Universal Agents in Repeated Matrix Games

Jan Poland[*]
Graduate School of Information Science and Technology
Hokkaido University, Japan
jan@ist.hokudai.ac.jp

Marcus Hutter
IDSIA
Galleria 2, CH-6928 Manno
Switzerland
marcus@idsia.ch

ABSTRACT

We study and compare the learning dynamics of two universal learning agents, one based on Bayesian learning and the other on prediction with expert advice. Both approaches have strong asymptotic performance guarantees. When confronted with the task of finding good long-term strategies in repeated 2×2 matrix games, they behave quite differently.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

1. INTRODUCTION

Today, data mining and machine learning is typically treated in a *problem-specific* way: People propose algorithms to solve a particular problem (such as learning to classify points in a vector space), they prove properties and performance guarantees of their algorithms (e.g. for Support Vector Machines), and they evaluate the algorithms on toy or real data, with the (potential) aim to use them afterwards in real-world applications. In contrast, it seems that a *universal agent*, i.e. a *single* algorithm that is applied for *all* (or at least "many") problems, is neither feasible in terms of computational costs nor competitive in (practical) performance. Nevertheless, understanding universal agents is important: On the one hand, their practical success would lead a way to real Artificial Intelligence. On the other hand, *principles* and ideas from universal learning can be of immediate use, and of course machine learning research aims at exploring and establishing more and more general concepts and algorithms.

Because of its practical restrictions, most of the understanding of universal learning so far is theoretical. Some approaches that have been suggested in the past are (adaptive) Levin search [8, 14], Optimal Ordered Problem Solver [13] and reinforcement learning with split trees [9] among others. For a thorough discussion see e.g. [6]. In this paper, we concentrate on two approaches with strong theoretical guarantees in the limit: the AIξ agent based on Bayesian learning [6] and *FoE* based on Prediction with expert advice [12].

Both models work in the setup of a *sequential decision problem*: An *agent* interacts with an *environment* in discrete time t. At each time step, the agent does some *action* and receives a *feedback* from the environment. The feedback consists of a *loss* (or *reward*) plus maybe more information. (It is usually just a matter of convenience if losses or rewards are considered, as one can be transformed into the other by reverting the sign. Accordingly, in this paper we switch between both, always preferring the more convenient one.) In addition to this *instantaneous* loss, we will also consider the *cumulative* loss which is the sum of the instantaneous losses from $t=1$ up to the current time step, and the *average per round* loss which is the cumulative loss divided by the total number of time steps so far (and the same for reward).

Most learning theory known so far concentrates on *passive* problems, where our actions have an influence on the instantaneous loss, but *not on the future behavior of the environment*. All regression, classification, (standard) time-series prediction tasks, common Bayesian learning and prediction with expert advice, and many others fall in this category. In contrast, here we deal with *active* problems. The environment may be *reactive*, i.e. react to our actions, which is the standard situation considered in reinforcement learning. These cases are harder in theory, and it is often impossible to obtain relevant performance bounds in general.

Both approaches we consider and compare are based on finite or countably infinite *base classes*. In the Bayesian decision approach, the base class consists of *hypotheses* or *models* for the environment. A model is a complete description of the (possibly probabilistic) behavior of the environment. In order to prove guarantees, it is usually assumed that the true environment is contained in the model class. Experts algorithms in contrast work with a class of decision-makers or *experts*. Performance guarantees are proven without any assumptions *in the worst case*, but only relative to the best expert in the class. In both approaches, the model class is endowed with a *prior*. If the model class is finite and contains n elements, it is common to choose the uniform prior $\frac{1}{n}$. For *universal* learning it turns out that universal

[*]This work was supported by SNF grant 2100-67712.02 and JSPS 21st century COE program C01.

base classes for both approaches can be constructed from the set of all programs on some fixed universal (prefix) Turing machine. Then each program naturally corresponds to an element in the base class, and a prior weight is defined by $w(\textit{program})=2^{-\textit{length}(\textit{program})}$ (provided that the input tape of the Turing machine is binary). The prior is a (sub-) probability distribution on the class, i.e. $\sum w\leq 1$.

Contents. The aim of this paper is to better understand the *actual learning dynamics* and properties of the two universal approaches, which are both "universally optimal" in a sense specified later. Clearly, the universal base class is computationally very expensive or infeasible to use. So we will restrict on simpler base classes that are "universal" in a much weaker sense: we will employ complete *Markov* base classes where each element sees only the previous time step. Although these classes are not truly universal, they are general enough (and not tailored towards our applications), such that we expect the outcome to be a good indication for the dynamics of true universal learning. The problems we study in this paper are 2×2 *matrix games.* (We will not go into the deep literature on learning equilibria in matrix games, as our primary interest is the universal learning dynamics.) Matrix games are simple enough such that a "universal" algorithm with our restricted base class can learn something, yet they provide interesting and nontrivial cases for reactive environments, where really active learning is necessary. Moreover, in this way we can set up a direct competition between the two universal learners. The paper is structured as follows: In the next two sections, we present both universal learning approaches together with their theoretical guarantees. Section 4 contains the simulations, followed by a discussion in Section 5.

2. BAYESIAN SEQUENTIAL DECISIONS (AIξ)

Passive problems. Inductive inference problems can be brought into the following form: Given a string $x_{<t}\equiv x_{1:t-1}:=x_1x_2...x_{t-1}$, guess its continuation x_t. Here and in the following we assume that the symbols x_t are in a finite alphabet $\mathcal{X}$; for concreteness the reader may think of $\mathcal{X}=\{0,1\}$. If strings are sampled from a probability distribution $\mu:\mathcal{X}^*\to[0,1]$, then predicting according to $\mu(x_t|x_{<t})$, the probability conditioned on the history, is optimal. If μ is unknown, predictions may be based on an approximation of μ. This is what happens in *Bayesian sequential prediction*: Let the *model class* $\mathcal{M}:=\{\mu_1,\mu_2,...\}$ be a finite or countable set of distributions on strings $\mu_i(x_{1:t}|y_{1:t})$ that are additionally conditionalized to the past actions $y_{<t}$. The actions are necessary for dealing with reactive environments as introduced above. We agree on the convention that the learner issues action y_t *before* seeing x_t. Let $\{w_1,w_2,...\}$ be a prior on $\mathcal{M}$ satisfying $\sum w_i\leq 1$. Then the *Bayes mixture* is the weighted average

$$\xi(x_{1:t}|y_{1:t}):=\sum\nolimits_i w_i\cdot\mu_i(x_{1:t}|y_{1:t}).$$

One can show that the ξ-predictions rapidly converge to the μ-predictions almost surely, if we assume that $\mathcal{M}$ contains the true distribution: $\mu\in\mathcal{M}$. This is not a serious constraint if we include *all* computable probability distributions in $\mathcal{M}$. This universal model class corresponds to all programs on a fixed universal Turing machine (cf. the introduction and [15, 6]).

In a passive prediction problem, the behavior of the environments μ_i do not depend on our actions $y_{1:t}$. Here we may interpret our action y_t as a prediction of x_t. Assume that $\ell:(y_t,x_t)\mapsto[0,1]$ is a function defining our instantaneous loss. Then *the average per round regret of ξ tends to 0 at least at rate $t^{-1/2}$*, precisely [6]

$$\tfrac{1}{t}L^\xi_{1:t}\leq\tfrac{1}{t}L^\mu_{1:t}+2\sqrt{\ln w_\mu^{-1}/t}. \qquad (1)$$

Here, $L^\xi_{1:t}$ is the cumulative μ-expected loss of the ξ-predictions. The ξ-prediction (and likewise the μ-prediction) is chosen *Bayes optimal* for the given loss function: $y^\xi_t=\mathrm{argmin}_{y_t}\sum_{x_t}\ell(y_t,x_t)\xi(x_{1:t}|y_{1:t})$. The difference $L^\xi_{1:t}-L^\mu_{1:t}$ is termed *regret.*

Active problems. If the environment is *reactive*, i.e. depends on our action, then it is easy to construct examples where the greedy Bayes optimal loss minimization is not optimal. Instead, the *far-sighted AIξ-agent* chooses the action

$$y^{\xi,d}_t=\arg\min_{y_t}\sum_{x_t}...\min_{y_{t+d}}\sum_{x_{t+d}}\ell_{t:t+d}\xi(x_{1:t+d}|y_{1:t+d}). \qquad (2)$$

where $\ell_{t:t+d}=\ell(y_{t:t+d},x_{t:t+d})=\sum_{s=t}^{t+d}\ell(y_s,x_s)$ and d is the depth of the *expectimin-tree* the agent computes by means of (2). We refer to $t+d$ as the (current) *horizon.* If we knew the final time T in advance and had enough computational resources, we could choose $d=T-t$ according to the *fixed horizon T*. Taking d fixed and small (e.g. $d=8$) is computationally feasible; this is the *moving horizon* variant. However, this can cause consistency problems: A sequence of actions that is started some time step t may not seem favorable any more in the next time step $t+1$ (since the horizon shifts), and thus is disrupted. We therefore also use an *almost consistent horizon variant* that takes $d=8$ in the first step, then $d=7$, and so on down to $d=2$, after which we start again with $d=8$. (Actually, we do not go down to $d=1$, since then the agent would be greedy, which can for instance disrupt consecutive runs of cooperation in the Prisoner's dilemma, see below.) A theoretically very appealing alternative is to consider the future *discounted* loss and infinite depth, which is a solution of the Bellman equations. This can be found in [6], together with more discussion and the proof of the following *optimality Theorem* for AIξ.

THEOREM 1 (PERFORMANCE OF AI$\boldsymbol{\xi}$). *If there exists a self-optimizing policy ρ in the sense that its expected average loss $\frac{1}{T}L^\rho_{1:T}$ converges for $T\to\infty$ to the optimal average $\frac{1}{T}L^\mu_{1:T}$ for all environments $\mu\in\mathcal{M}$, then this also holds for the universal policy ξ, i.e.*

$$\textit{If}\ \exists\rho\forall\mu:\ \tfrac{1}{T}L^\rho_{1:T}\ \overset{T\to\infty}{\longrightarrow}\ \tfrac{1}{T}L^\mu_{1:T}\quad\Rightarrow\quad\tfrac{1}{T}L^\xi_{1:T}\ \overset{T\to\infty}{\longrightarrow}\ \tfrac{1}{T}L^\mu_{1:T}$$

Matrix games (as defined in Section 4) can be stated in a straightforward way in our setup. We just have to consider that the opponent, i.e. the environment, does not know our action y_t when deciding its reaction x_t: $\mu_i(x_t|x_{<t},y_{1:t})=\mu_i(x_t|x_{<t},y_{<t})$. AI$\xi$ for 2×2 matrix games can then be

```
function $\ell = \mathrm{AI}\xi(s_0, x_{<t}, y_{<t}, d)$
//input: current state $s_0 = x_t$ (unknown),
//history $x_{<t}, y_{<t}$, depth $d$
$\ell^0 := \ell(0,s_0)$, $\ell^1 := \ell(1,s_0)$
If $d > 1$ Then
    For $a \in \{0,1\}$ //the agent's possible actions
        For $s \in \{0,1\}$ //the env.'s possible next states
            $\ell^{(s,a)} := \mathrm{AI}\xi(s, [x_{<t}s_0], [y_{<t}a], d{-}1)$
            $\ell^a := \ell^a + \xi(s|s_0,a,x_{<t},y_{<t}) \cdot \ell^{(s,a)}$
Return $\min\{\ell^0, \ell^1\}$
```

Figure 1: The AIξ recursion for known loss matrix ℓ.

```
function FoE $(\gamma_t)_{t\geq 1}, (\eta_t)_{t\geq 1}, (B_t)_{t\geq 1}$
//input: sequences of exploration rates $\gamma_t$, learning
//rates $\eta_t$, time control parameters $B_t$
For $\tau = 1,2,3,...$
Sample $r_\tau \in \{0,1\}$ independ. s.t. $P[r_\tau = 1] = \gamma_\tau$
If $r_\tau = 0$ Then
    Invoke subroutine FPL($\tau$):
        Sample $q_\tau^i \overset{d.}{\sim} Exp$ independently for $1 \leq i \leq n$
        Select $I_\tau^{FPL} = \arg\min_{1\leq i\leq n} \{\eta_\tau \hat{\ell}^i_{<\tau} + \ln w^i - q_\tau^i\}$
    $\langle$end of subroutine FPL($\tau$)$\rangle$
    Play $I_\tau^{FoE} := I_\tau^{FPL}$ for $B_\tau$ elementary time steps
    Set $\hat{\ell}_\tau^i = 0$ for all $1 \leq i \leq n$
Else
    Sample $I_\tau^{FoE} \in \{1...n\}$ uniformly
    Play $I := I_\tau^{FoE}$ for $B_\tau$ elementary time steps
    Let $t_0(\tau) := \sum_{\tau'=1}^{\tau-1} B_{\tau'}$ and $\ell_\tau^I := \sum_{t=t_0(\tau)+1}^{t_0(\tau)+B_\tau} \ell_t^I$
    Let $\hat{\ell}_\tau^I = \ell_\tau^I n / \gamma_t$ and $\hat{\ell}_\tau^i = 0$ for all $i \neq I$
```

Figure 2: The algorithm *FoE*. The parameters η_t, γ_t, and B_t will be specified in Theorem 2.

implemented recursively as shown in Figure 1, if we additionally assume that the environments are *Markov players* with two internal states, corresponding to the reaction x_t they are playing. (Note that in the description of the algorithm, we denote the hypothetical future states by s, as opposed to the observed history x.) Since in step t, we don't know x_t yet, AIξ must evaluate both AI$\xi(0,x_{<t},y_{<t},d)$ and AI$\xi(1,x_{<t},y_{<t},d)$ and compute a weighted mixture for both possible actions $a = 0,1$. As long as we do not yet know the loss matrix $\ell:(y_t,x_t) \mapsto [0,1]$ completely (which we assume to be deterministic), we additionally compute an expectation over all assignments of losses that are consistent with the history. To this aim, we pre-define a finite set $\mathcal{L} \subset \mathbb{N}$ that contains all possible losses. In the simulations below, we use $\mathcal{L} = \{0...4\} \cup \{-16\}$, where the actual losses are always in $\{0...4\}$ and the large negative value of -16 encourages the agent to explore as long as he doesn't know the losses completely. This is for obtaining interesting results with moderate tree depth: otherwise, when the loss observed by AIξ is relatively low, AIξ would explore only with a large depth. This phenomenon is explained in detail in Section 4.

Markov Decision Processes (MDP) have been intensively studied. In an MDP, the environmental behavior depends only on the last action and observation, precisely $\mu(x_t|x_{<t},y_{<t}) = \mu(x_t|x_{t-1},y_{t-1})$. For a 2×2 game, a Markov player is modelled by a $2\times 2\times 2$ transition matrix. It turns out that the (uncountable) class of all transition matrices with a uniform prior admits a closed-form solution:

$$\xi(x_t|x_{<t}, y_{<t}) = \frac{N^{y_{t-1}}_{x_{t-1}\to x_t} + 1}{N^{y_{t-1}}_{x_{t-1}\to 0} + N^{y_{t-1}}_{x_{t-1}\to 1} + 2},$$

where $N^{y_{t-1}}_{x_{t-1}\to x_t}$ counts how often in the history the state x_{t-1} transformed to state x_t under action y_{t-1}. This is just Laplace' law of succession [6, Prob.2.11&5.14]. (Observe that ξ is not Markov but depends on the full history.) Note that the ξ posterior estimate changes along the expectimin tree (2). Disregarding this important fact as is done in Temporal Difference learning and variants would result in greedy policies where one has to rescue exploration by ad-hoc methods (like ε-greedy). One can show that there exist self-optimizing policies $\tilde{p}$ for the class of ergodic MDPs [6]. Although the class of transition matrices contains non-ergodic environments, a variant of Theorem 1 applies, and hence the Bayes optimal policy p^ξ is self-optimizing for ergodic Markov players (which we will exclusively encounter). The intuitive reason is that the class is compact and the non-ergodic environments have measure zero.

3. ACTING WITH EXPERT ADVICE (FOE)

Instead of predicting or acting optimally with respect to a model class, we may construct an agent from a class of *base agents*. We show how this can be accomplished for fully active problems. The resulting algorithm will radically differ from the AIξ agent.

***Prediction* with expert advice** has been very popular in the last two decades. The base predictors are called experts. Our goal is to design a master algorithm which in each time step t selects one expert i and follows its advice (i.e. predicts as the expert does). Thereby, we want to keep the master's *regret* $\ell^{master}_{1:T} - \ell^*_{1:T}$ small, where $\ell^*_{1:T}$ is the cumulative loss of the best expert in hindsight up to time T. Usually, $T \geq 1$ is not known in advance. The state-of-the-art experts algorithms achieve this: Loss bounds similar to (1) can be proven, with $\ell^\mu_{1:T}$ replaced by $\ell^*_{1:T}$ and w^μ replaced by the prior weight of the best expert in hindsight, w^*. These bounds hold *in the worst case*, i.e. without any assumption on the data generating process. In particular, the environment that provides the feedback may be an adaptive adversary. Since these bounds imply bounds in expectation in the Bayesian setting (with slightly larger constants than (1)), expert advice is in a sense the stronger prediction strategy [5].

In order to protect against adaptive adversaries, we need to randomize. In this work, we build on the *Follow the Perturbed Leader FPL* algorithm introduced by [4]. (We won't discuss the more popular alternative of weighted sampling at all.) We don't even need to be told the true outcome after the master's decision. All we need for the analysis is learning the *losses* of all experts, which are bounded (this is an important restriction which applies to all standard experts algorithms) w.l.o.g. in [0,1]. In this way, the master's actual decision is based on the *past cumulative loss* of the experts. A key concept is that we must prevent the master from learning too fast (or too slowly). This is achieved by introducing a *learning rate* η_t, which decreases to zero at an appropriate rate with growing t. Most of the literature assumes experts classes of finite size n with uniform prior $\frac{1}{n}$, in particular when the learning rate η_t is non-stationary. For *FPL*, the case of arbitrary non-uniform prior and countable expert classes has been treated in [7].[1]

Active problems. In the passive *full observation game* discussed so far (i.e. we learn all losses), the notion of regret is not problematic even against an adaptive adversary.[2] However, the situation changes if the reaction of the environment depends on our past actions. Consider the simple case of two experts, one always suggesting action 0 and the other one action 1. The environment is reactive and "unfair": Each expert incurs no loss as long as we stay with its initial action (e.g. the action sequences 00000 and 111 have no loss). But as soon as we perform a different action (e.g. 001), in all subsequent rounds both experts incur loss 1. Each sensible strategy will soon explore both actions, and compared to the pure experts, we incur large loss. Consequently, we need to consider a different notion of regret: Our performance is compared to what an expert could achieve when he is actually put in our situation. In this example, after the action sequence 001, we perform badly, but so do all experts.

Another problem with reactive environments is that we do not necessarily get valid feedback for all experts in each round. In the previous example, if we chose 0 as the first action and learned that expert 0 had no loss at time $t=2$, it is not legitimate to make any assumption on the loss of expert 1 at $t=2$. Even if the environment tells us that the *pure* expert 1 had no loss, we are interested in the loss of expert 1 put in our situation, i.e. after the first action 0. But this loss we do not know. Precisely, we know only the loss of an expert with the *correct action history* after the last time step in the past, where we (maybe coincidentally) acted as he suggested. Instead of trying to track the action history (which is possibly expensive), we therefore use *only the feedback from the currently selected expert* i and discard all other information. This is commonly referred to as *bandit setup*. Fortunately, this issue can be successfully addressed by forcing exploration, i.e. sampling according to the prior, with a certain probability γ_t [10]. This *exploration rate* γ_t is decreased to zero appropriately with growing t. Thus, in each time step we decide to either *follow* the perturbed leader or *explore*. Accordingly, we call our algorithm *FoE* (Follow or Explore). Bounds for the bandit setup are typically similar to (1), but with $-\ln w^*$ replaced by (something larger than) $1/w^*$. Hence they are exponentially larger in w^*, and one can show that this is sharp in general.

[1]Given the large amount of recent literature, it should be not too difficult to obtain similar assertions for weighted sampling algorithms. However, as far as we know, the only result proven so far requires rapidly decaying weights [3], which is therefore not appropriate for universal expert classes.

[2]One can even prove the following strong statement[7, 11]: If a strategy performs well against an *oblivious* adversary which does not at all depend on our actions, then it also performs well against an adaptive adversary (in case of partial observations, this holds only for the weak regret).

Increasing horizon. If the environment is reactive, it is not sufficient to consider only the short-term performance of the selected expert i. This was first recognized by [1], who considered the repeated game of "Prisoner's Dilemma" and the "tit for tat" opponent as a motivating example (see Section 4 for details). In this case, a good long term strategy is cooperating (because of the particular opponent). However defecting is dominant, i.e. the instantaneous loss of defecting is *always* smaller than that of cooperating. So in order to notice that an expert (for instance the always cooperating one) performs well, we have to evaluate it at least over two time steps. In general, if we evaluate a chosen expert over an *increasing number of time steps*, we hope that we perform well in arbitrary reactive environments. This means that the master works at a *different time scale* τ: in its τth time step, it gives the control to the selected expert for $B_\tau \geq 1$ time steps (in the original time scale t). As a consequence, the instantaneous losses which the master observes are no longer uniformly bounded in [0,1], but in $[0,B_\tau]$. Fortunately, it turns out that the analysis remains valid if B_τ grows unboundedly but slowly enough. Only the convergence rate of the average master's loss to the optimum is affected: we will obtain a final rate of $t^{-1/10}$. The resulting algorithm *FoE* (for a finite expert class) is specified in Figure 2 together with its subroutine *FPL*. We may have instantaneous and cumulative losses in both time scales, this is always clear from the notation (e.g. ℓ_t^i vs. ℓ_τ^i). Not surprisingly, most of *FoE* works in the master time scale.

Note that *FoE* makes use of its observation *only if he decided to explore*, i.e. if $r_\tau = 1$. This seems an unnecessary waste of information, which is motivated from the analysis, since *FoE* needs an *unbiased loss estimate* $\hat{\ell}$ (with respect to *FoE*'s randomization). We just chose the simplest way to guarantee this. For the simulations, we concentrate on the following *faster learning variant*: approximate the probability p_τ^i of the selected expert i (jointly for exploration and exploitation) by a Monte-Carlo simulation. Then always learn a (close to) unbiased estimate $\hat{\ell}_\tau^i = \ell_\tau^i / p_\tau^i$. The analysis of *FoE* works in the same way for this modification. On the other hand, we will see that modified *FoE* learns faster.

In case of a non-uniform prior and possibly infinitely many experts, the exploration must be according to the prior weights. This causes another problem: *FoE*'s loss estimates $\hat{\ell}$ need to be bounded, which forbids exploring experts with very small prior weights. Hence we define for each expert i, an *entering time* $\mathcal{T}^i \geq 1$ (at the master time scale). Then *FoE* (including its subroutine *FPL*) is modified such that it uses only *active* experts from $\{i : \tau \geq \mathcal{T}^i\}$. This guarantees additionally that we have only a finite active set in each step, and the algorithm remains computationally feasible.

Theorem 2 (Performance of FoE). *Assume FoE acts in an online decision problem with bounded instantaneous losses $\ell_t^i \in [0,1]$. Let the exploration rate be $\gamma_\tau = \tau^{-1/4}$ and the learning rate be $\eta_\tau = \tau^{-3/4}$. In case of uniform prior, choose $B_\tau = \lfloor \tau^{1/8} \rfloor$. Then, for all experts i and all*

$T \geq 1$, we have

$$\begin{aligned}\tfrac{1}{T}\mathbf{E}\ell_{1:T}^{FoE} &\leq \tfrac{1}{T}\ell_{1:T}^{i} + O(n^2 T^{-1/10}), \text{ and} \\ \tfrac{1}{T}\ell_{1:T}^{FoE} &\leq \tfrac{1}{T}\ell_{1:T}^{i} + O(n^2 T^{-1/10}) \text{ w.p. } 1 - T^{-2}.\end{aligned}$$

Consequently, $\limsup_{T\to\infty}\frac{1}{T}(\ell_{1:T}^{FoE} - \ell_{1:T}^{i}) \leq 0$ a.s. For non-uniform prior, let $B_\tau = \lfloor t^{1/16} \rfloor$ and $\mathcal{T}^i = \lceil (w^i)^{-16} \rceil$. Then corresponding assertions hold with O-terms replaced by $O\ \log(w^i)T^{-1/10} + (w^i)^{-22}T^{-1}$.

The proof of this main theorem on the performance of *FoE* can be found in [12]. Similar bounds hold for larger $B_\tau < \tau^{\frac{1}{4}}$. In the simulations, we used $B_\tau = \tau^{0.24}$ for faster learning. For playing 2×2 matrix games, we will use the class of all 16 deterministic four-state Markov experts. That is, each expert consists of a lookup table with all the actions for each of the 4 possible combination of moves in the last round. In the first round, the expert plays uniformly random. (Compare the standard results on learning matrix games with expert advice by [2].)

4. SIMULATIONS

A 2×2 matrix game consists of two matrices $R_1, R_2 \in \mathbb{R}^{2\times2}$, the first one containing rewards for the row player, the second one rewards for the column player. (Recall that using losses or rewards is just a matter of convenience, in this section we will use rewards.) A *single game* proceeds in the following way: the first player chooses a row action $i \in \{0,1\}$ and simultaneously the second player chooses a column action $j \in \{0,1\}$, both players without knowing the opponent's move. Then reward $R_k(i,j)$ is payed to player k ($k = 1,2$), and i and j are revealed to both players. A repeated game consists of T single games. We chose $T = 20000$, if at least one opponent is *FoE* (which has slow learning dynamics, as we will see), and $T = 100$ for the fast learning AIξ (unless it is plotted in the same graph as *FoE*). If at least one randomized player participates, the run is repeated 10 times[3], and the average is shown.

All games we consider have rewards in $\{0...4\}$. The AIξ and *FoE* agents are used as specified in the previous sections, with the classes of all two-state Markov environments and all deterministic four-state Markov experts, respectively. For AIξ, we will concentrate the presentation on the almost consistent horizon variant, since it performs always better than the moving horizon variant. For *FoE*, we will concentrate on the faster learning variant.

Prisoner's Dilemma. This dilemma is classical. The reward matrices are $R_1 = \begin{smallmatrix} 1 & 4 \\ 0 & 3 \end{smallmatrix}$ and $R_2 = R_1^T$, with the following interpretation: The two players are accused of a crime they have committed together. They are being interrogated separately. Each player can either cooperate with the other player (don't tell the cops anything), or he defects (tells the cops everything but blame the colleague). The punishments are according to the players' joint decision: if none of them gives evidence, both get a minor sentence. If one gives evidence and the other one keeps quiet, the traitor gets free, while the other gets a huge sentence. If both give evidence, then they both get a significant sentence. (There is also an easier variant "Deadlock" which we do not discuss.)

[3]We stress again that we are only interested in the *qualitative* behavior of the simulations, i.e. we need to assess the learning dynamics only roughly. Therefore we do not need large sample sizes, and we will not complicate the graphs with error bars or other information used for precise statistics.

It is clear that giving evidence, i.e. defecting, is an instantaneously *dominant* action: regardless of what the opponent does, the immediate reward is always larger for defecting. However, if both players would agree to cooperate, this is the "social optimum" and guarantees the better long-term reward in the repeated game. A well-known instance for this case is playing against the "tit for tat" strategy strategy which cooperates in the first move and subsequently performs the action we did in the previous move. Similar but harder to learn are "two tit for tat" and "three tit for tat", which defect in the first move and cooperate only if we cooperated two respectively three times in a row. Note that although "two tit for tat" and "three tit for tat" are not in AIξ's model class, probabilistic versions of the strategies are: if the probability of "adversary defected, I cooperate, then the adversary will cooperate in the next round" is chosen correctly (namely $\frac{1}{2}$ for 2-tit for tat and ≈ 0.57 for 3-tit for tat), then the expected number of rounds I have to cooperate until the adversary will do so is 2 respectively 3.

Figure 3 shows that in most cases, AIξ learns very quickly the best actions. (This is the consistent horizon variant, the moving horizon variant will be discussed with the next game, Stag Hunt.) If the opponent is memoryless as for example the uniform random player, AIξ constantly defects after short time. Against tit for tat and two tit for tat, AIξ cooperates after short time. The figure shows the average per round rate of cooperation, which after a few exploratory moves converges to the optimal action as $\frac{1}{t}$. However, AIξ does not learn to cooperate against three tit for tat. The reason is the general problem that in order to increase exploration, AIξ needs exponential depth of the expectimin tree. Assume that a certain action sequence of length n is favorable against the true environment, which has however not too large a current weight. In this instance, cooperating three times in a row is favorable against (the probabilistic version of) 3-tit for tat. In order to recognize that this is worth exploring, AIξ has to build a branch of depth $n = 3$ in the expectimin tree, which has (because of the relatively low prior weight) very small probability $\sim \exp(-n)$ however. Then it needs an exponentially large subtree below this branch to accumulate enough (virtual) reward in order to encourage exploration.

One more problem arises when AIξ plays against another AIξ. Here, the perfectly symmetric setting results in both playing the same actions in each move, hence they are not correctly learning. We might try to remedy this by varying the tree depth of the second AIξ (denoted AIξ2 in the figure), however it turns out that in this case, both AIξ's do not learn at all to cooperate (see [6, Sec.8.5.2] for a possible reason).

We now turn to the performance of *FoE* (the faster learning variant) as evaluated in Figure 4. As expected, *FoE* learns much slower than AIξ (note the different time scale). On the other hand, its exploration is strong enough to learn 3-tit for

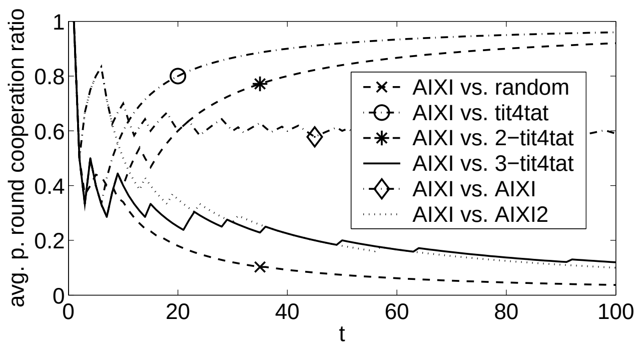

Figure 3: AIξ in Prisoner's Dilemma

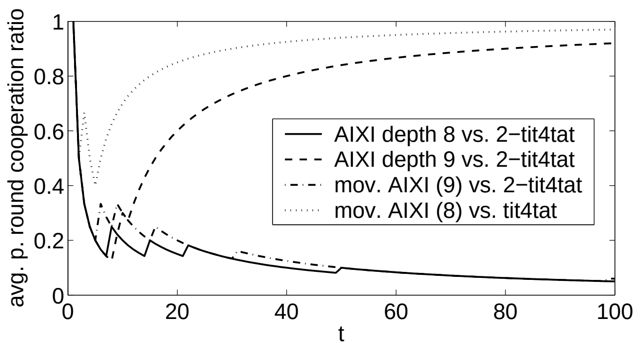

Figure 5: Stag Hunt: AIξ and its moving horizon variant

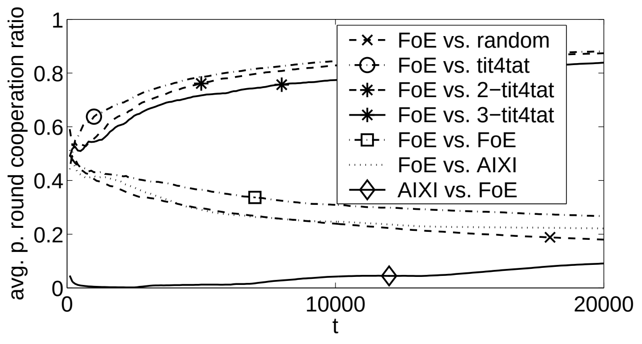

Figure 4: *FoE* (and AIξ) in Prisoner's Dilemma

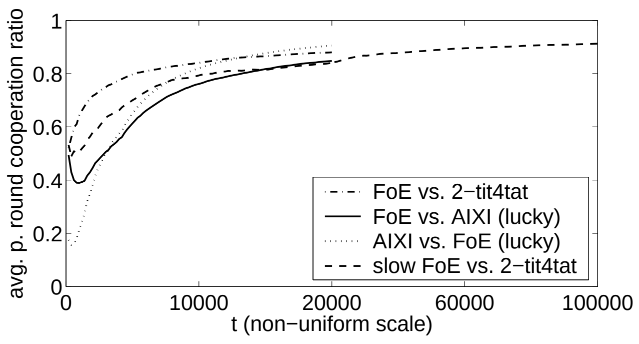

Figure 6: Stag Hunt: *FoE* and its slower learning variant

tat (and even harder opponents). When playing against another instance of *FoE*, we notice however that they usually do not succeed to overcome the dominance of mutual defection. Also when *FoE* competes with AIξ, they tend to learn mutual defection rather than cooperation. (Sometimes, they learn cooperation in one or two of the possible states of the MDP.)

Stag Hunt. This game is also known as "Assurance". The reward matrices are $R_1 = \begin{smallmatrix}2&3\\0&4\end{smallmatrix}$ and $R_2 = R_1^T$. Two players are hunting together. If they cooperate, they will catch the stag. However, one player might not trust the other, in which case he chases rabbits on his own instead. In this case, the other one won't get anything if he tries to cooperate. If both defect, then they are in conflict, and each player gets less rabbits. Although the optimum for both players is to cooperate, they need to trust each other sufficiently. If one player plays uniformly random, it is better for the other to go for the rabbits. Also, defecting has the lower variance.

Maybe it is surprising to observe that AIξ (with a depth of 8) does not learn to cooperate against 2-tit for tat (Figure 5). The reason is that defecting has a relatively good payoff, and therefore exploration is not encouraged as discussed in the previous subsection. If the depth of the tree is increased to 9, AIξ learns cooperation against 2-tit for tat (but not against 3-tit for tat). We also see that the moving horizon variant of AIξ has even more problems with exploration: It does not learn cooperating against 2-tit for tat, even with depth 9. The explanation is that even if AIξ decides to explore in one time step, in the next step this exploration might not be correctly continued, as the tree is now explored to a different level. This observation can also be made for the Prisoner's Dilemma. In fact, the consistent horizon variant performs *always* better than moving horizon.

As before, *FoE* learns much slower but explores more robustly (Figure 6), neither 2- nor 3-tit for tat are a problem. Unlike in the Prisoner's Dilemma, if AIξ and *FoE* are competing, they learn mutual cooperation in almost half of the cases, an average over such lucky instances is given in the figure. The same is valid for *FoE* against *FoE*, while AIξ against AIξ has the same symmetry problem as already observed in the Prisoner's Dilemma. The original slower learning variant of *FoE* reaches the same average level of performance only after 10^5 time steps instead of $2\cdot 10^4$ steps, and moreover with a variance twice as high.

Chicken. The reward matrices $R_1 = \begin{smallmatrix}0&4\\1&2\end{smallmatrix}$ and $R_2 = R_1^T$ of the "Chicken" game (also known as "Hawk and Dove") can be interpreted as follows: Two coauthors write a paper, but each tries to spend as little effort as possible. If one succeeds to let the other do the whole work, he has a high reward. On the other hand, if no one does anything, there will be no paper and thus no reward. Finally, if both decide to cooperate, both get some reward. Here, in the repeated game, it is socially optimal to take turns cooperating and defecting.[4] Still the best situation for one player is if he

[4]We assume that the authors are not very good at cooperating, and that the costs of cooperating more than compensate

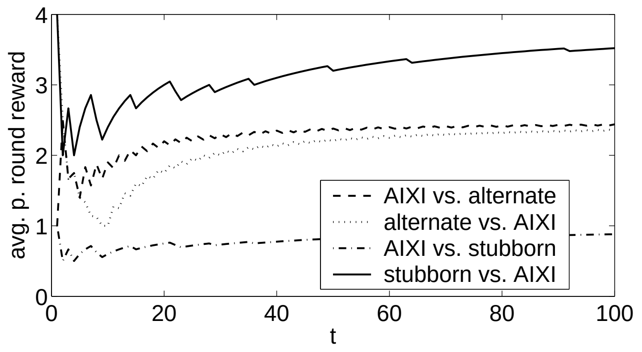

Figure 7: AIξ in the Chicken game

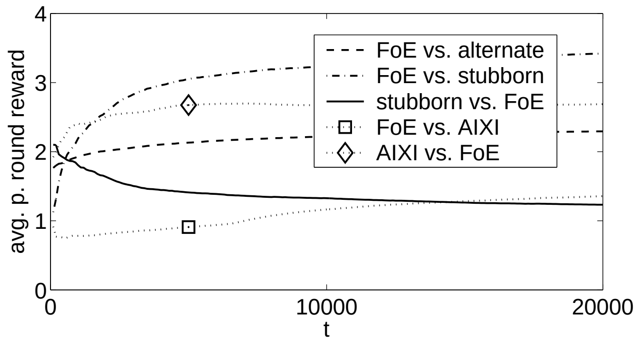

Figure 8: *FoE* (and AIξ) in the Chicken game

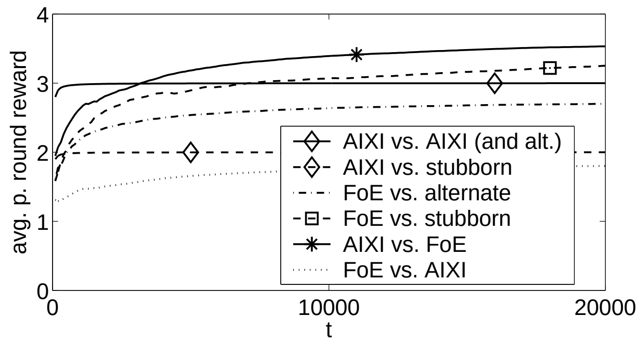

Figure 9: AIξ and *FoE* in Battle of Sexes

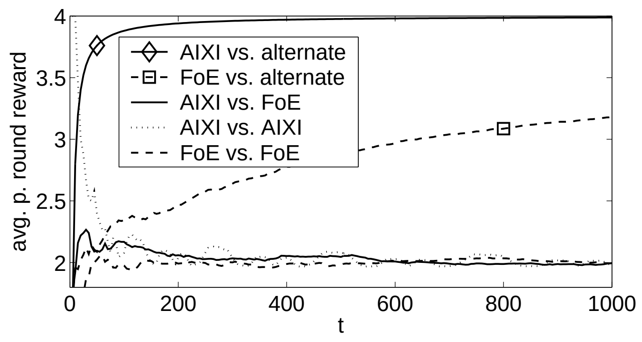

Figure 10: AIξ and *FoE* in Matching Pennies

emerges as the "dominant defector", defecting in most or all of the games, while the other one cooperates.

If the opponent steadily alternates between cooperating and defecting, then AIξ quickly learns to adapt. This can be observed in Figure 7, where the performance is given in terms of average per round reward instead of cooperation rate. However, AIξ is not obstinate enough to perform well against a "stubborn" adversary that would cooperate only after his opponent has defected for three successive time steps. Here, AIξ learns to cooperate, leaving his opponent the favorable role as the dominant defector. (However, AIξ learns to dominate the less stubborn adversary which cooperates after two defecting actions.) When two AIξs play against each other, they again have the symmetry problem. Interestingly, if we break symmetry by giving the second AIξ a depth of 9, he will turn out the dominant defector (not shown in the graph).

FoE behaves differently in this game (Figure 8). While he learns to deal with the steadily alternating adversary and emerges as the dominant defector against the stubborn one, he would give precedence to AIξ in most cases. This is not hard to explain, since *FoE* in the beginning plays essentially random. Thus AIξ learns quickly to defect, and for *FoE* remains nothing but learning to cooperate. However, this does not always happen: In the minority of the cases, *FoE* defects enough such that AIξ decides to cooperate, and *FoE* will be the dominant defector. (Hence the average shown in the graph is less clear in favor of AIξ.)

for the synergy. We could assign a reward of 3 instead of 2 to mutual cooperation. This is the less interesting situation of "Easy Chicken", where cooperating is the optimal long-term strategy like in the previous games.

Battle of Sexes. In this game, a married couple wants to spend the evening together, but they didn't settle if they would go to the theater (her preference) or the pub (his preference). However, if they fail to meet, both have a boring evening (and no reward at all). The reward matrices are $R_1 = \begin{smallmatrix}2 & 0\\ 0 & 4\end{smallmatrix}$ and $R_2 = \begin{smallmatrix}4 & 0\\ 0 & 2\end{smallmatrix}$. Coordination is clearly important in the repeated game. Like in "Chicken", taking turns is a social optimum, while it is best for one player if his choice becomes dominant.

In Figure 9, our universal learners show similar performance like in the Chicken game. Both learn to deal with an alternating partner. *FoE* also learns to dominate over a stubborn adversary which plays his less favorite action only after the opponent insists three times on that. AIξ is dominated by this stubborn player. However, AIξ always dominates *FoE*. Finally, in contrast to the Chicken game, AIξ against AIξ does not have the symmetry problem, but they both learn to alternate.

Matching Pennies. Each player conceals in his palm a coin with either heads or tails up. They are revealed simultaneously. If they match (both heads or both tails), the first player wins, otherwise the second. This is the only zero-sum game of the games we consider, where $R_1 = \begin{smallmatrix}4 & 0\\ 0 & 4\end{smallmatrix}$ and $R_2 = \begin{smallmatrix}0 & 4\\ 4 & 0\end{smallmatrix}$. Thus, there is a minimax strategy for both players, which is actually uniform random play. On the other hand, deterministic repeated play is potentially exploitable by the adversary.

Figure 10 shows the results for this last game we discuss. Both AIξ and *FoE* learn to exploit a predictable adversary, namely the player alternating between 0 and 1. The other games are balanced in the long run, only in the beginning AIξ succeeds to exploit *FoE* a little. If two AIξs compete, it is important to break symmetry, then both learn to alternate (this situation is shown in the graph). If symmetry is not broken, the row player (who tries to match) always wins.

5. DISCUSSION

In principal, universal agents perform well in repeated matrix games. They usually learn to prefer the optimal long-term action to greedy behavior (Prisoner's Dilemma and Stag Hunt). If possible they are able to exploit a predictable adversary (Matching Pennies). And they learn good strategies when it is necessary to foresee the opponent's action (Chicken and Battle of Sexes). Of the two approaches we presented and compared, AIξ learns much faster than *FoE*, but *FoE* explores more thoroughly. Of course, there is a trade-off between exploration and fast learning. Interestingly, it may depend on the adversary (and thus on the environment) if fast learning or exploration is the better *long-term* strategy: In Chicken and Battle of Sexes, AIξ profits against *FoE* by learning fast and dictating its preferred action, but looses against the stubborn opponent because of not exploring enough.

6. REFERENCES

[1] D. P. de Farias and N. Megiddo. How to combine expert (and novice) advice when actions impact the environment? In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[2] Y. Freund and R. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–193, 1999.

[3] C. Gentile. The robustness of the p-norm algorithm. *Machine Learning*, 53(3):265–299, 2003.

[4] J. Hannan. Approximation to Bayes risk in repeated plays. In M. Dresher, A. W. Tucker, and P. Wolfe, editors, *Contributions to the Theory of Games 3*, pages 97–139. Princeton University Press, 1957.

[5] M. Hutter. Online prediction – Bayes versus experts. Technical report, http://www.idsia.ch/~marcus/ai/bayespea.htm, July 2004. Presented at the *EU PASCAL Workshop on Learning Theoretic and Bayesian Inductive Principles (LTBIP-2004)*.

[6] M. Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability.* Springer, Berlin, 2004. 300 pages, http://www.idsia.ch/~marcus/ai/uaibook.htm.

[7] M. Hutter and J. Poland. Adaptive online prediction by following the perturbed leader. *Journal of Machine Learning Research*, 6:639–660, 2005.

[8] L. A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9:265–266, 1973.

[9] A. K. McCallum. Instance-based utile distinctions for reinforcement learning with hidden state. In *Proc. 12th International Conference on Machine Learning*, pages 387–395, 1995.

[10] H. B. McMahan and A. Blum. Online geometric optimization in the bandit setting against an adaptive adversary. In *17th Annual Conference on Learning Theory (COLT)*, volume 3120 of *Lecture Notes in Computer Science*, pages 109–123. Springer, 2004.

[11] J. Poland. FPL analysis for adaptive bandits. In *3rd Symposium on Stochastic Algorithms, Foundations and Applications (SAGA'05)*, pages 58–69, 2005.

[12] J. Poland and M. Hutter. Defensive universal learning with experts. In *Proc. 16th International Conf. on Algorithmic Learning Theory (ALT'05)*, volume 3734 of *LNAI*, pages 356–370, Singapore, 2005. Springer, Berlin.

[13] J. Schmidhuber. Optimal ordered problem solver. *Machine Learning*, 54(3):211–254, 2004.

[14] J. Schmidhuber, J. Zhao, and M. Wiering. Shifting inductive bias with success-story algorithm, adaptive Levin search, and incremental self-improvement. *Machine Learning*, 28:105–130, 1997.

[15] R. J. Solomonoff. A formal theory of inductive inference: Part 1 and 2. *Inform. Control*, 7:1–22, 224–254, 1964.