On Ensemble Techniques for AIXI Approximation

Joel Veness[1], Peter Sunehag[2], and Marcus Hutter[2]
veness@cs.ualberta.ca,
{peter.sunehag,marcus.hutter}@anu.edu.au

1. University of Alberta, 2. Australian National University

Abstract One of the key challenges in AIXI approximation is model class approximation - i.e. how to meaningfully approximate Solomonoff Induction without requiring an infeasible amount of computation? This paper advocates a bottom-up approach to this problem, by describing a number of principled ensemble techniques for approximate AIXI agents. Each technique works by efficiently combining a set of existing environment models into a single, more powerful model. These techniques have the potential to play an important role in future AIXI approximations.

1 Introduction

In statistical data compression, one modeling approach used by many high performance programs is to use an ensemble method to combine the predictions of multiple statistical models [Mattern, 2012]. Each model is typically tailored towards a particular kind of structure that occurs in popular file types. By specifying a number of specialized models, as well as one or more general-purpose models, excellent compression performance can be obtained across a variety of file types. This approach is taken by the powerful PAQ family [Mahoney, 2005] of data compressors, which currently obtain the best compression performance across many well-known benchmarks.

Within reinforcement learning [Sutton and Barto, 1998], some efforts [Veness et al., 2010, 2011] have recently been made towards approximating AIXI [Hutter, 2005], an optimality notion for general reinforcement learning agents. Impressively, these agents have been shown to be able to learn, *from scratch*, to play TicTacToe, Pacman, Kuhn Poker, and other simple games by trial and error alone – even the rules of each game were not communicated to the agent. The mathematical framework used in these works can be considered a natural generalization of the statistical data compression setting to reinforcement learning. The distinguishing feature of this setting is an extra source of side information – namely, the history of actions chosen by some control algorithm – which is incorporated into a sequential, probabilistic framework. Inspired by the success of ensemble methods within data compression, the goal of this paper is to explore a number of principled techniques for combining one or more probabilistic models within reinforcement learning. We restrict our attention to that of *universal* methods, i.e. methods that provide competitive theoretical guarantees with respect to an interesting class of candidate environments. Our contribution is to survey some general techniques from Bayesian statistics, online learning and online convex programming and show how they can be used to define a variety of principled ensemble techniques for information theoretic agents.

2 Background

We now describe our probabilistic agent setting. A more detailed overview of this framework can be found in the work of Hutter [2005] and Veness et al. [2011].

Notation. A string $x_1 x_2 \dots x_n$ of length n is denoted by $x_{1:n}$. The prefix $x_{1:j}$ of $x_{1:n}$, $j \leq n$, is denoted by $x_{\leq j}$ or $x_{<j+1}$. The notation generalises to blocks of symbols: e.g. $ax_{1:n}$ denotes $a_1 x_1 a_2 x_2 \dots a_n x_n$ and $ax_{<j}$ denotes the string $a_1 x_1 a_2 x_2 \dots a_{j-1} x_{j-1}$. The empty string is denoted by ϵ. The concatenation of two strings s and r is denoted by sr. The finite action, observation, and reward spaces are denoted by $\mathcal{A}, \mathcal{O}$, and $\mathcal{R}$ respectively. Also, $\mathcal{X}$ denotes the joint perception space $\mathcal{O} \times \mathcal{R}$.

The following definition states that the environment takes the form of a probability distribution over possible observation-reward sequences conditioned on actions taken by the agent.

Definition 1 *An environment ρ is a sequence of parametrized probability mass functions $\{\rho_0, \rho_1, \rho_2, \dots\}$, where $\rho_n \colon \mathcal{A}^n \to$ Density $(\mathcal{X}^n)$, that satisfies*

$$\forall a_{1:n} \forall x_{<n} : \; \rho_{n-1}(x_{<n} \,|\, a_{<n}) = \sum_{x_n \in \mathcal{X}} \rho_n(x_{1:n} \,|\, a_{1:n}). \tag{1}$$

In the base case, we have $\rho_0(\epsilon \,|\, \epsilon) = 1$.

Equation (1), called the chronological condition by Hutter [2005], captures the natural constraint that action a_n has no effect on earlier perceptions $x_{<n}$. For convenience, we drop the index n in ρ_n from here onwards. Now, given an environment ρ, we define the predictive probability

$$\rho(x_n \,|\, ax_{<n} a_n) := \rho(x_{1:n} \,|\, a_{1:n}) / \rho(x_{<n} \,|\, a_{<n}) \tag{2}$$

$\forall a_{1:n} \forall x_{1:n}$ such that $\rho(x_{<n} \,|\, a_{<n}) > 0$. It now follows that

$$\rho(x_{1:n} \,|\, a_{1:n}) = \rho(x_1 \,|\, a_1) \rho(x_2 \,|\, ax_1 a_2) \cdots \rho(x_n \,|\, ax_{<n} a_n). \tag{3}$$

Definition 1 is used in two distinct ways. The first is to describe the true environment, which is typically not known by the agent. The second is to describe an agent's *subjective* model of the environment. This model is usually adaptive, and will often only be an approximation to the true environment. To make the distinction clear, we will refer to an agent's *environment model* when talking about the agent's model of the environment. Additionally, we introduce the notion of an ϵ-positive environment model. This is defined as an environment model ρ satisfying $\rho(x_n \,|\, ax_{<n}a_n) \geq \epsilon$ for some real $\epsilon > 0$, for all $n \in \mathbb{N}$, for all $x_{1:n} \in \mathcal{X}^n$ and for all $a_{1:n} \in \mathcal{A}^n$. From here onwards we assume all environment models are ϵ-positive.

Redundancy. We will also introduce a notion of regret, *redundancy*, which we will later use to analyze the performance of our ensemble techniques. This is defined as

$$-\log_2 \mu(x_{1:n} \,|\, a_{1:n}) - \min_{\rho \in \mathcal{M}} -\log_2 \rho(x_{1:n} \,|\, a_{1:n})$$

for an arbitrary environment model μ, with respect to some class $\mathcal{M}$ of environment models. Our typical goal will be to show that the redundancy grows $o(n)$. Informally, such a result implies that the average performance of μ will eventually match that of the best model in $\mathcal{M}$ as n gets large.

3 Ensemble Techniques

This section discusses a number of principled ways to construct an enriched environment model from two or more existing environment models. A competitive analysis is given for each method, which justifies their usage in various situations.

3.1 Weighting / Model Averaging

A straightforward way to construct an adaptive environment model that can perform nearly as well as any single model from a finite set of candidate environment models is to use Bayesian Model Averaging (also known as weighting).

Definition 2 *Given a finite set of environment models $\mathcal{M} := \{\rho_1, \rho_2, \ldots\}$ and a prior weight $w_0^\rho > 0$ for each $\rho \in \mathcal{M}$ such that $\sum_{\rho \in \mathcal{M}} w_0^\rho = 1$, the mixture environment model is $\xi(x_{1:n} \,|\, a_{1:n}) := \sum\limits_{\rho \in \mathcal{M}} w_0^\rho \rho(x_{1:n} \,|\, a_{1:n})$.*

The above can easily be shown (for example, see Proposition 1 in the work of Veness et al. [2011]) to define a valid environment model. Because of this, we can simply use

$$\xi(x_n \,|\, ax_{<n}a_n) = \xi(x_{1:n} \,|\, a_{1:n}) \,/\, \xi(x_{<n} \,|\, a_{<n}) \tag{4}$$

to predict the next observation reward pair. Equation (4) can also be expressed in terms of a convex combination of model predictions, with each model weighted by its posterior probability. Formally,

$$\xi(x_n \,|\, ax_{<n}a_n) = \frac{\sum\limits_{\rho \in \mathcal{M}} w_0^\rho \rho(x_{1:n} \,|\, a_{1:n})}{\sum\limits_{\rho \in \mathcal{M}} w_0^\rho \rho(x_{<n} \,|\, a_{<n})} = \sum_{\rho \in \mathcal{M}} w_{n-1}^\rho \rho(x_n \,|\, ax_{<n}a_n),$$

where the posterior weight w_{n-1}^{ρ} for environment model ρ is given by

$$w_{n-1}^{\rho} := \frac{w_0^{\rho}\rho(x_{<n} \,|\, a_{<n})}{\sum\limits_{\nu\in\mathcal{M}} w_0^{\nu}\nu(x_{<n} \,|\, a_{<n})}. \tag{5}$$

This method is justified whenever there exists a model $\rho^* \in \mathcal{M}$ that predicts well, since

$$-\log_2 \xi(x_{1:n} \,|\, a_{1:n}) = -\log_2 \sum_{\rho\in\mathcal{M}} w_0^{\rho}\rho(x_{1:n} \,|\, a_{1:n}) \leq -\log_2 w_0^{\rho^*} - \log_2 \rho^*(x_{1:n} \,|\, a_{1:n}), \tag{6}$$

which implies that we suffer constant redundancy when using ξ in place of ρ^*.

Algorithm. The weights specified by Equation (5) can be maintained in $O(|\mathcal{M}|)$ time and space by using the identity $\rho(x_{1:n} \,|\, a_{1:n}) = \rho(x_{<n} \,|\, a_{<n})\rho(x_n \,|\, ax_{<n}a)$ to incrementally maintain the probability of the data under each environment model. Note however that in some special cases, more efficient techniques exist with time complexity sublinear in $|\mathcal{M}|$. One example is Context Tree Weighting [Willems et al., 1995], which was used as the basis for our previous AIXI approximations [Veness et al., 2010, 2011].

3.2 Switching / Tracking

While weighting provides an easy way to combine models, as an ensemble method it is somewhat limited in that it only guarantees performance in terms of the best *single* model in $\mathcal{M}$. It is easy to imagine situations where this would be insufficient in practice. Instead, one could consider weighting over *sequences* of models chosen from a fixed base class $\mathcal{M}$. Variants of this fundamental idea have been considered numerous times in the literature, for example by Volf and Willems [1998], Herbster and Warmuth [1998] and Erven et al. [2008]. We now show how these ideas can be cast into our probabilistic agent setting, by describing an adaptation of the FixedShare algorithm [Herbster and Warmuth, 1998]. We also provide a short competitive analysis.

Definition 3 *Given a finite set $\mathcal{M} = \{\rho_1, \ldots, \rho_N\}$, $N > 1$, of environment models and a switching sequence $\alpha = \alpha_2\alpha_3 \ldots \in [0,1]^{\infty}$, for all $n \in \mathbb{N}$, for all $x_{1:n} \in \mathcal{X}^n$, the switching environment model with respect to $\mathcal{M}$ and α is defined as*

$$\tau_{\alpha}(x_{1:n} \,|\, a_{1:n}) := \sum_{i_{1:n}\in\mathcal{I}_n(\mathcal{M})} w_{\alpha}(i_{1:n}) \prod_{k=1}^{n} \rho_{i_k}(x_k \,|\, ax_{<k}a_k) \tag{7}$$

where $\mathcal{I}_n(\mathcal{M}) := \{1, 2, \ldots, N\}^n$ and the prior over model sequences is recursively defined by

$$w_{\alpha}(i_{1:n}) := \begin{cases} 1 & \textit{if} \quad i_{1:n} = \epsilon \\ \frac{1}{N} & \textit{if} \quad n = 1 \\ w_{\alpha}(i_{<n}) \times \Big((1-\alpha_n)\mathbb{I}[i_n = i_{n-1}] + \frac{\alpha_n}{N-1}\mathbb{I}[i_n \neq i_{n-1}]\Big) & \textit{otherwise,} \end{cases} \tag{8}$$

Algorithm 1 SwitchMixture - $\tau_\alpha(x_{1:n} \mid a_{1:n})$

```
Require: A finite model class $\mathcal{M} = \{\rho_1, \ldots, \rho_N\}$ such that $N > 1$
Require: A weight vector $(w_1, \ldots, w_N) \in \mathbb{R}^N$, with $w_i = \frac{1}{N}$ for $1 \le i \le N$
Require: A switching sequence $\alpha_2, \alpha_3, \ldots, \alpha_n$

1: $r \leftarrow 1$
2: for $i = 1$ to $n$ do
3:     $r \leftarrow \sum_{j=1}^{N} w_j \rho_j(x_i \mid ax_{<i}a_i)$
4:     $k \leftarrow (1 - \alpha_{i+1})N - 1$
5:     for $j = 1$ to $N$ do
6:         $w_j \leftarrow \frac{1}{N-1}\left[\alpha_{i+1} r + k w_j \rho_j(x_i \mid ax_{<i}a_i)\right]$
7:     end for
8: end for
9: return $r$
```

Now, using the same argument to bound $-\log_2 \tau_\alpha(x_{1:n} \mid a_{1:n})$ as we did in Equation 6, we see that the inequality

$$-\log_2 \tau_\alpha(x_{1:n} \mid a_{1:n}) \le -\log_2 w_\alpha(i_{1:n}) - \log_2 \rho_{i_{1:n}}(x_{1:n} \mid a_{1:n}) \tag{9}$$

holds for any sequence of models $i_{1:n} \in \mathcal{I}_n(\mathcal{M})$, where $\rho_{i_{1:n}}(x_{1:n} \mid a_{1:n})$ denotes the product $\prod_{k=1}^{n} \rho_{i_k}(x_k \mid ax_{<k}a_k)$ of the sequence of conditional probabilities defined by $i_{1:n}$. Next, we state an upper bound on $-\log_2 w_\alpha(i_{1:n})$ that holds for any sequence of model indices.

Lemma 1 *Given a base model class $\mathcal{M}$ and a decaying switch rate $\alpha_t := \frac{1}{t}$ for $t \in \mathbb{N}$,*

$$-\log_2 w_\alpha(i_{1:n}) \le (m(i_{1:n}) + 1)\left(\log_2 |\mathcal{M}| + \log_2 n\right),$$

for all $i_{1:n} \in \mathcal{I}_n(\mathcal{M})$, where $m(i_{1:n}) := \sum_{k=2}^{n} \mathbb{I}[i_k \neq i_{k-1}]$ denotes the number of switches in $i_{1:n}$.

Proof. See the work of Veness et al. [2012].

Combining Equation 9 with Lemma 1 gives the following bound.

Theorem 1 *Given a base model class $\mathcal{M}$ and switch rate $\alpha_t := \frac{1}{t}$ for $t \in \mathbb{N}$, for all $n \in \mathbb{N}$, for all $i_{1:n} \in \mathcal{I}_n(\mathcal{M})$,*

$$-\log_2 \tau_\alpha(x_{1:n} \mid a_{1:n}) \le (m(i_{1:n}) + 1)\left[\log_2 |\mathcal{M}| + \log_2 n\right] - \log_2 \rho_{i_{1:n}}(x_{1:n} \mid a_{1:n}).$$

Thus if there exists an environment model $\rho_{i_{1:n}}$ with $m(i_{1:n}) \ll n$ that predicts well, then τ_α will also predict well. In the case where the best sequence of models satisfies $m(i_{1:n}) = 0$, Theorem 1 gives an extra cost of $\log_2 n$ bits compared to a uniform weighting. Assuming both bounds are tight, $\log_2 n$ can be thought of as the cost of using switching in situations where weighting would have been sufficient.

Algorithm. A direct computation of Equation 7 is intractable. For example, given a history $ax_{1:n}$ and a model class $\mathcal{M}$, the sum in Equation 7 would require $|\mathcal{M}|^n$ additions. Fortunately, the structured nature of the model sequence weights $w_\alpha(i_{1:n})$ can be exploited to derive Algorithm 1. The same argument used to derive the correctness of this procedure for the sequence prediction setting [Veness et al., 2012] can be easily generalised to our agent setting. Assuming that every conditional probability can be computed in constant time, Algorithm 1 runs in $\Theta(n|\mathcal{M}|)$ time and uses only $\Theta(|\mathcal{M}|)$ space. Furthermore, only $\Theta(|\mathcal{M}|)$ work is required to process each new symbol.

3.3 Convex Mixing

This next section introduces *convex mixing*, a technique which, unlike weighting or switching, can sometimes be expected to perform better than any single model or sequences thereof from some base class of environment models $\mathcal{M}$. The key insight is to consider arbitrary convex combinations of the individual model predictions at each time step. More formally, given a set of base environment models $\mathcal{M}$, consider the product of an arbitrary sequence of convex combinations of the conditional probabilities determined by each environment model.

Definition 4 *Given a finite set of ϵ-positive environment models $\mathcal{M}$ and a sequence of weights $\lambda := \{\lambda_1, \lambda_2, \dots\}$, where each $\lambda_i := \{\ \lambda_i^\rho\ \}_{\rho\in\mathcal{M}}$ such that $\lambda_i^\rho \in \mathbb{R}$, $\lambda_i^\rho \geq 0$ and $\sum_{\rho\in\mathcal{M}} \lambda_i^\rho = 1$ for $i \in \mathbb{N}$, the convex environment model with respect to λ is defined as*

$$\nu_\lambda(x_{1:n} \,|\, a_{1:n}) := \prod_{i=1}^{n} \sum_{\rho\in\mathcal{M}} \lambda_i^\rho \; \rho(x_i \,|\, ax_{<i}a_i). \tag{10}$$

The above can easily be seen to define a valid chronological measure.

Proposition 1. *A convex environment model is an environment model.*

Proof. As each environment model $\rho \in \mathcal{M}$ is ϵ-positive, every conditional probability $\rho(x_k \,|\, ax_{<k}a_k)$ is well defined. Therefore we just need to check that Equation (1) is satisfied. Now, $\forall a_{1:n} \in \mathcal{A}^n$ and $\forall x_{<n} \in \mathcal{X}^{n-1}$ observe that

$$\begin{aligned}
\sum_{x_n\in\mathcal{X}} \nu_\lambda(x_{1:n} \,|\, a_{1:n}) &= \sum_{x_n\in\mathcal{X}} \prod_{i=1}^{n} \sum_{\rho\in\mathcal{M}} \lambda_i^\rho \; \rho(x_i \,|\, ax_{<i}a_i) \\
&= \nu_\lambda(x_{<n} \,|\, a_{<n}) \sum_{x_n\in\mathcal{X}} \sum_{\rho\in\mathcal{M}} \lambda_n^\rho \; \rho(x_n \,|\, ax_{<n}a_n) \\
&= \nu_\lambda(x_{<n} \,|\, a_{<n}) \sum_{\rho\in\mathcal{M}} \lambda_n^\rho \sum_{x_n\in\mathcal{X}} \rho(x_n \,|\, ax_{<n}a_n) \\
&= \nu_\lambda(x_{<n} \,|\, a_{<n}) \sum_{\rho\in\mathcal{M}} \lambda_n^\rho \\
&= \nu_\lambda(x_{<n} \,|\, a_{<n}),
\end{aligned}$$

Algorithm 2 ConvexMixture - $\nu_\lambda(x_{1:n}|a_{1:n})$

```
Require: A history $ax_{1:n} \in (\mathcal{A} \times \mathcal{X})^n$, $n \in \mathbb{N}$
Require: An initial weight vector $\boldsymbol{\lambda}_1 \in \Delta^{|\mathcal{M}|-1}$
Require: A sequence $\eta_1, \eta_2, \ldots, \eta_n$, of positive, real-valued step sizes
1: $r \leftarrow 1$
2: for $i = 1$ to $n$ do
3:     $r \leftarrow r \times \sum_{\rho \in \mathcal{M}} \lambda_i^\rho \; \rho(x_i \,|\, ax_{<i}a_i)$
4:     $\boldsymbol{\lambda}_{i+1} = \textsc{SimplexProject}(\boldsymbol{\lambda}_i - \eta_i \nabla \ell_i(\boldsymbol{\lambda}_i\,;\, x_i))$
5: end for
6: return $r$
```

which is what we need. The first three steps follow from Equation (10) and standard calculations, the fourth step follows from Equation (2), and the final step follows since $\sum_{\rho \in \mathcal{M}} \lambda_n^\rho = 1$ *by definition.*

Adaptive Convex Mixing. We will now show how to apply the framework of online convex programming [Zinkevich, 2003, Hazan, 2006] to dynamically (i.e. as a function of $ax_{1:n}$) produce a sequence of weights $\hat{\lambda}$ whose redundancy

$$-\log_2 \nu_{\hat{\lambda}}(x_{1:n} \,|\, a_{1:n}) - \min_{\lambda_* \in \Delta^{|\mathcal{M}|-1}} \left\{ -\log_2 \prod_{i=1}^{n} \sum_{\rho \in \mathcal{M}} \lambda_*^\rho \rho(x_i \,|\, ax_{<i}a_i) \right\} \tag{11}$$

grows $O(\sqrt{n})$ with respect to the best set of *constant* weights in $\Delta^{|\mathcal{M}|-1}$, for all $n \in \mathbb{N}$ and for all $x_{1:n} \in \mathcal{X}^n$, where Δ^k denotes the standard k-simplex. This can be considered as an alternative to weighting over the probability simplex, which will invariably require more restrictive assumptions on the environment model in order to gain computational tractability.

To begin with, we require a sequence of history dependent convex loss functions. These can be obtained by noticing that

$$-\log_2 \nu_\lambda(x_{1:n} \,|\, a_{1:n}) = \sum_{i=1}^{n} -\log_2 \sum_{\rho \in \mathcal{M}} \lambda_i^\rho \; \rho(x_i \,|\, ax_{<i}a_i),$$

which lets us naturally define the loss function at time $n \in \mathbb{N}$ to be

$$\ell_n(\lambda_n\,; ax_{1:n}) := -\log_2 \sum_{\rho \in \mathcal{M}} \lambda_n^\rho \; \rho(x_n \,|\, ax_{<n}a_n).$$

The next proposition shows us that this class of loss functions is convex.

Proposition 2. *$\forall n \in \mathbb{N}$, $\forall ax_{1:n} \in (\mathcal{A} \times \mathcal{X})^n$, $\ell_n(\cdot\,; ax_{1:n})$ is convex.*

Proof. Denote $g_n(\lambda) := \sum_{\rho \in \mathcal{M}} \lambda_n^\rho \; \rho(x_n \,|\, ax_{<n}a_n)$ *and* $h(x) := -\log_2(x)$. *First observe that as a linear function,* g_n *is concave. Also, note that the extended-value extension of* h*, defined by*

$$\tilde{h}(x) = \begin{cases} -\log_2 x & \textit{if } x \in (0, \infty], \\ \infty & \textit{otherwise} \end{cases}$$

Algorithm 3 SimplexProject($\boldsymbol{w}$)

```
Require: A vector $\boldsymbol{w} = (w_1, \ldots, w_d) \in \mathbb{R}^d$ for $d \geq 2$
 1: $i = 1$, $s = -1$
 2: $\boldsymbol{y} \leftarrow$ SortDescending($w_1, \ldots, w_d$)
 3: loop
 4:     $s = s + y_i$
 5:     $r = s/i$
 6:     if $i = d$ or $r \geq y_{i-1}$ then
 7:         $t \leftarrow r$
 8:         break loop
 9:     end if
10:     $i \leftarrow i + 1$
11: end loop
12: for $i = 1$ to $d$ do
13:     $w_i \leftarrow \max(0, w_i - t)$
14: end for
15: return $\boldsymbol{w}$
```

is non-increasing on $\mathbb{R}$. Therefore, since h is convex, it follows (see Section 3.2.4 of [Boyd and Vandenberghe, 2004]) that for all $n \in \mathbb{N}$, $\ell_n(\cdot, ax_{1:n})$ is convex.

The gradient $\nabla \ell_n(\lambda_n\,; ax_{1:n})$ of the loss with respect to λ_n can now be determined by repeatedly using the identity

$$\frac{\partial \ell_n}{\partial \lambda_n^\rho} = \frac{-\rho(x_n \mid ax_{<n}a_n)}{\ln 2 \sum_{\nu \in \mathcal{M}} \lambda_n^\nu \, \nu(x_n \mid ax_{<n}a_n)},$$

for all $\rho \in \mathcal{M}$, to construct the relevant $|\mathcal{M}|$-dimensional column vector. Note that due to the ϵ-positive assumption, we can bound each coefficient in the gradient by

$$\left| \frac{-\rho(x_n \mid ax_{<n}a_n)}{\ln 2 \sum_{\nu \in \mathcal{M}} \lambda_n^\nu \, \nu(x_n \mid ax_{<n}a_n)} \right| \leq \frac{1}{\ln 2 \sum_{\nu \in \mathcal{M}} \lambda_n^\nu \epsilon} = \frac{1}{\epsilon \ln 2},$$

which implies that

$$\|\nabla \ell_n(\lambda_n\,; ax_{1:n})\|_2 \leq \sqrt{|\mathcal{M}|}\frac{1}{\epsilon \ln 2}. \tag{12}$$

Theoretical Analysis. Since we have cast our problem into the framework of online convex programming, the argument of Zinkevich [2003] can be used to state a redundancy bound for convex environment models. This analysis assumes the existence of a known upper bound $G := \sup_{1 \leq i \leq n} ||\nabla l_i(\lambda_i; ax_{1:i})||_2$ on the l_2-norm of the gradients as well as on the diameter $D := \max_{c_1, c_2 \in \mathcal{C}} \|c_1 - c_2\|_2$ of the convex set (the simplex for us) that we perform the optimization over. The result, formulated by Hazan [2006] in Theorem 2.1 (page 12), says that by setting $\eta_i = \frac{D}{G\sqrt{i}}$, the cumulative regret after n steps is bounded by $3GD\sqrt{n}$.

Theorem 2 *Using Algorithm 2 with a step size of $\eta_i = \frac{\epsilon \ln 2}{\sqrt{i}}$ for $1 \leq i \leq n$,*

$$\max_{\lambda_* \in \Delta^{|\mathcal{M}|-1}} \left\{ \log_2 \prod_{i=1}^{n} \sum_{\rho \in \mathcal{M}} \lambda_*^{\rho} \rho(x_i \mid ax_{<i}a_i) \right\} - \log_2 \nu_{\hat{\lambda}}(x_{1:n} \mid a_{1:n}) \leq \frac{3|\mathcal{M}|\sqrt{n}}{\epsilon \ln 2}$$

Proof. The result follows by using Proposition 2, the fact that the diameter D of $\Delta^{|\mathcal{M}|-1}$ is $\sqrt{|\mathcal{M}|}$, the bound (12) *that gives us G and the theorem by Zinkevich [2003] as formulated by Hazan [2006] in Theorem 2.1.*

Algorithm. Algorithm 2 shows how to efficiently compute a convex mixture environment. It uses the notation $\boldsymbol{\lambda}_i$ to compactly denote the vector $(\lambda_i^{\rho_1}, \ldots \lambda_i^{\rho_{|\mathcal{M}|}})$ formed from the weights of each environment model in $\mathcal{M}$ at time i. The subroutine SimplexProject projects an arbitrary vector in $\mathbb{R}^{|\mathcal{M}|}$ onto the closest (in terms of Euclidean Distance) point inside the probability simplex $\Delta^{|\mathcal{M}|-1}$. The pseudocode for this routine, derived from the technique presented by Chen and Ye [2011], is given in Algorithm 3; it runs in $O(|\mathcal{M}| \log |\mathcal{M}|)$ time. The SortDescending subroutine in Algorithm 3 returns a vector $\boldsymbol{y}$ whose components $y_1, \ldots, y_d$ are a permutation of the components of the input vector satisfying $y_1 \geq y_2 \cdots \geq y_d$. The overall complexity of the algorithm (not including the cost of running the models in $\mathcal{M}$) is $O(n|\mathcal{M}| \log |\mathcal{M}|)$, and can be computed incrementally using $O(|\mathcal{M}| \log |\mathcal{M}|)$ time to process each percept.

3.4 A Second Order Method.

Additionally, we can exploit a stronger property of our class of loss functions to describe a more computationally demanding algorithm with better redundancy behaviour. To do this, we begin by showing that our class of loss functions is α-exp-concave. Recall that a function f is said to be α-exp-concave for a real $\alpha > 0$ if the function $\exp\{-\alpha f(\cdot)\}$ is concave.

Proposition 3. *$\forall n \in \mathbb{N}$, $\forall ax_{1:n} \in (\mathcal{A} \times \mathcal{X})^n$, $\ell_n(\cdot\,; ax_{1:n})$ is 1-exp-concave.*

Proof. $\forall n \in \mathbb{N}$, $\forall ax_{1:n} \in (\mathcal{A} \times \mathcal{X})^n$, observe that

$$\exp\{-\alpha\, \ell_n(\lambda_n\,; ax_{1:n})\} = \left(\sum_{\rho \in \mathcal{M}} \lambda_n^{\rho}\, \rho(x_n \mid ax_{<n}a_n) \right)^{\alpha}.$$

Thus when $\alpha = 1$, $\exp\{-\alpha\, \ell_n(\cdot\,; ax_{1:n})\}$ is a convex combination of conditional probabilities, which is a concave function. Hence $\ell_n(\cdot\,; ax_{1:n})$ is 1-exp-concave.

This property, along with our previous upper bound on the Euclidean norm of the gradient of the loss, permits us to use the second order OnlineNewtonStep method of Hazan et al. [2006] in place of Algorithm 2. The resultant method would enjoy a guaranteed redundancy of $O(\log n)$, at the cost of a more complicated implementation whose space complexity is $O(|\mathcal{M}|^2)$, and whose *per* *time-step* complexity is $O(|\mathcal{M}|^2)$ plus the cost of solving a convex quadratic program to compute a generalized projection onto the probability simplex. We defer a more thorough empirical comparison between these two approaches to future work.

4 Conclusion

This paper has described a number of principled ensemble techniques for universal reinforcement learning agents. Each technique works by efficiently combining a set of existing environment models into a single, more powerful model. We expect these techniques to play an important role in future AIXI approximations. For example, the MC-AIXI agent could be extended by using these techniques to combine multiple instantiations of FAC-CTW, with each instantiation using a different notion of context as per Section 9.3 of the work of Veness et al. [2011].

Acknowledgments. This work was supported by ARC grant DP0988049.

5 Bibliography

S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

Y. Chen and X. Ye. Projection Onto A Simplex. *ArXiv e-prints 1101.6081*, January 2011.

Tim Van Erven, Peter Grünwald, and Steven De Rooij. Catching Up Faster in Bayesian Model Selection and Model Averaging. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 417–424. MIT Press, Cambridge, MA, 2008.

Elad Hazan. *Efficient algorithms for online convex optimization and their applications.* PhD thesis, Princeton, NJ, USA, 2006.

Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. Logarithmic regret algorithms for online convex optimization. In *In 19th COLT*, pages 499–513, 2006.

Mark Herbster and Manfred K. Warmuth. Tracking the best expert. *Machine Learning*, 32: 151–178, August 1998.

Marcus Hutter. *Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability.* Springer, 2005.

M. Mahoney. Adaptive weighing of context models for lossless data compression. Technical report, Florida Institute of Technology, 2005.

Christopher Mattern. Mixing strategies in data compression. In *Data Compression Conference (DCC)*, pages 337–346, 2012.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

J. Veness, K. S. Ng, M. Hutter, and D. Silver. Reinforcement learning via AIXI approximation. In *Proc. 24th AAAI Conference on Artificial Intelligence*, pages 605–611, Atlanta, 2010. AAAI Press.

Joel Veness, Kee Siong Ng, Marcus Hutter, William Uther, and David Silver. A Monte Carlo AIXI approximation. *Journal of Artificial Intelligence Research*, 40:95–142, 2011.

Joel Veness, Kee Siong Ng, Marcus Hutter, and Michael H. Bowling. Context Tree Switching. In *Data Compression Conference (DCC)*, pages 327–336, 2012.

Paul A. J. Volf and Frans M. J. Willems. Switching between two universal source coding algorithms. In *In Data Compression Conference*, pages 491–500, 1998.

Frans M.J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. The Context Tree Weighting Method: Basic Properties. *IEEE Transactions on Information Theory*, 41:653–664, 1995.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, pages 928–936, 2003.