k^2}$$

where $N_k = N\lambda^k + \sum_{i=0}^{k-1} \lambda^i$, and $N = \sum_{i=1}^{n} \lambda_i$ is the actual (decayed) data size; $\lambda = \lambda_n$ is the current observation decay rate. The resulting algorithm is shown on figure 3.3. Moreover, the limit can be proved to exist, so the algorithm is convergent.

Proof: to prove the convergence of the sequence $V^k = avg_{(b^*_{i=1..k})}\ fdisp^k_\Lambda(a)$, we will find an (a_k) such that $|V^k - V^{k-1}| \leq a_k$ for every k, and $\sum_{i=1}^{k} a_i$ forms a convergent series. Then the series $\sum_{i=1}^{k}(V^i - V^{i-1}) = V^k$ is also convergent.

Note that:

$$|V^k - V^{k-1}| = |Mpsqr^k - Mpsqr^{k-1} + Msqr^{k-1} - Msqr^k| =$$
$$= |(\frac{1-\Lambda^k}{1-\Lambda^{k+1}} - 1)Mpsqr^{k-1} + (1 - \frac{(1-\Lambda^k)^2}{(1-\Lambda^{k+1})^2})Msqr^{k-1}$$
$$+\frac{(1-\Lambda)\Lambda^k}{1-\Lambda^{k+1}} Psqr^k - \frac{2(1-\Lambda)\Lambda^k}{1-\Lambda^{k+1}} MP^k + \frac{\Lambda^{2k}(1-\Lambda)^2}{(1-\Lambda^{k+1})^2} Psqr^k|$$
$$\leq |\frac{\Lambda^k(\Lambda-1)}{1-\Lambda^{k+1}} Mpsqr^{k-1}| + |\frac{\Lambda^k(2-2\Lambda-\Lambda^k+\Lambda^{k+2})}{(1-\Lambda^{k+1})^2} Msqr^{k-1}|$$
$$+|\frac{(1-\Lambda)\Lambda^k}{1-\Lambda^{k+1}} Psqr^k| + |\frac{2(1-\Lambda)\Lambda^k}{1-\Lambda^{k+1}} MP^k| + |\frac{\Lambda^{2k}(1-\Lambda)^2}{(1-\Lambda^{k+1})^2} Psqr^k|$$
$$\leq \frac{\Lambda^k(1-\Lambda)}{1-\Lambda^{k+1}} + \frac{\Lambda^k(1-\Lambda)(2-\Lambda^k(1+\Lambda))}{(1-\Lambda^{k+1})^2} +$$
$$+\frac{\Lambda^k(1-\Lambda)}{1-\Lambda^{k+1}} + \frac{2\Lambda^k(1-\Lambda)}{1-\Lambda^{k+1}} + \frac{\Lambda^{2k}(1-\Lambda)^2}{(1-\Lambda^{k+1})^2}$$

because $0 \leq Mpsqr, Msqr, MP, Psqr \leq 1$ by (8). Thus

$$|V^k - V^{k-1}| \leq \frac{\Lambda^k(1-\Lambda)}{(1-\Lambda^{k+1})^2} + 2\frac{\Lambda^k(1-\Lambda)}{(1-\Lambda^{k+1})^2} + \frac{\Lambda^k(1-\Lambda)}{(1-\Lambda^{k+1})^2}$$
$$+2\frac{\Lambda^k(1-\Lambda)}{(1-\Lambda^{k+1})^2} + \frac{\Lambda^k(1-\Lambda)}{(1-\Lambda^{k+1})^2} \leq$$
$$\leq 7\frac{\Lambda^k(1-\Lambda)}{(1-\Lambda^{k+1})^2} \leq 7\frac{\Lambda^k(1-\Lambda)}{(1-\Lambda)^2} = \frac{7}{1-\Lambda}\Lambda^k,$$

QED. □

Note also that, for every k, $V^k \geq 0$ (because it's a sum of nonnegative elements); on the other hand $V^k \leq 1$ (because $V^k = Mpsqr^k - Msqr^k$, and $0 \leq Mpsqr^k, Msqr^k \leq 1$). In consequence:

$$0 \leq Csize_\Lambda(a) \leq 1 \quad (20)$$

The way both measures work has been studied through some experiments in section 4.

4 Simulations

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 with a relative frequency distribution, counting the user's responses; at the same time she computes a confidence value for the profile acquired so far. 1000000 independent interactions (a sequence of 100 rounds each) with a random user process have been simulated; the average results are presented on the following charts.[1]

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

Figures 2 and 3 show how the confidence values evolve for a static user (a user whose policy does

[1] only the output of the first 40 rounds is presented on most charts to emphasize the part where the main differences lie. The results for rounds $41 - 100$ were more or less the same.

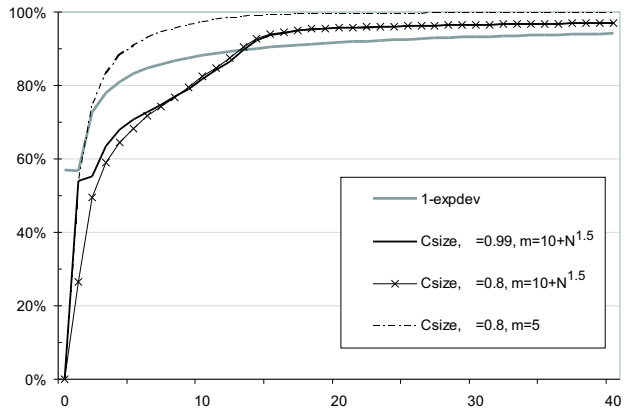

Figure 2: Confidence vs. accurateness: *Csize*

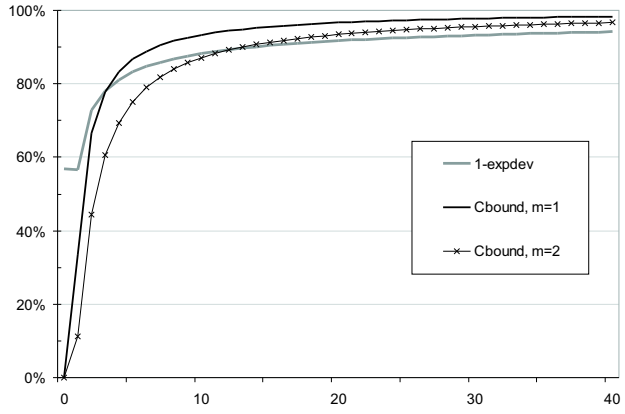

Figure 3: Confidence vs. accurateness: *Cbound*

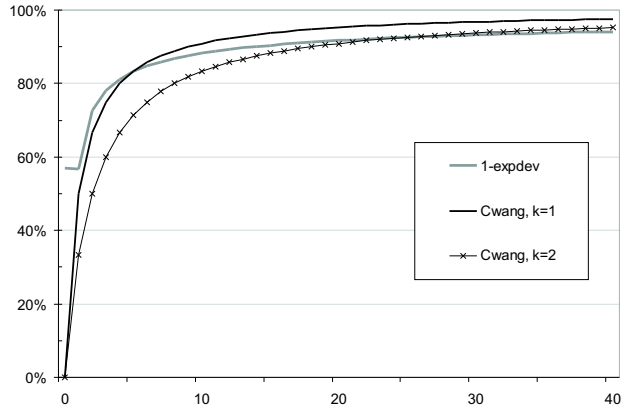

Figure 4: Wang's confidence for $k = 1$ and 2

not change throughout the experiment) and decay rate $\lambda = 1$, while figure 4 show the characteristics of the Wang's confidence *Cwang* $= N/(N + k)$. 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 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.

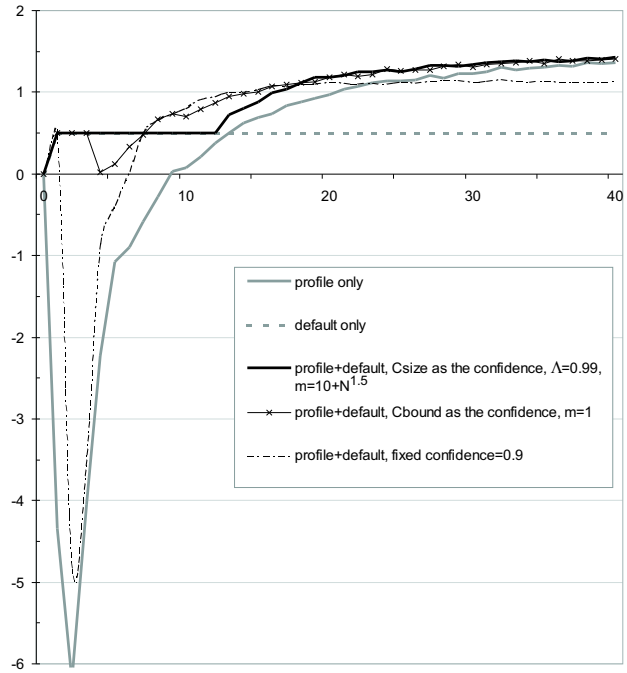

Figure 5: Hybrid agents vs. single-model agents: the average payoffs

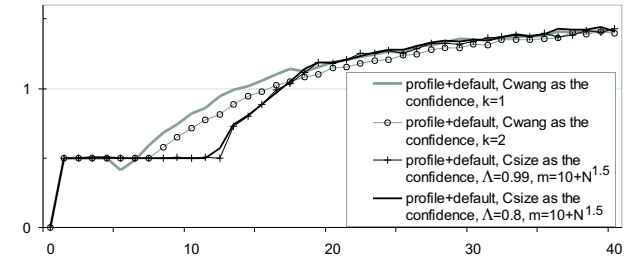

Figure 6: Hybrid agents vs. single-model agents: the average payoffs continued

The user's profile is computed as a plain frequency distribution. The default model, on the other hand, is defined in the Game Theory fashion: the user is assumed an enemy who always cheats. To get

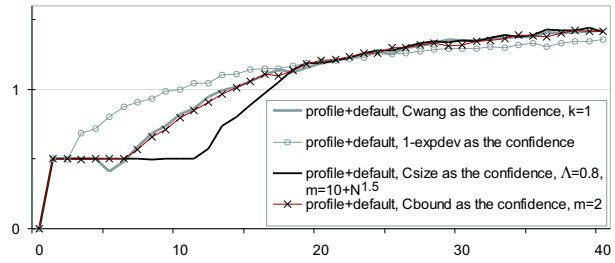

Figure 7: Hybrid agents vs. single-model agents: the average payoffs continued

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). Now the agent can evaluate her actions with their expected payoffs. She computes the evaluation based on the user's profile: $e_p(a)$, and the evaluation based on the default model as well: $e_d(a)$; then she chooses the action with maximal value of $C \cdot e_p(a) + (1 - C) \cdot e_d(a)$. In consequence – as the charts show – the agent doesn't lose money at the beginning of an interaction (because C 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.

Figures 5, 6 and 7 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.[2] *Cwang* (for $k = 1$) and *Cbound* (for $\mathfrak{m} = 2$) give best results, while *Csize* fares slightly worse despite quite complicated parameters setting ($\Lambda = 0.8$ and variable $\mathfrak{m} = 10 + N^{1.5}$). Experiments with other payoff tables gave similar results.

The measures presented here are primarily designed to tackle lack of data, not the user's dynamics. However, some experiments with dynamic users have also been run. Figure 8 presents the confidence evolution for a dynamic user and $\lambda = 0.95$. Figure 9 shows the results of the 'banking game' in the dynamic case. Here, the hybrid agent using *Cbound* fares best, with other hybrid agents close behind. Most notably, the *Cbound* agent is never worse than both single-model agents. The agent using only the default model is omitted on the chart to make it clearer: as before, her average payoff has been about 0.5 per round all the time.[3]

[2] unfortunately, it isn't possible to present all the results on a single chart because the chart would be completely unreadable then.

[3] a similar chart for $\lambda = 1$ shows the same regularities, although the payoffs are generally worse because the learning method is less flexible.

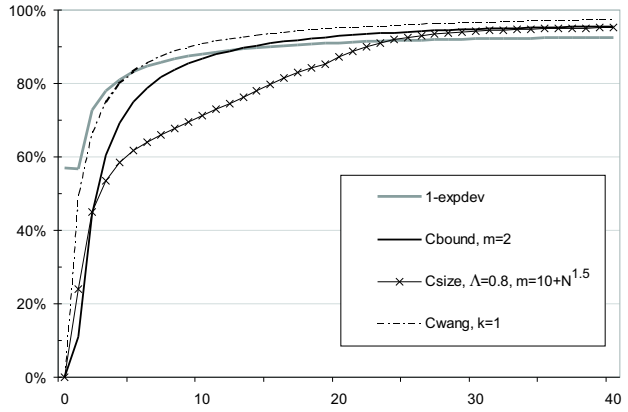

Figure 8: Confidence: interaction with a dynamic user, $\lambda = 0.95$

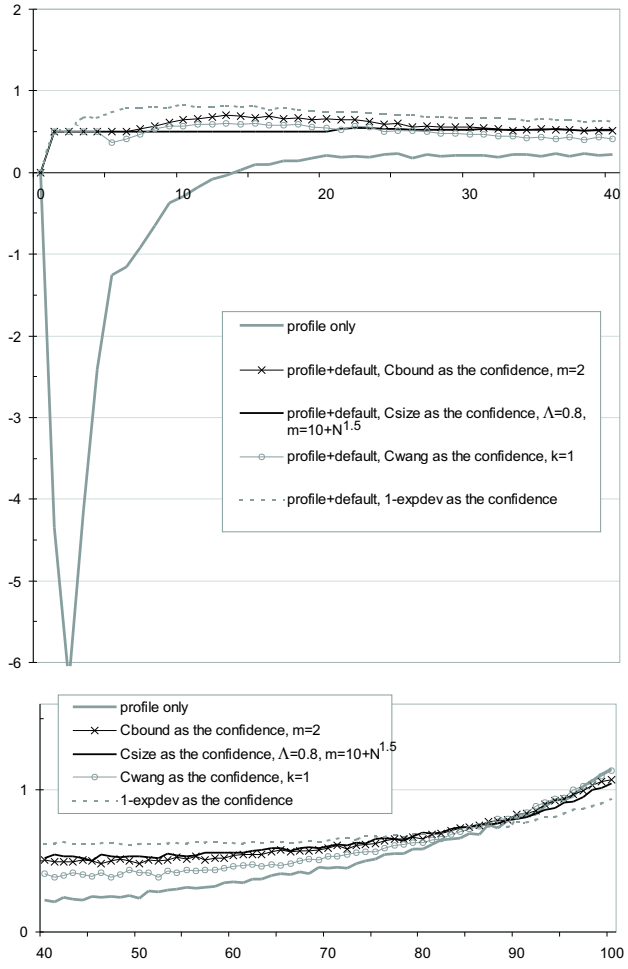

Figure 9: Hybrid agents vs. single-model agents: playing with a dynamic opponent, $\lambda = 0.95$

It should be obvious that the confidence needed to combine alternative models in the manner presented here is neither meta-probability nor meta-likelihood. The simulations suggest that the practical uncertainty concerned here is rather related to the distance/deviation of the model from the reality in a way.

Interestingly, an agent using $1 - expdev$ as her confidence measure gets positively best payoff in the initial phase of the interaction (and after that plays slightly worse) – see figures 7 and 9. Perhaps the expected absolute deviation isn't the best deviation measure for this purpose but it seems a close shot at least.

One issue should be made clear at the end, namely the way the dynamic users were simulated. It wasn't easy, because human users are hardly random with respect to their policies. True, humans' preferences drift – and the drift is never completely predictable – but neither is it completely chaotic. Real users are usually committed to their preferences somehow, so the preferences drift more or less inertly (the drift changes its direction in a long rather than short run). Here, random initial and final policies p_0, p_{100} were generated for every simulation, and the user was changing his preferences from p_0 to p_{100} in a linear way: $p_i(b) = p_0(b) + \frac{i}{100}(p_{100}(b) - p_0(b))$. It should be noted that the learning method used in the experiments (i.e. counting with decay) is *not* linear, so it's not true that this particular type of user simulation dynamics was chosen to suit the learning algorithm.

5 Conclusions

The experiments showed that the confidence measures can be useful – at least in some settings. Two measures have been proposed: *Cbound* and *Csize*, both based on the variance of the potential model evolution. In the simulations the agent using *Cbound* received slightly better results, especially for 'magnifying glass' parameter $\mathfrak{m} = 2$ (as long as the payoff was concerned). Moreover, the experiments with *Csize* revealed its important deficiency: in most cases a fixed $\mathfrak{m}$ proved too mild so the agent's confidence was reaching 1 almost at once (which is usually too fast). In consequence, the agent whose strategy was based upon *Csize* had to employ quite complicated 'tuning' scheme ($\mathfrak{m} = 10 + N^{1.5}$) plus an additional parameter Λ, while the agent using *Cbound* took (almost) the raw value of the bound. This suggests that the agent using *Csize* may expect serious problems with successful tuning of the confidence value in any real-life application, where the dynamics of the environment is changing, and the tuning process itself can hardly be cost-free.

The results of the simulations suggest that such practical uncertainty measure can be somehow based upon the estimated deviation of the model from the real state of affairs instead of the meta-probability of the model correctness or even the (potential) model variability over time. They show also that it may be captured approximately using very simple means: $Cwang = N/(N+1)$ for instance – in many domains of application.

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

References

D.A. Berry and B.W. Lindgren. 1996. *Statistics: Theory and Methods*. Wadsworth Publishing Company : Belmont, CA.

D. Carmel and S. Markovitch. 1996. Learning and using opponent models in adversary search. Technical Report CIS9609, Technion, Haifa.

D. Draper. 1995. Assessment and propagation of model uncertainty. *Journal of the Royal Statistical Society Series*, B(57):45–97.

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

G.J. Klir. 1999. Uncertainty and information measures for imprecise probabilities: An overview. In *Proceedings of the First International Symposium on Imprecise Probabilities and Their Applications*.

I. Koychev. 2000. Gradual forgetting for adaptation to concept drift. In *Proceedings of ECAI 2000 Workshop "Current Issues in Spatio-Temporal Reasoning"*, pages 101–106.

I. Koychev. 2001. Learning about user in the presence of hidden context. In *Proceedings of Machine Learning for User Modeling: UM-2001 Workshop*, pages 101–106. `http://www.dfki.de/~rafer/um01-ml4um-ws/proceedings.html`.

S. Kumar. 1998. 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. Tech. Report AI98-267.

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

E.C. Marshall and D.J. Spiegelhalter. 1999. Strategies for inference robustness in complex modelling: An application to longitudinal performance measures. Technical report, MRC Biostatistics Unit, Cambridge, UK.

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

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

D.J. Spiegelhalter, N.G. Best, and B.P. Carlin. 1998. Bayesian deviance, the effective number of parameters, and the comparison of arbitrarily complex models. Technical report, MRC Biostatistics Unit, Cambridge, UK.

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

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