

# Upper Confidence Trees and Billiards for Optimal Active Learning

P. Rolet and M. Sebag and O. Teytaud

TAO, Inria, Lri, UMR CNRS 8623, Bat. 490 Univ. Paris-Sud F-91405

**Abstract** : This paper focuses on Active Learning (AL) with bounded computational resources. AL is formalized as a finite horizon Reinforcement Learning problem, and tackled as a single-player game. An approximate optimal AL strategy based on tree-structured multi-armed bandit algorithms and billiard-based sampling is presented together with a proof of principle of the approach. **Mots-clés** : Apprentissage actif, Fouille d'arbre Monte-Carlo, Bandits

## 1 Introduction

Active Learning, a most active topic in the Machine Learning (ML) field (Kulkarni *et al.*, 1993; Cohn *et al.*, 1994; Schohn & Cohn, 2000; Dasgupta, 2005; Castro *et al.*, 2006; Hoi *et al.*, 2006; Hanneke, 2007), aims at finding accurate hypotheses with a significantly lesser number of labelled examples than the standard ML setting, through a judicious selection of the instances to be labelled by the expert. Prominent approaches in the active learning field (more in section 2) rely on the properties of the hypothesis space (VC dimension or covering numbers) and/or propose various criteria estimating the additional information provided by an example; they mostly proceed by iteratively selecting the optimal example in the sense of the above criterion.

This paper presents a new perspective on Active Learning (AL), formalized as a finite horizon Reinforcement Learning problem (Sutton & Barto, 1998). The selection of a new sample to be labelled by the expert is viewed as an action, and the final reward associated to a sequence of actions is the generalization error of the hypothesis learned from the so completed training set. Under mild assumptions (Bayesian realizable setting) detailed in section 3, this paper presents a provably optimal AL strategy, indexed by the considered finite horizon. As could have been expected, the formal derivation of an optimal active learning strategy is intractable. A tractable approximation thereof is thus presented (section 4) and constitutes the contribution of the paper<sup>1</sup>. This tractable approximation involves two main ingredients. Firstly, the tree-structured multi-armed bandit algorithm first presented by (Kocsis & Szepesvari, 2006; Coulom, 2006) and extensively investigated in the domain of games (Gelly & Silver, 2007), is adapted to active learning, viewed as a single-player game. Secondly, an unbiased and frugal sampler of the hypothesis and instance spaces, based on billiard-based mechanisms (Rujan

<sup>1</sup>The theoretical part of the paper is under review at the time of writing.

& Marchand, 2000; Herbrich *et al.*, 2001) is used. A proof of principle of the *BAAL* (*Bandit-based Active Learner*) algorithm is presented, with convincing empirical evidence (section 6).

The paper is organized as follows. Section 2 briefly introduces the notations used throughout the paper and reviews the state of the art. Section 3 formally describes AL as a Reinforcement Learning problem, and how to tackle it as a single-player game. The *BAAL* algorithm is described in section 4; some key issues are discussed in section 5 and the goal of experiments is presented. A proof of principle of the approach is given in section 6 and the paper concludes with some perspectives for further research.

## 2 Background and State of the art

Notations and definitions used in the paper are the following. Let  $s_T = \{(x_t, y_t), x_t \in X, y_t \in Y, t = 1 \dots T\}$  denote a  $T$ -size training set, with  $X$  the instance space and  $Y$  the label space (unless otherwise specified, only the classification case will be considered in the paper). From  $s_T$ , a learner  $\mathcal{A}$  extracts some hypothesis  $h$  in hypothesis space  $\mathcal{H}$ , mapping  $X$  onto  $Y$ . The learning performance most usually refers to the expectation of the loss  $\ell(h(x), h^*(x))$  incurred by  $h$  over the target hypothesis  $h^*$ , where the expectation is taken over the joint distribution  $P_{XY}$  on the problem domain. Whereas the standard supervised learning setting assumes that training examples  $(x_t, y_t)$  are iid after  $P_{XY}$ , active learning selects in each time step  $t$  a sample  $x_t$  in the instance space  $X$  (or in the sample pool), the label  $y_t$  of which is determined by the oracle. A sampler  $S$  is a mapping from  $(X \times Y)^N$  to  $X$ , also referred to as *policy* or *strategy*, selecting a new sample  $x$  to be labelled after the current training set  $s_T$ . A learner  $\mathcal{A}$  is a mapping from  $(X \times Y)^N$  to  $\mathcal{H}$  associating a hypothesis  $h$  to any training set  $s_T$ . The *Version Space* (VS) associated to a training set  $s_T$ , noted  $\mathcal{H}(s_T)$ , is the set of hypothesis consistent with  $s_T$ , i.e. such that  $h(x_t) = y_t$  for  $1 \leq t \leq T$ . A first AL research direction focuses on the *uncertainty region* (set of samples where VS hypotheses disagree). (Cohn *et al.*, 1994) reduce the VS through selecting new samples in the uncertainty region. Query-by-Committee algorithms (Seung *et al.*, 1992; Freund *et al.*, 1997) also aim to reducing the VS; their quasi-optimality in the realizable classification case (i.e. when the target hypothesis  $h^*$  belongs to  $\mathcal{H}$ ) has been established by (Dasgupta, 2005). More generally, when there exists a probability measure  $P_H$  on  $\mathcal{H}$ , AL aims to either reducing the measure of the version space, or the variance of the VS hypothesis labels<sup>2</sup>.

Another AL research direction focuses on *error reduction*, meant as the expected generalization error improvement brought by an instance (Cohn *et al.*, 1996; Freund *et al.*, 1997; Iyengar *et al.*, 2000; Roy & McCallum, 2001; Lindenbaum *et al.*, 2004); many criteria, reflecting various measures of the expected error reduction, have been proposed and AL proceeds by greedily selecting the optimal samples in the sense of the considered criterion.

Other approaches exploit prior, learner-dependent knowledge about what makes a sample informative, e.g. its margin (Schohn & Cohn, 2000). (Hoi *et al.*, 2006) considers *batch* active learning, querying a subset of samples that results in the largest

<sup>2</sup>In case such a probability measure  $P_H$  does not exist, then the reduction of the version space can be expressed in terms of its diameter, that is, the measure of points where VS hypotheses differ.

reduction of the Fisher information. These approaches however happen to face learning instabilities, which might require to mix the active learning selection with a uniform selection (Xiao *et al.*, 2005). Such instabilities suggest that an efficient AL system can hardly be obtained by iteratively selecting the most informative samples, at least using the criteria considered so far.

On the theoretical side, significant results have been obtained in terms of lower and upper bounds on the reduction of the sample complexity brought by AL, e.g. depending on the complexity of the hypothesis search space measured through covering numbers or Kolmogorov complexity (Kulkarni *et al.*, 1993; Vidyasagar, 1997; Castro *et al.*, 2006); the suitedness of hypothesis spaces to active learning has been studied (Dasgupta, 2006; Hanneke, 2007) and an “almost” optimal (though intractable) algorithm has been proposed by (Dasgupta, 2006) in the realizable setting for finite VC-dimension.

### 3 An optimal Active Learning strategy

The rest of the paper relies on three assumptions: i) *Bayesian setting* (the existence of a probability measure  $P_H$  on the hypothesis space  $\mathcal{H}$ ); ii) *realizable setting* (the target concept  $h^*$  is deterministic and belongs to  $\mathcal{H}$ ); iii) *short time horizon* (the total number  $T$  of samples to be labelled along the AL process is in the order of tens or hundreds). Under these assumptions, this section formalizes AL as a Partially Observable Markov decision process (POMDP). The search for an optimal AL strategy is viewed as a *Reinforcement learning* problem, and tackled as a one-player game.

#### 3.1 AL as a Markov Decision Process

Markov decision processes are basically described in terms of states, actions, reward, policy and transition functions (Sutton & Barto, 1998). As a first step, the state space  $\mathcal{S}$  of AL is viewed as the set of possible training sets  $s_t$  (this statement will be refined in section 3.2 since the *unobserved* target hypothesis is also part of the state). An action corresponds to the selection of a new sample to be labelled; let  $A$  denote the set of actions.

The reward function associated to state  $s_t$  corresponds to the generalization error of the hypothesis  $\mathcal{A}(s_t)$  learned from  $s_t$ ; accordingly, the reward function is unknown except for horizon states ( $t = T$ ).

Any sampler  $S$ , mapping a state  $s_t$  onto an action (a new sample  $x_{t+1}$ ), is a policy. Lastly, the transition function  $p : \mathcal{S} \times A \times \mathcal{S} \rightarrow \mathbb{R}_+$  defines the probability of arriving at some state  $s_{t+1}$  by selecting action  $x$  in state  $s_t$  (see below).

Considering horizon  $T$ , let  $S_T(h)$  denote the training set built by applying  $T$  times policy  $S$  when learning the target concept  $h$ , and denote  $\mathbf{Err}(\mathcal{A}(S_T(h)), h)$  the generalization error of the hypothesis learned from  $S_T(h)$ . It comes naturally that an optimal AL strategy is one minimizing the expectation of  $\mathbf{Err}(\mathcal{A}(S_T(h)), h)$  when  $h$  ranges in  $\mathcal{H}$ :

$$S_T^* = \arg \min_S \mathbb{E}_{h \sim \mathcal{H}} \mathbf{Err}(\mathcal{A}(S_T(h)), h). \quad (1)$$

Likewise, let  $s_T$  be a training set built after  $T$  decisions of the  $S$  policy; after the realizable assumption, the target hypothesis  $h$  is only bound to be in the version space of  $s_T$ ,

noted  $\mathcal{H}(s_T)$ . The reward function (value at horizon  $T$ )  $V(s_T)$  thus is the expectation of the generalisation error, taken over  $\mathcal{H}(s_T)$ :

$$V(s_T) = \mathbb{E}_{h \sim \mathcal{H}(s_T)} \mathbf{Err}(\mathcal{A}(s_T), h). \quad (2)$$

It remains to characterize transition function  $p(s_t, x, s_{t+1})$ . By construction,  $p$  is zero for all  $s_{t+1}$  except for  $s_{t+1} = (s_t, (x_{t+1} = x, y_{t+1}))$ . In the latter case,  $p$  reflects the probability for the label of  $x$  to be  $y_{t+1}$ . After the same argument as above,  $p(s_t, x, s_{t+1})$  is thus expressed as a function of the probability of the label of  $x$ , conditionally to the fact that the target hypothesis varies in  $\mathcal{H}(s_t)$ :

$$\begin{aligned} p(s_t, x, (s_t, (x_{t+1} = x, y_{t+1}))) = \\ p(h(x) = y_{t+1} | h \in \mathcal{H}(s_t)) \end{aligned} \quad (3)$$

It is shown that the above value (Eq. (2)) and transition (Eq. (3)) functions yield an optimal active learning sampling strategy in the sense of the minimization of the expected generalization error (Eq. (1)).

**Theorem 1 (Optimal active learning policy)**

Let  $\mathbb{E}$  denote the expectation operator defined after Eq. 3. Let value function  $V^*$  be recursively defined as follows, where  $|s|$  denotes the number of examples in training set  $s$ :

$$V^*(s) = \begin{cases} \mathbb{E}_{h \sim \mathcal{H}(s)} \mathbf{Err}(\mathcal{A}(s), h) & \text{if } |s| = T \\ \inf_{x \in X} \mathbb{E}_{s' \sim p(s_t, x, s')} V^*(s') & \text{otherwise} \end{cases} \quad (4)$$

Let the associated Bellman optimal strategy  $S^{*,T}$  be defined as:

$$S^{*,T}(s) = \arg \inf_{x \in X} \mathbb{E}_{s' \sim p(s_t, x, s')} V^*(s') \quad (5)$$

Then  $S^{*,T}$  is optimal in the sense of Eq. (1).

**3.2 A Partially Observable MDP**

As mentioned earlier on, AL actually is a Partially Observable Markov Decision Process (POMDP): a state involves both the (observed) current training set, and the (unknown) target hypothesis. After (Astrom, 1965) a POMDP can however be solved by MDP techniques, provided that the state set include all possible events (here, the possible sequences of samples and associated labels). In the AL context the size of the state set is however overwhelmingly large. An alternative to “embedding” the hidden variable in the state set is to model the hidden variable within the transition function of the MDP (Eq. (3)) and the reward function (Eq. (2)) (Astrom, 1965).

This alternative leads to formalizing AL as a one player game, as follows. In each game, the AL strategy plays against an (unknown) hypothesis  $h$ , sampled in the version space  $\mathcal{H}(s_*)$  of the initial training set  $s_*$  after the realizable setting assumption<sup>3</sup>. Upon

---

<sup>3</sup>For the sake of simplicity and when no confusion is to fear, the initial training set  $s_*$  is omitted and  $\mathcal{H}$  is used instead of  $\mathcal{H}(s_*)$ .

each move (selection of a sample  $x$ ), hypothesis  $h$  is used as an oracle to label  $y = h(x)$ . At the end of the game (after  $T$  samples have been selected, defining training set  $s_T$ ), the reward can actually be estimated as the generalization error of  $\mathcal{A}(s_T)$  (the hypothesis learnt from  $s_T$ ) against  $h$ . The empirical reward thus is computed as a uniform draw of the random variable  $\text{Err}(\mathcal{A}(s), h)$ , where  $h$  ranges in  $\mathcal{H}$ . The average empirical reward collected by the AL player after many games thus asymptotically converges toward the true expectation of this random variable, that is, the desired reward function (Eq. (2)).

## 4 Tractable Approximations of Optimal AL strategy

This section presents a tractable approximation of the optimal active learning strategy, relying on two components. Firstly, the UCT algorithm proposed for tree-structured multi-armed bandits (Kocsis & Szepesvari, 2006) is extended to the one-player game of AL. Secondly, a fair and frugal sampling algorithm, based on billiard approaches (Herbrich *et al.*, 2001), is proposed.

### 4.1 Bandit-based Active Learning

The UCT (Upper Confidence Tree) algorithm can be viewed as a Monte-Carlo tree-search algorithm (Coulom, 2006), with the specificity of using the well-founded multi-armed bandit framework (Auer, 2003) to select an arm (child node) at each node, thus offering good guarantees about sequential optimal decision under uncertainty. Notably, UCT became famous in the domain of strategic games as it inspired the computer-Go program MoGo, first to ever win over professional human players (Gelly & Silver, 2007).

UCT-based game strategy provides the basis for the proposed *BAAL* algorithm. The actual tree  $\mathcal{T}$  is initialized to the root node (the current game position, here the initial training set  $s_*$ ).  $\mathcal{T}$  is constructed iteratively through  $N$  tree-walks, or simulations, where  $N$  is a user-supplied parameter controlling the computational resources. Each tree-walk starts by drawing the pretended target hypothesis  $h$  in  $\mathcal{H}(s_*)$  (section 3.2), which will be used as oracle during this tree-walk. At each node  $s$ , either its child nodes belong to  $\mathcal{T}$ ; or  $s$  is a leaf node of  $\mathcal{T}$ . In the former case, some child node (sample  $x$ ) is selected after Eq. (6), its label is set to  $h(x)$ , and the next node is  $s \cup (x, h(x))$ . In the latter case, a random path (selection of samples) is followed until reaching the maximum path length  $T$ , thus defining a training set  $s_T$ . At this point the empirical reward (generalization error of  $\mathcal{A}(s_T)$  w.r.t  $h$ ) is computed, and used to update the reward estimates of the path nodes belonging to  $\mathcal{T}$ ; further, the first node in the path not belonging to  $\mathcal{T}$  is added to  $\mathcal{T}$ .

The equation used to select the child node, referred to as Upper Confidence Bound (UCB) (Auer, 2003), selects the  $i$ -th arm which maximizes the sum of the empirical reward  $\hat{\mu}_i$  (exploitation), and an exploration term depending on the number  $n_i$  of times arm  $i$  has been selected; the exploration vs exploitation tradeoff is adjusted using some tuned constant  $C$ :

$$\arg \max_{i \in \text{child nodes}} \hat{\mu}_i + C \sqrt{\frac{\log(\sum_{j \in \text{child nodes}} n_j)}{n_i}} \quad (6)$$

**Algorithm 1** Left: The BAAL algorithm, with input: measure  $P_H$  on hypothesis space  $\mathcal{H}$ ; initial training set  $s_*$ ; time horizon  $T$ ; number  $N$  of allowed tree-walks, returns a sample  $x$  to be labelled by the oracle. Right:

<pre> <b>Tree-Walk</b>(<math>s, t, h</math>) Increment <math>n(s)</math>     // number of times <math>s</math> has been visited <b>if</b> <math>t==0</math> <b>then</b>     Compute <math>r = Err(\mathcal{A}(s), h)</math> <b>else</b>     <math>\mathcal{X}(s) = ArmSet(s, n(s))</math>     Select <math>x^* = UCB(s, \mathcal{X}(s))</math>     // recursively call Tree-Walk     <math>r = Tree-Walk(s \cup \{(x^*, h(x^*))\}, t - 1, h)</math> <b>end if</b>     // update reward of leaf node <math>s</math> <math>r(s) \leftarrow (1 - \frac{1}{n(s)})r(s) + \frac{1}{n(s)} r</math> Return <math>r</math>                 </pre>	<pre> <b>BAAL</b>(<math>P_H, s_*, T, N</math>) <b>for</b> <math>i=1</math> to <math>N</math> <b>do</b>     <math>h = DrawHypothesis(P_H, s_*)</math>     // <math>h \sim P_H, h \in \mathcal{H}(s_*)</math>     <math>Tree-Walk(s_*, T, h)</math> <b>end for</b>     // select the first sample on the most visited branch Return <math>x = \arg \max_{x' \in \mathcal{X}} \{n(s \cup \{x'\})\}</math> <b>UCB</b>(<math>s, \mathcal{X}</math>) <b>for</b> <math>x \in \mathcal{X}</math> <b>do</b>     <b>if</b> <math>n(s \cup \{x\}) == 0</math> <b>then</b>         Return <math>x</math>     <b>end if</b> <b>end for</b> return <math>\arg \max_{x \in \mathcal{X}} r(s \cup \{x\}) + \sqrt{\frac{\log(\sum_{x'} n(s \cup \{x'\}))}{n(s \cup \{x\})}}</math>                 </pre>
--	--

BAAL (Alg. 1) implements UCT with two specific ingredients, the `DrawHypothesis` function used to select the random variable  $h$  for each tree-walk (section 4.3), and an `ArmSet` function needed to handle the huge number of arms (samples) (section 4.2). BAAL takes as input the probability distribution  $P_H$  on hypothesis space  $\mathcal{H}$  (Bayesian setting); the initial training set  $s_*$ ; the time horizon  $T$ ; the number  $N$  of allowed tree-walks. Its output is the sample  $x$  to be labelled by the oracle.

## 4.2 Progressive Widening

After Eq. (6), UCB requires each arm to be selected at least once. In the case where the number of arms is large w.r.t the time horizon, UCB thus tends to degenerate into pure exploration; the tuning of the tradeoff constant  $C$  cannot enforce an effective exploration vs exploitation tradeoff. UCT faces the same limitation, and even more so as many arms need be considered at each node of the tree.

The *progressive widening* heuristics has been proposed by (Coulom, 2006) to handle such cases. Let  $n$  denote the number of times node  $s$  has been visited so far; then the number of arms (branches) that can be considered from  $s$  is empirically limited to  $m = \lfloor n^{1/4} \rfloor$  (Coulom, 2006; Chaslot *et al.*, 2007; Wang *et al.*, 2008). Typically, a single arm will be explored from the root node in the first fifteen tree-walks; an additional arm is considered in the sixteenth random walk; UCB will be used to select among both arms during random walks 16 to 80; a third arm is considered in random walk 81, and so forth.

The rationale behind this heuristics is that the better (i.e. the more visited)  $s$ , the

more careful the investigation of its subtrees should be. In the AL context, progressive widening relaxes the finite sample pool assumption: function `ArmSet` returns the first  $m$  candidate samples, with  $m = \lfloor n^{1/4} \rfloor$ , extracted from the pool or from the instance space in a random (fixed) order. `ArmSet` also provides room for hybridizing *BAAL* with AL criteria such as Maximum Uncertainty, as will be shown in section 5.2.

### 4.3 Billiard-based Hypothesis Sampling

*BAAL* relies on the uniform sampling of the hypotheses in  $\mathcal{H}(s_*)$ . The sampling mechanism we used is a ray-tracing a.k.a. billiard algorithm inspired from (Rujan & Marchand, 2000; Herbrich *et al.*, 2001).

Let us consider a connected domain  $\Omega \subset \mathbb{R}^d$ , defined from a set of constraints  $g_1, \dots, g_n: \Omega = \{x \in \mathbb{R}^d \text{ s.t. } g_i(x) \geq 0, i = 1 \dots n\}$ . The billiard algorithm considers a point  $z \in \mathbb{R}^d$  (not necessarily in  $\Omega$ ) and a direction  $\vec{v}$  ( $\vec{v} \in \mathbb{R}^d, \|\vec{v}\| = 1$ ). The trajectory followed by  $z$  is such that: i) the set of constraints satisfied by  $z$  does not decrease; ii)  $z$  “bounces” when it meets an active constraint  $g$ , i.e. its direction  $v$  is changed into its symmetrical with respect to  $g$  (elastic shock); iii) the trajectory is stopped when the computational resources are exhausted, that is when its total length reaches some user-defined parameter  $L$ , and the final point is returned. Under strong conditions (small  $d$  and “sufficiently regular” constraints), and up to a few exceptions (excluding for instance the case of a rational initial direction  $\vec{v}$ ), a billiard trajectory is ergodic, i.e. it covers the whole domain when  $L$  goes to infinity; the distribution of the final trajectory point converges toward the uniform distribution on  $\Omega$ . Although these

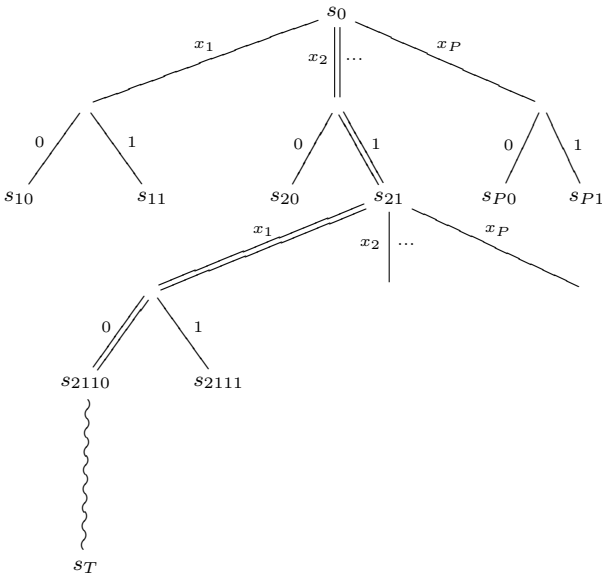


Figure 1: Search Tree developed by *BAAL* (binary classification case). At any state  $s_t$ , a sample  $x$  is selected and labelled using the current random variable  $h$  draw.

good properties are only conjectured in the general case, billiard algorithms have been successfully used in ML, e.g. to estimate the Bayes classifier in a (high-dimensional) kernel feature space (Rujan & Marchand, 2000).

---

**Algorithm 2** Billiard. Input: set of constraints  $g_i$ , length  $L$ . Output: final point.

---

```

Randomly select  $z \in \mathbb{R}^d$  satisfying at least one (but not necessarily all) constraints,
and a direction  $\vec{v}$ .
while  $L > 0$                                      //Find the set of satisfied constraints do
   $J = \{j; g_j(p) \geq 0\}$                              //Go as far as possible while  $g_j \geq 0, j \in J$ 
   $\lambda^* = \sup\{\lambda \geq 0 \text{ s.t. } \forall \ell < \lambda, \forall j \in J, g_j(z + \ell\vec{v}) \geq 0\}$ 
  if  $J = \{1, \dots, n\}$                                (all constraints satisfied) then
    if  $L > \lambda^*$  then
       $p = p + \lambda^* \vec{v}$                              //Go until some  $g_i$  is saturated
       $L = L - \lambda^*$ 
    else Return  $p + L\vec{v}$                              //out of resources
    end if
  end if
   $\vec{v} = \text{symmetric}(\vec{v}, g_i)$                        //Bounce against  $g_i$ 
end while

```

---

In the active learning case, the target domain is the hypothesis version space defined from the initial training set; each training example defines a constraint. In the simple case where the hypothesis space  $\mathcal{H}$  is the set of separating hyperplanes, the bounce operator is defined as follows. Let  $x_i$  be the “saturated constraint”, let  $\vec{v} = \alpha x_i + \beta v'$  (with  $\langle v', x_i \rangle = 0$ ), then  $\text{symmetric}(\vec{v}, x_i) = -\alpha x_i + \beta v'$  (up to normalization).

## 5 Discussion

A key issue for UCT-based approaches concerns the trade-off between the *number* of tree-walks, aka simulations, used to estimate the targeted rewards, and the *precision and cost* of each simulation. In *BAAL*, each simulation involves two types of random variables: the random hypothesis  $h$  used as oracle (`DrawHypothesis`), and the choice of a new candidate arm when the number of visits to the current node reaches a new threshold (`ArmSet`). This section discusses the weaknesses of both functions, proposes some refinements, and finally defines the goal of experiments.

### 5.1 A better reward estimate

Although *BAAL* is based on an unbiased estimate of the reward (section 3.2), this estimate might have a high variance for it is based on a single hypothesis draw. Such a high variance might adversely affect the convergence of *BAAL*. UCB (Eq. 6) might stick to an arm for a while, even though this arm is non optimal, if this arm has gotten a few lucky rewards. In such cases, UCT will investigate in more depth the tree branches corresponding to these lucky arms, after the asymmetric development of the search tree (section 4.1). As these branches are more investigated, “cumulatively misleading” evidence is produced: the associated rewards look excellent while the true optimal branches are missed. All in all, a high reward variance might cause *BAAL* to leave aside

optimal samples for a long while. In order to overcome this weakness, a better reward estimate will be considered: after a  $T$ -size training set  $s_T$  has been built based on the draw of  $h$ ,  $R$  additional hypotheses  $h_1, \dots, h_R$  are uniformly sampled in  $\mathcal{H}(s_T)$ . The estimated reward is computed by averaging  $Err(\mathcal{A}(s_T), h)$  and  $Err(\mathcal{A}(s_T), h_i)$  for  $i = 1 \dots R$ . This *BAAL* variant referred to as *Av(R)-BAAL* involves  $R$  additional calls to the `DrawHypothesis` function, and reduces the reward variance by a factor  $\sqrt{R}$ .

## 5.2 BAAL with Maximum Uncertainty

Likewise, *BAAL* can ignore optimal samples for a while if `ArmSet` considers an unlucky sample order: late introduced samples are at a disadvantage compared to the first ones, more investigated. Two variants of *BAAL* inspired from the uncertainty region (Cohn *et al.*, 1994; Freund *et al.*, 1997) are investigated to overcome this weakness. The uncertainty region, including all samples for which some hypotheses in the version space disagree, shrinks as the training set increases. In such cases, when `ArmSet` considers a randomly ordered sample pool, *BAAL* (referred to as *RND-BAAL* in the following) tends to select samples outside of the uncertainty region, thus gaining little information.

A first variant, referred to as *MU-BAAL*, thus orders the sample pool considered by `ArmSet` after the Maximum Uncertainty criterion, akin (Freund *et al.*, 1997).

A second and less aggressive variant, referred to as *UR-BAAL*, only biases the sample order toward the uncertainty region, with two motivations: computational savings, and better sample diversity compared to *MU-BAAL*. Practically, when a new sample is needed, the uncertainty region is sampled as follows. A hypothesis  $h_{UR}$  is sampled in the version space of the current  $s_t$ ; ii) a sample  $x$ , uniformly selected in the instance space, is projected onto  $h_{UR}$ , thus defining a sample  $x_{UR}$  (i.e.  $h_{UR}(x_{UR}) = 0$ ); iii)  $x_{UR}$  is returned. The rationale behind this heuristics is that hypotheses “close to”  $h_{UR}$  likely also belong to the version space;  $x_{UR}$  being on the frontier of  $h_{UR}$  will thus be classified differently by hypotheses close to  $h_{UR}$ , that is,  $x_{UR}$  belongs to the uncertainty region. In the experiments, hypothesis space  $\mathcal{H}$  is set to the set of separating hyperplanes; the projection of  $x$  onto  $h_{UR}$  thus is straightforward.

## 5.3 Goal of the experiments

The experimental validation of *BAAL* will be conducted in order to answer three questions. The first question naturally regards the performance of *BAAL* w.r.t the dimension of the instance space and the computational resources available, namely the time horizon and the number  $N$  of tree-walks allowed; along the same line, the impact of the billiard algorithms on the scalability of the approach is examined. A second question relates to the dilemma *Better vs More* simulations. Two types of refinement have been considered, enforcing low-variance estimates of the reward (section 5.1), or the selection of samples in the uncertainty region (section 5.2).

## 6 Experimental validation

This section describes the experimental setting and reports on the validation of the approach.

## 6.1 Experimental setting

Following (Freund *et al.*, 1997) and followers, the hypothesis space  $\mathcal{H}$  is set to the homogenous separating hyperplanes on the instance space  $X = \mathbb{R}^d$ . Distribution  $P_H$  is uniform on the unit sphere of  $\mathbb{R}^d$ .

Each run is based on the selection of a target hypothesis  $h$ ; a  $T$ -size training set  $s_T$  is built, where instance  $x_1$  is set to  $BAAL(\{\}, T, N)$ ,  $x_2$  is set to  $BAAL(\{(x_1, h(x_1))\}, T - 1, N)$ , and so forth. The performance of the run is  $Err(\mathcal{A}(s_T), h)$ , where learner  $\mathcal{A}$  uniformly samples  $\mathcal{H}(s_T)$ . The overall performance is averaged over 300 independent runs. Time horizon  $T$  is set to 10. Dimension  $d$  is set to 2, 4, and 8;  $N$  is set to  $2^i$  for  $i = 0, \dots, 18$ .

Two baseline algorithms are considered: the random active learner (RND) and the greedy Maximum Uncertainty, selecting in each step the example with maximum uncertainty in a 1,000-sample pool (MU-1000). By construction, the performance of the baseline corresponds to  $N = 1$  tree-walks.

## 6.2 Performance and Scalability

RND-BAAL (using a randomly ordered 1,000 sample pool) significantly improves on the baseline as the number  $N$  of tree-walks increases (Fig. 2 (a) and (b)); the baseline performances correspond to  $N = 1$ ). The generalization error decreases from circa 7% to 1% (respectively from 28% to 23%) for  $N = 2^{15}$  in dimension 2 (resp. dim. 8).

MU-BAAL (sorting a 1,000 sample pool after Maximum Uncertainty) reaches the same

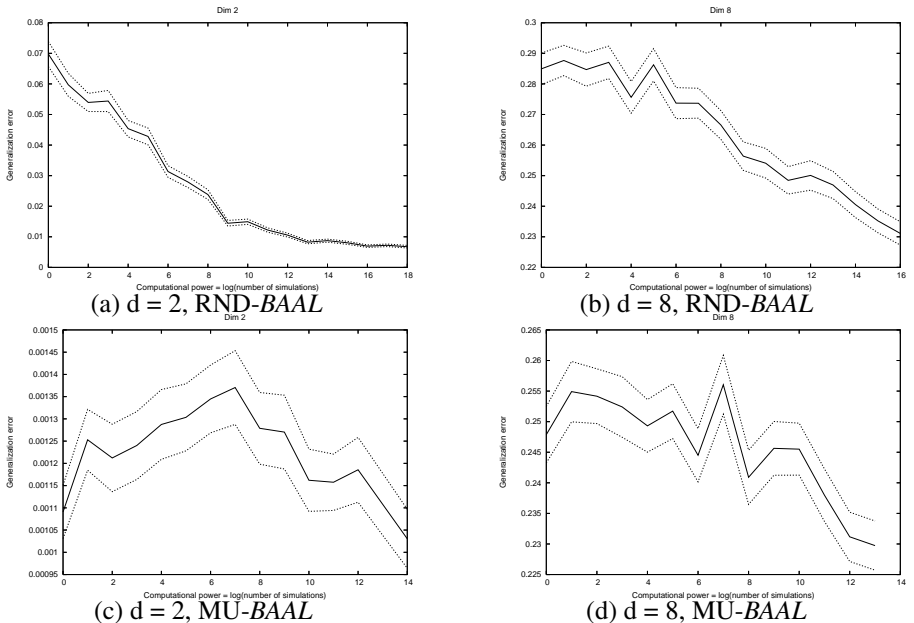


Figure 2: BAAL: Generalization error vs  $\log(N)$  (computational effort), time horizon 10, dimension = 2,8.

Dim. $d$	Horizon $T$	Reject	Billiard
4	10	87s	40s
	20	1h10m	138s
8	10	300s	51s
	20	>1 day	169s

Table 1: Computational cost of billiard-based sampling and rejection-based sampling w.r.t dimension  $d$  and time horizon  $T$ , for  $N = 16,000$

performances as MU-based active learning, aka Query-by-Committee (Freund *et al.*, 1997) in dimension 2, whatever the computational effort. This result is explained as the maximum uncertainty criterion is quasi-optimal for homogeneous hyperplanes in dimension 2 (Dasgupta, 2005). Interestingly, in dimension 8 MU-BAAL *does* improve on Query-by-Committee; a tentative interpretation for this fact is that the 1,000 sample pool is sorted after a 100 committee; as the sample order becomes less accurate due to the higher dimension, it offers some room for improvement, and MU-BAAL does modestly but significantly improve on Query-by-Committee.

Independently, the billiard-based sampling of the version space  $\mathcal{H}(s_T)$  is compared to the rejection-based sampling (uniformly drawing  $h$  in  $\mathcal{H}$ , until  $h$  belong to  $\mathcal{H}(s_T)$ ); with same performances, billiard-based sampling shows one or several orders of magnitude cost reduction (Table 1).

### 6.3 More vs better simulations

Several variants of BAAL have been investigated (results omitted due to lack of space). Unexpectedly, Av(R)-BAAL, involving a low-variance reward estimate through averaging the reward over  $R = 10$  or  $R = 50$  hypotheses sampled in  $\mathcal{H}(s_T)$  (section 5.1) does not improve the performance for  $d = 4$  or 8, (Fig. 3.(a)) while significantly increasing the computational cost. Similar results were observed in the context of computer-Go (Gelly & Silver, 2007). A tentative interpretation goes as follows: the more sophisticated the simulations, the more likely they suffer from some hidden bias, and the more chances they mislead the game strategy. The sampling of the uncertainty region (UR-BAAL, (Fig. 3.(b)) significantly improves on both RND-BAAL and MU-BAAL.

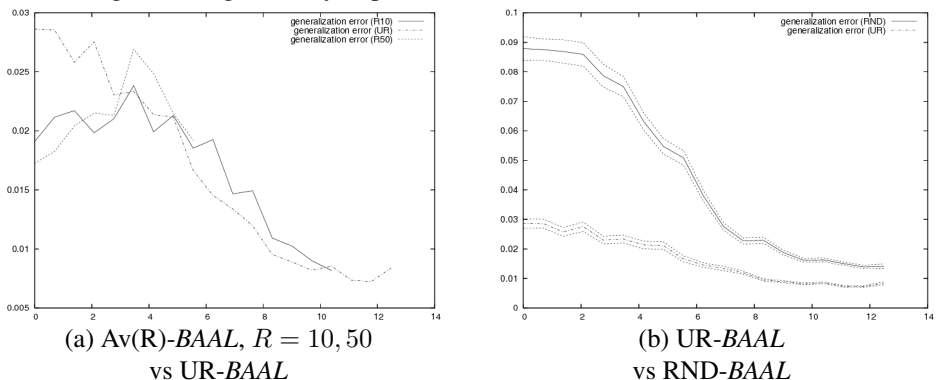


Figure 3: BAAL variants: Time horizon 20, dimension 4

## 7 Conclusion and Perspectives

The main contributions of this paper regard AL with bounded computational resources. It is believed that this perspective is relevant to many potential application domains of ML, such as Numerical Engineering, which can only afford some tens or hundreds of labelled examples as computing the response of a single sample might require days or weeks of computation.

The presented *BAAL* algorithm relies on a sound formalization of AL as a finite horizon reinforcement problem. Its practical implementation relies on viewing active learning as a one-player game, tackled through a tree-structured multi-armed bandit algorithm. A proof of principle of the validity of the approach shows convincing empirical evidence. Further developments include the extension of *BAAL* to non-linear hypothesis spaces, including the development and analysis of billiard algorithms in such spaces.

## References

- ASTROM K. (1965). Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, **10**, 174–205.
- AUER P. (2003). Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research*, **3**, 397–422.
- CASTRO R., WILLETT R. & NOWAK R. (2006). Faster rates in regression via active learning. In Y. WEISS, B. SCHÖLKOPF & J. PLATT, Eds., *Advances in Neural Information Processing Systems 18*, p. 179–186. Cambridge, MA: MIT Press.
- CHASLOT G., WINANDS M., UITERWIJK J., VAN DEN HERIK H. & BOUZY B. (2007). Progressive strategies for monte-carlo tree search. In P. WANG & OTHERS, Eds., *Proc. of the 10th Joint Conf. on Information Sciences*, p. 655–661: World Scientific Publishing.
- COHN D., ATLAS L. & LADNER R. (1994). Improving generalization with active learning. *Mach. Learn.*, **15**(2), 201–221.
- COHN D., GHAHRAMANI Z. & JORDAN M. (1996). Active Learning with Statistical Models. *Journal of Artificial Intelligence Research*, **4**, 129–145.
- COULOM R. (2006). Efficient selectivity and backup operators in Monte-Carlo tree search. In P. Ciancarini and H. J. van den Herik, editors, *Proc. of the 5th Int. Conf. on Computers and Games*.
- DASGUPTA S. (2005). Analysis of a greedy active learning strategy. In L. K. SAUL, Y. WEISS & L. BOTTOU, Eds., *NIPS04*, p. 337–344. MIT Press.
- DASGUPTA S. (2006). Coarse sample complexity bounds for active learning. In Y. WEISS, B. SCHÖLKOPF & J. PLATT, Eds., *NIPS05*, p. 235–242. MIT Press.
- FREUND Y., SEUNG H. S., SHAMIR E. & TISHBY N. (1997). Selective sampling using the query by committee algorithm. *Mach. Learn.*, **28**(2-3), 133–168.
- GELLY S. & SILVER D. (2007). Combining online and offline knowledge in uct. In *ICML '07*, p. 273–280, New York, NY, USA: ACM Press.
- HANNEKE S. (2007). A bound on the label complexity of agnostic active learning. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, p. 353–360, New York, NY, USA: ACM.
- HERBRICH R., GRAEPEL T. & CAMPBELL C. (2001). Bayes point machines. *Journal of Machine Learning Research*, **1**, 245–279.
- HOI S. C. H., JIN R., ZHU J. & LYU M. R. (2006). Batch mode active learning and its application to medical image classification. In *ICML '06*, p. 417–424, New York, NY, USA: ACM.
- IYENGAR V. S., APTE C. & ZHANG T. (2000). Active learning using adaptive resampling. In *KDD00*, p. 91–98.
- KOCSIS L. & SZEPESVARI C. (2006). Bandit-based monte-carlo planning. In *ECML'06*, p. 282–293.
- KULKARNI S. R., MITTER S. K. & TSITSIKLIS J. N. (1993). Active learning using arbitrary binary valued queries. *Mach. Learn.*, **11**(1), 23–35.
- LINDENBAUM M., MARKOVITCH S. & RUSAKOV D. (2004). Selective sampling for nearest neighbor classifiers. *Machine Learning*, **54**, 125–152.
- ROY N. & MCCALLUM A. (2001). Toward optimal active learning through sampling estimation of error reduction. In *ICML01*, p. 441–448: Morgan Kaufmann, San Francisco, CA.
- RUIJAN P. & MARCHAND M. (2000). *Advances in Large Margin Classifiers*, chapter Computing the Bayes Kernel Classifier, p. 329–347. MIT Press.
- SCHOHN G. & COHN D. (2000). Less is more: Active learning with support vector machines. *ICML00*, **282**, 285–286.
- SEUNG H. S., OPPER M. & SOMPOLINSKY H. (1992). Query by committee. In *COLT '92*, p. 287–294, New York, NY, USA: ACM.
- SUTTON R. & BARTO A. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.
- VIDYASAGAR M. (1997). *A Theory of Learning and Generalization, with Applications to Neural Networks and Control Systems*. Springer-Verlag.
- WANG Y., AUDIBERT J.-Y. & MUNOS R. (2008). Algorithms for infinitely many-armed bandits. In *NIPS08*, p. to appear.
- XIAO G., SOUTHEY F., HOLTE R. C. & WILKINSON D. (2005). Software testing by active learning for commercial games. In *AAAI-05*, p. 609–616.