# Recent Results in Universal and Non-Universal Induction

Jan Poland[⋆]

Graduate School of Information Science and Technology
Hokkaido University, Japan
jan@ist.hokudai.ac.jp
http://www-alg.ist.hokudai.ac.jp/~jan

**Abstract.** We present and relate recent results in prediction based on countable classes of either probability (semi-)distributions or base predictors. Learning by Bayes, MDL, and stochastic model selection will be considered as instances of the first category. In particular, we will show how analog assertions to Solomonoff's universal induction result can be obtained for MDL and stochastic model selection. The second category is based on prediction with expert advice. We will present a recent construction to define a universal learner in this framework.

## 1 Introduction

Consider the following general *online induction setup*. An *agent* (in the following also called *learner* or *predictor*) and an *environment* play a game that proceeds in rounds $t = 1, 2, \dots$ In each round $t$, the environment first provides some input $z_t \in \mathcal{Z}$ to the agent, where $\mathcal{Z}$ is the input space. The input will not be relevant to any result presented in this paper, consequently $\mathcal{Z} = \{\varnothing\}$ is possible. Input is however convenient to state important applications such as pattern classification. After seeing the input $z_t$, the agent and the environment *simultaneously* fix a decision $d_t \in \mathcal{D}$ and a current state $s_t \in \mathcal{S}$, respectively, such that no player knows the other's move while making his own move. Here, $\mathcal{D}$ is the space of possible decisions of the agent, and $\mathcal{S}$ the space of possible states of the environment. Now the environment learns the decision of the agent $d_t$ and provides an observation $x_t = G(s_t, d_t)$ to the agent. The observation function $G$ may depend on the current state and decision and possibly on some additional randomness. Finally, the current performance of the agent is measured in terms of a loss function $\ell_t = \ell(s_t, d_t)$, which may depend on the current state and the agent's decision. Note that the agent does not necessarily get to know its own loss $\ell_t$. Throughout this paper, we will assume *uniformly bounded* losses, i.e. (possibly after rescaling) $\ell : \mathcal{S} \times \mathcal{D} \to [0, 1]$.

We will focus on two variants of this induction framework. First, cases will be considered where the current decision $d_t$ is meant as an estimate of the current state $s_t$. Here, the agent will work with a base class of possible stochastic

[⋆] This work was supported by JSPS 21st century COE program C01.

environments, one of which is the data generating or *true* one. According to the observations, the agent updates its belief state by using Bayes rule. This setup is subject of Section 2.

In the other setup, which is studied in Section 3, the agent directly tries to minimize the cumulative loss. Here we may assume that we play against an *adversary* who chooses the state $s_t$ (and even the input $z_t$) in order to maximize the agent's cumulative loss $L_{1:T} := \sum_{t=1}^{T} \ell_t$, while of course the agent aims to minimize this quantity. We consider the *worst case*, without any assumption on the data generating process. Both observation function and loss function are assumed to be deterministic in this setup. The agent has access to the recommendations of a class of base agents or *experts* and follows the advice of one of them (if the decision space and the loss function are convex, it could also follow a mixture recommendation). Experts are weighted according to their past performance. All performance guarantees for the agent will be relative to the expert class.

Although our online induction framework is already quite general, still extensions and generalizations are thinkable. For example, the input $z_t$ could be chosen by the agent instead of the environment. This would enhance the possible applications to *active learning*, where the aim is not only a small cumulative loss, but also some guaranteed quality of the learner's hypothesis after a certain (small) amount of time steps. Another modification is considering loss functions which depend on the actual observation $x_t$ instead of (or additionally to) the state $s_t$. Later (Theorem 9), we will see how a bound on such a cumulative loss $\sum \ell(x_t, d_t)$ follows from a bound on a suitable cumulative loss $\sum \ell(s_t, d_t)$ in a Bayesian setting. In the experts setting, since the observation function is deterministic, a bound on $\sum \ell(x_t, d_t)$ is always weaker than a bound on $\sum \ell(s_t, d_t)$.

In both variants, induction based on a model class and on an expert class, it is possible to work with *universal* classes which are defined by some fixed universal Turing machine. We will show below how to obtain such constructions (Definitions 2 and 10). This introduction is concluded by discussing some standard problem setups in our framework.

**Pattern classification**. This is one of the most commonly studied problems in machine learning. In case of binary classification, the state is simply the probability that the observation will be one, depending on the input pattern (note that we allow stochastic concepts). In the multi-class case we have a probability vector instead. Although classification has been also studied in the experts framework [1], it is usually treated by considering a model class (e.g. linear separators, then classification can be done by support vector machines). In this case, with stochastic concepts, the agent's decision $d_t$ is an estimate of the true probability $s_t$, and the loss $\ell_t$ often measures their difference.

**Universal Induction**. Solomonoff [2] has studied sequence prediction (without inputs, i.e. $\mathcal{Z} = \{\varnothing\}$) based on a universal model class. We will discuss his construction below (Definition 2 and Theorem 6). Parallel constructions can be obtained for a Minimum Description Length (MDL) predictor (Theorem 7) and a learner based on stochastic model selection (Theorem 8), the latter however has no (direct) finite bound for universal model class.

**Bandit Problems**. In this setup, the agent decides to pull one "arm" of a $K$-armed bandit and learns only the loss of the selected arm, but not the losses of the alternative arms. This is a basic instance of the exploration-exploitation tradeoff problem and has been studied both in the model-based framework [3] and the expert framework [4].

**Geometric online decision** problems are characterized by the fact that the loss is linear in the current state and decision, $\ell_t = \langle s_t, d_t \rangle$. Based on [5], Kalai and Vempala [6] show a very elegant way to treat these problems, on which our methods in Section 3 build.

**"Active" decision making agents** can be based on either model or expert classes. They are characterized by the fact that they are guaranteed (or at least expected) to perform well also in certain cases where the future behavior of the environment depends on the agent's decision (*reactive problems*). We will present a construction with a universal expert class in Section 3, for a construction with a model class we point to [7].

## 2 Model Classes and Bayes Learning

Let the observation space $\mathcal{X} = \{1 \ldots |\mathcal{X}|\}$ be a finite alphabet ($\mathcal{X} = \{0,1\}$ in case of binary alphabet). Then, in case that there are inputs, a *probability distribution* $\nu : \mathcal{Z} \ni z \mapsto \big(\nu(x|z)\big)_{x\in\mathcal{X}} \in [0,1]^{|\mathcal{X}|}$ is a function which assigns each input to a probability vector over $\mathcal{X}$, such that $\sum_{x\in\mathcal{X}} \nu(x|z) = 1$ holds for all $z \in \mathcal{Z}$. That is, for fixed input, the distributions are independently and identically distributed (i.i.d.). In the case of sequence prediction, i.e. if there are no inputs, we conditionalize on the history instead of the inputs an consider arbitrary non-i.i.d. *measures* and *semimeasures*: For any history of observations $x_{<t} = x_{1:t-1} = x_1 \ldots x_{t-1}$, the predictive (semi-)probability vector of $\nu$ is $\big(\nu(x|x_{<t})\big)_{x\in\mathcal{X}} = \big(\frac{\nu(x_{<t}x)}{\nu(x_{<t})}\big)_{x\in\mathcal{X}}$, where $\nu(\epsilon) \leq 1$ and $\nu(x_{<t}) \geq \sum_{x\in\mathcal{X}} \nu(x_{<t}x)$ for the empty string $\epsilon$ and any history $x_{<t} \in \mathcal{X}^*$ ($\mathcal{X}^*$ denotes the set of strings over $\mathcal{X}$). With equality in these inequalities, we call $\nu$ a measure, otherwise a semimeasure.

The reason to study semimeasures at all is that the universal model class (Definition 2 below) contains semimeasures. We could also consider semidistributions and conditionalize w.r.t. the history in the classification setup (with inputs), however this makes the presentation more complicated.

Now, we consider a *countable model class* $\mathcal{C} = \{\nu_1, \nu_2, \ldots\}$ of probability distributions or semimeasures, respectively. Each $\nu \in \mathcal{C}$ is assigned a prior weight $w_\nu \in (0,1)$ such that $\sum_{\nu\in\mathcal{C}} w_\nu \leq 1$.

*Example 1.* Consider a binary classification problem with $\mathcal{Z} = \mathbb{R}^2$ and $\mathcal{C} \cong \mathbb{Q}^2$, such that each model corresponds to a separating line with rational coefficients on the plane. For some $(q_1, q_2) \cong \nu \in \mathcal{C}$, we can set $w_\nu = 2^{-l(q_1)-l(q_2)}$, where $l(q)$ is the number of bits needed to specify a rational number $q$ in a prefix-code.

In the rest of this section, we will drop the inputs $z_t$ from the notation and restrict to the sequence prediction case. Please keep in mind that all results equally apply to the classification setup with inputs. Next, we define a *universal model class.*

**Definition 2.** (Universal Model Class) *Consider a function $f : \{0,1\}^* \to \mathcal{X}^*$ which is* monotone*, that is, if a string $x \in 0,1^*$ is a prefix of another string $x' \in 0,1^*$, then also $f(x)$ is a prefix of $f(x')$ [8, Def. 4.5.2, Def. 4.5.3]. Fix a reference monotone Turing machine, then each monotone function $f$ corresponds to a program on this machine. With $l(f)$ denoting the length of this program, we may set $w(f) = 2^{-l(f)}$ (assume that the program tape is binary). According to [8, Theorem 4.5.2], each such $f$ corresponds to a semimeasure $\nu \in \mathcal{M}$, where $\mathcal{M}$ is the set of all lower semicomputable (aka. enumerable) semimeasures. This defines $w_\nu$ for each $\nu \in \mathcal{M}$. It is possible to effectively enumerate $\mathcal{M}$ using the reference machine.*

Given a model class $\mathcal{C}$ together with the prior weights and a string of observations $x_{<t}$, we can use Bayes' rule to define posterior weights. Then, a predictor can use these posterior weights for predicting in three different ways: Bayes mixture, MDL, or stochastic model selection.

**Definition 3.** (Bayes mixture) *For a class $\mathcal{C}$ consisting of (semi-)measures, a string $x_{<t}$, and a character $x \in \mathcal{X}$, let $\xi(x_{<t}) = \sum_{\nu \in \mathcal{C}} w_\nu \nu(x_{<t})$ and $\xi(x|x_{<t}) = \frac{\xi(x_{<t}x)}{\xi(x_{<t})}$.*

One can show [8, Theorem 4.5.3] that the Bayes mixture $\xi(x_{<t})$ is, within a multiplicative constant, equal to the *Solomonoff prior*, which is the probability of obtaining $x_{<t}$ when running the reference monotone machine on an input of fair coin flips. Moreover, $\xi(x_{<t})$ is a semimeasure and lower semicomputable, thus a *universal element* of $\mathcal{M}$, and the $\xi$-predictions are approximable. However, $\xi$ and its predictions are not computable, i.e. they cannot be approximated by a program on the reference machine while knowing the approximation quality.

**Definition 4.** (MDL, [9]) *Let $\mathcal{C}$ be a class containing (semi-)measures, $x_{<t}, \tilde{x}_{<\tilde{t}} \in \mathcal{X}^*$, and $x \in \mathcal{X}$. Define*

$$\begin{aligned}
\varrho(x_{<t}) &= \max_{\nu \in \mathcal{C}} w_\nu \nu(x_{<t}) \text{ (MDL estimator)},\\
\varrho^{\tilde{x}_{<\tilde{t}}}(x_{<t}) &= w_{\tilde{\nu}} \tilde{\nu}(x_{<t}) \text{ where } \tilde{\nu} = \arg\max_{\nu \in \mathcal{C}} w_\nu \nu(\tilde{x}_{<\tilde{t}}),\\
\varrho(x|x_{<t}) &= \frac{\varrho(x_{<t}x)}{\varrho(x_{<t})} \text{ (dynamic MDL prediction)},\\
\varrho_{\text{norm}}(x|x_{<t}) &= \frac{\varrho(x_{<t}x)}{\sum_{a \in \mathcal{X}} \varrho(x_{<t}a)} \text{ (normalized dynamic MDL predictor)},\\
\varrho^{\text{static}}(x|x_{<t}) &= \frac{\varrho^{x_{<t}}(x_{<t}x)}{\varrho(x_{<t})} \text{ (static MDL prediction)},\\
\varrho^{\text{static}}_{\text{norm}}(x|x_{<t}) &= \frac{\varrho^{x_{<t}}(x_{<t}x)}{\sum_a \varrho^{x_{<t}}(x_{<t}a)} \text{ (normalized static MDL predictor)}.
\end{aligned}$$

Note that this definition is two-part MDL in the sense of the construction principle, choosing a model that simultaneously minimizes the description length of the model plus that of the data given the model. Thus, $\varrho$ coincides with a maximum a posteriori (MAP) estimator. Sometimes, the term MDL is used for more specific constructions, in particular with specific priors, or for a coding which avoids a redundancy arising in our construction [10]. Further observe that *dynamic MDL predictions* select a different model for each possible continuation of the currently observed string $x_{<t}$. Thus they are in practice (for finite model class) computationally more expensive than *static MDL prediction*, which use only the current string $x_{<t}$ for model selection. For the universal model class, none of the variants is computable. The MDL estimator $\varrho(x_{<t})$ is lower semi-computable, hence the dynamic MDL predictions are approximable. For the static MDL predictions, approximability is not obvious (and maybe not satisfied). Finally note that the normalized variants are constructed in order to define measures on $\mathcal{X}$.

**Definition 5.** (Stochastic model selection) *For a class $\mathcal{C} = \{\nu_1, \nu_2, \ldots\}$ containing only proper* measures*, a string $x_{<t}$, and a character $x \in \mathcal{X}$, the stochastic model selection predictor $\Xi$ samples a model according to the current (posterior) weights and uses this model for prediction, i.e.*

$$\Xi(x|x_{<t}) = \nu_J(x|x_{<t}) \text{ where } \mathbf{P}(J = i) \sim w_{\nu_i}\nu_i(x_{<t}).$$

We do not define stochastic model selection for semimeasures, as the bounds in Theorem 8 are infinite for the universal model class (Definition 2). This is because the entropy $\mathcal{H}(\mathcal{M}) = -\sum_{\nu\in\mathcal{M}} w_\nu \log w_\nu$ of this model class is infinite, which can be seen as follows: For each string $b \in \{0,1\}^*$, let $K(b)$ be its prefix Kolmogorov complexity [8]. To each such $b$, we can find a monotone program of length $K(b) + O(1)$ bits that generates $b$ and afterwards random coin flips. But then, $\mathcal{H}(\mathcal{M}) = \Omega\big(\sum_{b\in\{0,1\}^*} K(b)2^{-K(b)}\big) = \infty$, since $K(b)$ is the shortest possible coding of $b$ save for an additive constant. On the other hand, replacing the universal prior weights $w(f) = 2^{-l(f)}$ by the slightly smaller weights $\tilde{w}(f) = 2^{-l(f)}/l(f)^2$, we can define a univseral stochastic model selection algorithm with finite bound.

All of the above defined prediction methods have bounds on the expected cumulative difference of the predictive probabilities to the true probabilities, provided that there is a true data generating distribution $\mu$ in the model class, $\mu \in \mathcal{C}$. The *state* $s_t$ from the general framework in the introduction then becomes the current probability vector, $s_t = \mu(\cdot|x_{<t})$. For a predictive distribution $\varphi(\cdot|x_{<t})$, depending on the history (and possibly some additional randomness), we define the instantaneous quadratic and Hellinger distance, respectively:

$$q_t(\varphi, x_{<t}) = \sum_{x\in\mathcal{X}} \big(\varphi(x|x_{<t}) - \mu(x|x_{<t})\big)^2 \text{ and}$$
$$h_t(\varphi, x_{<t}) = \sum_{x\in\mathcal{X}} \big(\sqrt{\varphi(x|x_{<t})} - \sqrt{\mu(x|x_{<t})}\big)^2.$$

Then the expected cumulative quadratic and Hellinger distance are given by $Q_{1:T}(\varphi) = \sum_{t\leq T} \mathbf{E} q_t(\varphi, x_{<t})$ and $H_{1:T}(\varphi) = \sum_{t\leq T} \mathbf{E} h_t(\varphi, x_{<t})$, where the expectation is w.r.t the true distribution $\mu$ over the possible outcomes of $x_{<t}$ and the possible randomness of the predictor. Then we have the following performance guarantees for our previously defined predictors.

**Theorem 6.** (Bayes mixture performance guarantee, [2]) *Let $\mathcal{C}$ be a class of (semi-)measures and assume that there is a proper measure $\mu \in \mathcal{C}$ which generates the data. Then,*

$$Q_{1:T}(\xi) \leq \log w_\mu^{-1} \text{ and } H(\xi)_{1:T} \leq \log w_\mu^{-1}$$

*hold for all $T \geq 1$.*

Like the following result for MDL, the theorem applies to the universal model class from Definition 2. Moreover recall that the theorem and all following also apply to the classification setup, i.e. in the presence of inputs.

**Theorem 7.** (MDL performance guarantees, [9]) *Let $\mathcal{C}$ contain (semi-)measures and assume that there is a true measure $\mu \in \mathcal{C}$. Then,*

$$Q_{1:T}(\varrho) = O(w_\mu^{-1}) \text{ and } H(\varrho)_{1:T} = O(w_\mu^{-1})$$

*hold for any $T \geq 1$. The same is valid for all other MDL predictors $\varrho_{\text{norm}}$, $\varrho^{\text{static}}$, and $\varrho^{\text{static}}_{\text{norm}}$.*

The bound for MDL is exponentially larger than that for the mixture. One can show that this is sharp in general, even for classes containing only Bernoulli distributions [11]. This makes the bound practically irrelevant for all but very small model classes. Thus it motivates to select the prior more carefully when using MDL, resulting in better bounds [10, 11]. We will encounter a similarly exponential bound in Theorem 13, however for a different reason.

**Theorem 8.** (Stochastic model selection performance guarantee, [12]) *Let $\mathcal{C}$ contain proper measures and assume that there is a true measure $\mu \in \mathcal{C}$. Then, $Q_{1:T}(\Xi)$ and $H_{1:T}(\Xi)$ are both*

$$O\big(\Pi \cdot \log w_\mu^{-1}\big)$$

*for any $T \geq 1$. Here, $\Pi$ is the $\mu$-*entropy potential *defined as*

$$\Pi\big((w_\nu)_{\nu\in\mathcal{C}}\big) = \sup\big\{\mathcal{H}\big((\tfrac{\tilde{w}_\nu}{\sum_{\nu'}\tilde{w}_{\nu'}})_\nu\big) : \tilde{w}_\mu = w_\mu \wedge \tilde{w}_\nu \leq w_\nu \ \forall \nu \in \mathcal{C} \setminus \{\mu\}\big\}$$

.

One can show that a finite bound on $Q_{1:T}(\varphi)$ (or $H_{1:T}(\varphi)$) implies that the $\varphi$-predictive probabilities converge to the true $\mu$-probabilities almost surely. This nice property therefore holds for all of the above prediction methods. Moreover, consider the situation that the learner's aim is to minimize its expected loss which is a known function $\ell(x_t, \hat{x}_t)$ depending on the outcomes $x_t \in \mathcal{X}$ (rather than the state $s_t$, compare the introduction) and its estimate $\hat{x}_t = d_t \in \mathcal{X}$. Then the learner, which has access to a current belief probability $\varphi(\cdot|x_{<t})$ may choose its prediction $\hat{x}_t$ in a *Bayes optimal* way, i.e. $\hat{x}_t = \arg\min_{\hat{x}} \sum_x \varphi(x|x_{<t})\ell(x, \hat{x})$. In this case, finite bounds on the cumulative Hellinger distance imply $O(\sqrt{t})$ bounds on the loss, for arbitrary bounded loss function $\ell : (x_t, \hat{x}_t) \mapsto [0, 1]$ (the proof can be found in [9, Lemma 24–26]).

**Theorem 9.** (Loss bounds) *Assume $\mathcal{D} = \mathcal{X}$ and the losses $\ell(x_t, \hat{x}_t) \in [0, 1]$ depend on the outcome. Then a predictor $\varphi$, knowing this loss function and making Bayes optimal predictions w.r.t. it's belief probability, has the loss bound*

$$\mathbf{E}L^{\varphi}_{1:T} \leq \mathbf{E}L^{\mu}_{1:T} + 2H_{1:T}(\varphi) + 2\sqrt{2H_{1:T}(\varphi)\mathbf{E}L^{\mu}_{1:T}},$$

*where $L^{\mu}_{1:T}$ is the cumulative loss of the Bayes optimal predictor using the true probabilities.*

## 3 Expert Classes

Theorem 9 is a bridge to the other type of learning algorithm considered in this paper, an *agent based on a class of experts*. If there is a known loss function, each model $\nu \in \mathcal{C}$ recommends a decision $d^{\nu}_t \in \mathcal{X}$ in each round $t$, namely the Bayes optimal one. Hence we may randomly select a model using the regret minimization algorithm FPL (Definition 11 below) based on the past losses. Then we may apply a suitable performance guarantee in the spirit of Theorem 13, such as [13, Theorem 9] for countable model class. This gives essentially the assertion of Theorem 9 without expectation over $\mu$ (but with expectation over the agent's randomization) *for arbitrary observations* $x_1, x_2, \ldots$, with no assumptions (!) on the data generating process necessary. By an application of Jensen's inequality, the assertion also follows in $\mu$-expectation (as observed by M. Hutter).

Regret minimization algorithms like FPL are always applicable with a class of base experts $\mathcal{E} = \{e_1, e_2, \ldots\}$, each $e \in \mathcal{E}$ suggesting a decision $d^e_t$ in each round $t$: we only need to observe the experts' *losses* $\ell(s_t, d^e_t)$ (at least some losses from time to time). Parallel to the model class setup, each expert $e$ is assigned a prior weight $w_e > 0$ such that $\sum_{e\in\mathcal{E}} w_e = 1$. We may restrict to the case that state $s_t$ is the vector of all experts' current losses, $s_t = (\ell^e_t)_{e\in\mathcal{E}}$ [6]. Recall that the observation function $G$ is assumed to be deterministic. Performance guarantees (loss bounds) then usually hold in the worst case, without any assumption on the data generating process. However, the experts algorithms do *not* estimate probabilities, and they are likely to yield inferior predictions to a Bayesian method in cases where accurate estimates of the probabilities are possible and beneficial.

This section sketches construction and performance guarantees of an expert algorithm which works with the *universal expert class* $\mathcal{P}$ from Definition 10 below. The results are mainly from (and extensions of) [14, 15]. There is a huge literature of experts algorithms with different purpose which cannot be dealt with here.

**Definition 10.** (Universal expert class) *Let $\mathcal{D}$ be a finite or countable decision space. Fix a reference monotone universal Turing machine with binary output alphabet and let $\mathcal{P} = \{p_1, p_2, \ldots\}$ be an enumeration of all programs. Then $\mathcal{P}$ defines an expert class, and each program $p \in \mathcal{P}$ is an expert with prior weight $w_p = 2^{-l(p)}$: If the agent selects to follow $p$'s recommendation, it runs $p$ on the complete observation history, until the output tape contains the prefix code of a decision $d_t^p \in \mathcal{D}$. If this never happens, we define that the expert chooses a fixed decision $d_1 \in \mathcal{D}$. In this case, the computation does not halt, and (like the above learners for universal model class) the overall agent is not computable. It is however easy to impose an additional constraint on the computation time, e.g. aborting the computation after at maximum* $\exp(t)$ *steps (hence we have more time in later rounds).*

In order to simplify notation, we abbreviate the instantaneous and cumulative loss of each expert $e \in \mathcal{E}$ as $\ell_t^e$ and $L_{1:T}^e$, respectively.

**Definition 11.** (FPL: Follow the Perturbed Leader algorithm with full observation) *At time $t$, sample a perturbation $r_t^e$ independently from the exponential distribution for each expert $e \in \mathcal{E}$. With $\eta_t > 0$ being the learning rate, follow the recommendation of the expert with the best perturbed score, $\tilde{e} = \arg\min\{\eta_t L_{<t}^e - \log w_e - r_t^e : e \in \mathcal{E}\}$, that is, $d_t = d_t^{\mathrm{FPL}} = d_t^{\tilde{e}}$.*

Observe that, in case that experts are identified with models, FPL is a stochastic model selection algorithm with a non-Bayesian posterior, i.e. different from Definition 5. In contrast to Bayesian stochastic model selection, FPL usually does *not* provide predictive probabilities, but is designed for minimizing the cumulative regret.

The performance guarantee for this FPL algorithm states that the *regret*, which is the difference $\mathbf{E}L_{1:T}^{\mathrm{FPL}} - \mathbf{E}L_{1:T}^e$ (expectations are w.r.t. FPL's randomization) is bounded by $O(\sqrt{T})$ for *any reference expert $e$ and any sequence of loss vectors $s_1, s_2, \ldots$* This means, that the average per-round regret (divided by $T$) goes to zero in expectation and, by the Borel-Cantelli-Lemma, even with probability one. Of course, regret is measured against the *actual* performance of an expert. In certain *reactive* cases, where the future actions of the environment depend on the current decision, some strong experts could display an unnecessarily weak performance. An example is the repeated game of Prisoner's dilemma where the opponent plays the "tit-for-tat" strategy: Although defecting is the dominant action, cooperating is the better long term strategy [16]. A simple work-around giving better results in this and similar cases is to play a selected expert not for just one, but for $B_t$ time steps, with $B_t$ increasing in $t$. Hence, the expert selection or *master algorithm* will be invoked only after $B_1, B_1 + B_2, \ldots$ time steps, resulting in a *time scale change.* We will denote the new master time scale by $\tilde{t}$ and the elementary time scale by $t$. All analysis is then on the new master time scale $\tilde{t}$. We set $B_{\tilde{t}} = \lfloor \tilde{t}^{1/8} \rfloor$, then the original time scale $t$ is of order $\tilde{t}^{9/8}$.

The basic FPL algorithm assumes that all losses are observed, which is not necessarily so for all problems. In fact, for reactive problems, this assumption would be too strong, even if we observe the (hypothetical) performance of all experts: Those experts other than the selected one would have computed different decisions and therefore would have had different performance. In the following, we construct an agent for the *bandit setup*, i.e. in each round $\tilde{t}$ it learns only its own loss, but not that of the alternatives. A common technique in bandit and more general partial observation setups is to introduce the quantity of the *estimated loss* $\hat{L}^e_{1:\tilde{T}}$, which is updated in each round by setting $\hat{\ell}^{\tilde{e}}_{\tilde{t}} = \ell^{\tilde{e}}_{\tilde{t}} / \mathbf{P}(\tilde{e}\text{-observation})$ for the currently observed expert $\tilde{e}$, and $\hat{\ell}^e_{\tilde{t}} = 0$ for all experts not observed. By letting $\mathbf{P}(\tilde{e}\text{-observation})$ be the probability of observing the expert $\tilde{e}$'s loss, it is achieved that $\hat{\ell}_{\tilde{t}}$ and consequently $\hat{L}$ are *unbiased* estimates for the true performances.

Dividing by these probabilities, we have to be careful that they cannot become too small, since the analysis of the algorithm crucially relies on the fact of bounded (estimated) losses. In fact, there are two issues to address: First, the probabilities get arbitrarily small for experts with small prior weights. We therefore do not use all experts from the beginning, but for each expert $e \in \mathcal{E}$, we define an *introduction time* $\tau^e \geq 1$. At time $\tilde{t}$, we only work with the set of *active experts* $\{\tau \geq \tilde{t}\} := \{e : \tau^e \geq \tilde{t}\}$. Second, an expert could be assigned with small probability because of past bad performance. For the case that it is selected anyway, we replace the denominator by $\max\{\mathbf{P}(\tilde{e}\text{-observation}), \gamma_{\tilde{t}}\}$, where $\gamma_{\tilde{t}} > 0$ decreases in $\tilde{t}$.

Finally, unlike weighted averaging algorithms (on which there is a rich literature, but we know of no variant with dynamic learning rate and countably many experts so far), the FPL algorithm does not have access to the explicity sampling probabilities $\mathbf{P}(\tilde{e}\text{-observation})$. We can get around this problem by estimating this quantity for the selected expert by sampling the leader for other $\lceil 2\tilde{t}^2 \log(2\sqrt{\tilde{t}}) \rceil = O(t^2)$ times and taking the relative frequency estimate.

**Definition 12.** (Universal FPL algorithm) *The uFPL algorithm is defined as the FPL algorithm based on the universal expert class $\mathcal{P}$ and with the modifications stated in the last paragraphs (new master time scale, loss estimates by dividing by the probability estimates, always working with a finite expert class increasing over time).*

**Theorem 13.** (Performance of uFPL, adapted from [14, 15]) *The uFPL algorithm with learning rate $\eta_{\tilde{t}} = \frac{1}{2}\tilde{t}^{-1/2}$, time step parameter $B_{\tilde{t}} = \lfloor \tilde{t}^{1/8} \rfloor$, denominator threshold $\gamma_{\tilde{t}} = \frac{1}{2}\tilde{t}^{-3/4}$, and introduction times $\tau^p = \lceil w_p^{-8} \rceil$, has a regret bound of at most $\mathbf{E}L^{\mathrm{uFPL}}_{1:T} - \mathbf{E}L^p_{1:T} = O(w_p^{-12} + t^{2/3} \log w_p^{-1})$. This holds against* any *expert $p \in \mathcal{P}$, i.e. against any program on the fixed universal Turing machine, in expectation over FPL's randomization, on* any *sequence of loss vectors. Consequently, measured in terms of average per-round loss, uFPL performs (asymptotically) as well as any program.*

We would like to point the reader to an interesting detail: Most bandit variants for FPL have been defined such that the observations are used only in designated exploration phases. Thus, the resulting algorithm is label efficient and has a lower regret bound of $t^{2/3}$ [17], while our algorithm has a basic bound of $\sqrt{t}$. In fact a bound of $t^{\frac{1}{2}+\varepsilon}$ can be obtained at the cost of increasing power of $w_e^{-1}$ for decreasing $\varepsilon$. Working without explicit exploration is particularly facilitated by the fact that we work with losses instead of rewards: Then, experts which are not sampled can only gain weight, which results in implicit exploration [15].

Of course, each of the modifications of the paragraphs preceeding Definition 12 may cause its own additional regret which has to be taken into account in the analysis. This is carried out in [14, 15]. The conclusion of Theorem 13, "uFPL performs as well as any program, on any task", seems very strong. Moreover, uFPL is even computable, since it works with a finite expert class in each time step. With the trick stated in Definition 10, each expert can also be made computable, the reference is then the class fo resource bounded programs. Still, uFPL is far from being practically relevant, since $w_e^{-1}$ is huge for all but very small expert classes (in particular, $w_p^{-1}$ is huge for any moderate program). On the other hand, one can show that this bound is sharp for bandit problems [4], for which our setup is an instance.

This exponential blow-up, where bounds contain a huge $w_e^{-1}$ instead of a reasonable $\log w_e^{-1}$, occurs in different cases for different reasons. In MDL (Theorem 7), it can be viewed as the possible unbalancedness of the prior [9]. Even in stochastic model selection (Theorem 8), there are (bad) instances where $\Pi$ becomes exponentially large [12]. But most seriously, in (re-)active problem setups this phenomenon seems unavoidable in general. It is one of the major open problems to resolve it, i.e. define conditions and algorithms for which bounds of reasonable size hold also for reactive problems.

## References

1. Cesa-Bianchi, N., Freund, Y., Haussler, D., Helmbold, D., Schapire, R., Warmuth, M.K.: How to use expert advice. Journal of the ACM **44** (1997) 427–485
2. Solomonoff, R.J.: Complexity-based induction systems: comparisons and convergence theorems. IEEE Trans. Inform. Theory **24** (1978) 422–432
3. Gittins, J.C., Jones, D.M.: A dynamic allocation index for the sequential design of experiments. Progress in Statistics (1974) 241–266
4. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: The nonstochastic multiarmed bandit problem. SIAM Journal on Computing **32** (2002) 48–77
5. Hannan, J.: Approximation to Bayes risk in repeated plays. In Dresher, M., Tucker, A.W., Wolfe, P., eds.: Contributions to the Theory of Games 3. Princeton University Press (1957) 97–139
6. Kalai, A., Vempala, S.: Efficient algorithms for online decision. In: Proc. 16th Annual Conference on Learning Theory (COLT-2003). Lecture Notes in Artificial Intelligence, Berlin, Springer (2003) 506–521
7. Hutter, M.: Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability. Springer, Berlin (2004)

8. Li, M., Vitányi, P.M.B.: An introduction to Kolmogorov complexity and its applications. 2nd edn. Springer (1997)
9. Poland, J., Hutter, M.: Asymptotics of discrete MDL for online prediction. IEEE Transactions on Information Theory **51** (2005) 3780–3795
10. Rissanen, J.J.: Fisher Information and Stochastic Complexity. IEEE Trans. Inform. Theory **42** (1996) 40–47
11. Poland, J., Hutter, M.: On the convergence speed of MDL predictions for Bernoulli sequences. In: International Conference on Algorithmic Learning Theory (ALT). (2004) 294–308
12. Poland, J.: Potential functions for stochastic model selection. Technical Report TCS-TR-A-06-11, Hokkaido University (2006) Available online.
13. Hutter, M., Poland, J.: Adaptive online prediction by following the perturbed leader. Journal of Machine Learning Research **6** (2005) 639–660
14. Poland, J., Hutter, M.: Defensive universal learning with experts. In: 17h International Conference on Algorithmic Learning Theory (ALT). (2005) 356–370
15. Poland, J.: FPL analysis for adaptive bandits. In: 3rd Symposium on Stochastic Algorithms, Foundations and Applications (SAGA). (2005) 58–69
16. de Farias, D.P., Megiddo, N.: How to combine expert (and novice) advice when actions impact the environment? In Thrun, S., Saul, L., Schölkopf, B., eds.: Advances in Neural Information Processing Systems 16. MIT Press, Cambridge, MA (2004)
17. Cesa-Bianchi, N., Lugosi, G., Stoltz, G.: Regret minimization under partial monitoring. Technical report (2004)