ACTIVE LEARNING IN REGRESSION, WITH APPLICATION TO STOCHASTIC DYNAMIC PROGRAMMING

Olivier Teytaud*, Sylvain Gelly*,Jérémie Mary* **
**TAO (Inria, Univ. Paris-Sud, UMR CNRS-8623), France*
***Grappa (Inria Univ. Lille), France*
teytaud@lri.fr, gelly@lri.fr,mary@lri.fr

Keywords: Intelligent Control Systems and Optimization, Machine learning in control applications, Active learning

Abstract: We study active learning as a derandomized form of sampling. We show that full derandomization is not suitable in a robust framework, propose partially derandomized samplings, and develop new active learning methods (i) in which expert knowledge is easy to integrate (ii) with a parameter for the exploration/exploitation dilemma (iii) less randomized than the full-random sampling (yet also not deterministic). Experiments are performed in the case of regression for value-function learning on a continuous domain. Our main results are (i) efficient partially derandomized point sets (ii) moderate-derandomization theorems (iii) experimental evidence of the importance of the frontier (iv) a new regression-specific user-friendly sampling tool less-robust than blind samplers but that sometimes works very efficiently in large dimensions. All experiments can be reproduced by downloading the source code and running the provided command line.

1 INTRODUCTION

As pointed out in e.g. (Cohn et al., 1995a), the ability of the learner to select examples and modify its environment in order to get better examples is one of the main point in learning. In this model of learning, the learning algorithm is typically made of (i) a sampler, that chooses points in the domain, and (ii) a passive learner that takes these points and their labels as provided by some unknown oracle (target concept in classification or target function in regression). Various forms of active learning have been proposed:

1. Blind approaches in which points to be labelled by the oracle are well distributed in the domain, e.g. in a quasi-random manner, without taking into account the labels (Cervellera and Muselli, 2003); in this case, the process can be splitted in 3 successive steps: (i) sample the points, (ii) label by the oracle, (iii) learn the target concept/function;

2. Non-blind approaches, in which the sampler uses the labels provided by the oracle in order to choose the next points to be labelled; possibly, the learner is embedded in the sampler. These approaches use various criteria; (Lewis and Gale, 1994) chooses examples with the maximum uncertainty of the learner, (Seung et al., 1992) chooses examples that reduce maximally the size of the version space. Other approaches include (Schohn and Cohn, 2000), with application to Support Vector Machines (SVM), (Cohn et al., 1995b), with application to Neural Nets.

Limitations of non-blind approaches. In spite of the fact that the second approach is much more general, the superiority of non-blind approaches is often unclear, due to the nice robustness properties of blind approaches. If you exploit properties of the problem to avoid sampling areas that are probably less interesting, you can miss some ”surprisingly” interesting areas. Pessimistic theorems on the possibility of active learning can be found in e.g. (Vidyasagar, 1997), in a worst-case scenario. Mainly, as a conclusion of the state of the art, the best applied results for non-blind active-learning concern moderately large families of functions; classification more than regression, decision trees more than neural networks or SVM. In particular, we think that a main advantage of viability approaches in reinforcement learning is that it reduces the problem (with loss of generality unfortunately) to a classification problem in which active learning is more stable (Chapel and Deffuant, 2006). This is not very surprising as for example, choosing

points close to boundaries, what makes sense in classification or with regression-trees but not in numeric regression, is a good solution for efficient active learning. We propose in this paper a new non-blind approach, not always better than blind approaches as we will see in the sequel, but that (i) is easy to use for any type of state space (continuous or not) (ii) can be parametrized continuously from the pure blind approach to a very deterministic sampling (iii) can easily integrate expert information about the sampling.

Limitations of deterministic approaches. Deterministic samplings are not always better than pure random sampling. E.g. in numerical integration, deterministic blind samples are much better than random points for various criteria in small dimensions, but when the dimension increases, these strictly deterministic approaches (Sloan and Woźniakowski, 1998) have strong limitations (see also (Kearns et al., 1999; Rust, 1997) for some interesting properties of random sampling in the specific case of control problems). Best results in the traditional field of quasi-random sequences, namely integration, now come from *randomized* quasi-random sequences, in particular when dimension increases (L'Ecuyer and Lemieux, 2002), whereas former best sequences were strictly deterministic ones. This is a somewhat surprising element in the history of quasi-random sequences. We show here both empirically and theoretically a similar superiority of randomized, yet non-naive, samplings in the case of active learning. We also conclude empirically to some related limitations of non-blind approaches w.r.t blind approaches in terms of robustness but our theorems only concern almost-deterministic samplings, and non-blind approaches are concerned only if too strongly deterministic.

Why active learning is very important in dynamic problems. The importance of the exploration step is particularly strong in reinforcement learning, where exploitation (learning) is deeply mixed with exploration (gathering information); this is why active learning is decisive in particular for dynamic problems. A main trouble, with respect to elements above, is that value functions used in reinforcement learning lead to regression problems and not classification ones. Note that many works about active sampling of the environment use simulations; see e.g. the pioneer work (Barto et al., 1993) and many subsequent works. We will here avoid these techniques, that have strong advantages but also limitations as they can miss interesting parts of the state space that are not seen in simulations due to poor initial policies. We here only consider sampling methods that sample as efficiently as possible the 'full' domain. This is orthogonal to, and not competing with, simulation-based samplers. Examples of non simulation-based active learning for dynamic problems have been provided in (Munos and Moore, 1999) (active discretization of the domain), (Chapel and Deffuant, 2006) (active learning for SVM in a viability-framework which reformulates the regression task in a classification task).

Overview of results. We will here study the following questions, in the case of regression:
1. Is non-blind active-learning better than blind active learning? Essentially, we will see significant differences in both senses; for some problems the results are indeed much worse. This is an experimental answer but we provide also proofs for the related robustness of randomized techniques w.r.t too-much-deterministic methods. A non-blind sampler termed EAS is defined and tested.
2. Are deterministic blind-samplers better than random ones? Essentially, we prove (th. 2.4) robustness (universal consistency) results for random samplers, and non robustness for deterministic samplers. We then show that the quantity of randomness can be strongly reduced, in a classical random-shift framework preserving both (i) improved convergence rates (as shown in (Cervellera and Muselli, 2003)) for "smooth-enough" target-functions and (ii) universal consistency (as shown in this paper) for any target function. We propose and test an efficient blind-sampler termed GLD.

2 MATHEMATICAL ANALYSIS

We study here the derandomized samplings, i.e. sampling with less randomness. It has been shown (Cervellera and Muselli, 2003) that derandomizing improves convergence rates for smooth target functions when compared to naive random samplers. We show that however, some robustness properties require a random part (theorem 2.3), but that this random part can be reduced to combine (i) the robustness of naive random sampling (ii) the improved convergence rates of deterministic samplers for "easy" cases.

Definition 2.1 *Consider a domain $D = [0,1]^d$. Note E^* the set of finite families of elements of E. A* **learner** *A on D is a computable mapping from $(D \times \mathbb{R})^*$ to $\mathbb{R}^D$. Let $\mathcal{A}$ the set of learners. A* **sampler** *S on D is a computable mapping from $(D \times \mathbb{R})^* \times \mathcal{A}$ to D. If S can be restricted to a computable mapping from $(D \times \mathbb{R})^*$ to D (i.e. if the $\mathcal{A}$-component is not used) then it is called a* **blind-sampler** *A* **sample-learner** *on D, based on the sampler S and on a learner A, and noted $S+A$, is an algorithm that takes as input*

an effective method f on a domain D and that can be written as follows:

1. *set $n \leftarrow 0$;*
2. *choose x_n in D defined by $x_n \leftarrow S((x_0,y_0)\ldots,(x_{n-1},y_{n-1}),A)$;*
3. *define $y_n \leftarrow f(x_n)$;*
4. *set $f_n = A((x_0,y_0),\ldots,(x_n,y_n))$.*
5. *set $n \leftarrow n+1$ and go back to step 2.*

A sampler is said **almost-deterministic** *(AD) if for some $n \mapsto k(n)$ finite it only uses $k(n)$ random bits before choosing x_n (i.e. before the n^{th} epoch of the algorithm above). A learner is said* **almost-deterministic** *(AD) if for each run it only uses a finite number of random bits that only depends on the length of its inputs. A sample-learner on D is said* **universally consistent** *(UC) if, for any measurable f with values in $[0,1]$, almost surely $\int_D (f_n(x)-f(x))^2 dx \to 0$ as $n \to \infty$. We say that a sampler S on D is* **universally consistent** *(UC) if for at least one AD learner A, the sampler-learner $S+A$ is UC.*

We require in the definition of UC for a sampler that the learning algorithm is AD because otherwise stochasticity of the learner can be used for randomizing the sampler. Therefore, to distinguish AD-samplers and non-AD-samplers, we have to add this restriction.

Theorem 2.2 *The random sampling is UC.*

Proof: By any UC-algorithm (see e.g. (Devroye et al., 1994)).

Theorem 2.3 (UC samplers are stochastic) *Consider S an AD-sampler. Then, for any AD-learner A, $S+A$ is not UC. Therefore, S is not UC.*

Proof: Consider S and A, respectively an AD-sampler and an AD-learner. Then, consider $x_0,\ldots,x_n,\ldots$ the sequence of points provided by the sampler if f is the constant function equal to 1 (this sequence might be stochastic or not). By definition of AD, x_i takes values in a finite domain of cardinal c_i. Therefore, all the x_n take values in some countable domain V. Consider now g_p, for $p \in [0,1[$, the function equal to 1 in V, and to p in $D \setminus V$. If $S+A$ is UC, then on a target $f = g_p$ almost surely f_n converges in norm L^2 to the constant function equal to p. However, for all $f = g_p, p \in [0,1[$, f_n is distributed in the same manner, as f_n depends only on the f values in V. With $p = 0$ and $p = \frac{1}{2}$ for example, this leads to a contradiction. $S+A$ can not be UC. □

We showed in theorem 2.2 that random sampling is UC and in theorem 2.3 that no AD-sampling can be UC. But where is the limit? We show in the following theorem that a moderate derandomization, yet using a continuous random seed, can lead to universal consistency. Note that the result includes quasi-random sequences with a simple random shift, and therefore includes samplings that have proved faster under some smoothness hypothesis than the random sampling (Cervellera and Muselli, 2003).

Note that the theorem below has a very moderate requirement in terms of discrepancy (only convergence to 0, whereas low-discrepancy sequences typically verify $O(\log(n)^d/n)$).

Theorem 2.4 (A random shift is enough) *Consider a sampler S that outputs $x_0,\ldots,x_n$ and defined as follows:*

1. randomly uniformly draw $s \in D$.

2. $y_0,\ldots,y_n,\ldots$ is a deterministic sequence in D with dispersion

$$\sup_{x \in D} \inf_{i \in [[1,n]]} ||x - y_i||_\infty = O(1/n^{1/d})$$

and discrepancy

$$\sup_{r \in [0,1]^d} |\frac{1}{n} \#\{i \in [[1,n]]; \forall j, x_{i,j} \leq r_j\} - \pi_{i \in [[1,d]]} r_i| \to 0$$

as $n \to \infty$ where $x_{i,j}$ is the j^{th} component of x_i.

3. for any n, $x_n = y_n + s$ modulo 1 (i.e. x_n is such that $y_n - x_n \in \mathbb{Z}^d$ and $x_n \in [0,1[^d$).

Then, S is UC.

Interpretation: Adding a uniform random shift to a deterministic sequence is enough to get a UC sampler with the same improved convergence rates as in (Cervellera and Muselli, 2003) for smooth target functions. Comparing theorems 2.3 and 2.4 show that randomness is required for UC, but the quantity of randomness can be much smaller than in pure random sampling.

Proof: See `http://www.lri.fr/~teytaud/ldsfordplong.pdf`.

3 SAMPLING METHODS (SM) AND APPLICATIONS

We present below (i) the problem of function value learning (ii) some blind point-sets (iii) a non-blind learner-independent sampler (iv) our experiments on the OpenDP set of benchmark-problems.

3.1 Active function value learning

We introduce reinforcement learning and stochastic dynamic programing (SDP) in a very short manner; see (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998) for more information. The general idea is that a

good control for a control problem in discrete time is reached if for each of the finitely many time steps, one can learn efficiently the "expected cost-to-go" (also named Bellman-function or value-function in various fields of mathematical programming and computer science), i.e. a function that maps a state x and time step t to the expected cost (for an optimal strategy) starting from this state x at time t. This cost-to-go has to be learnt from examples. These examples can be sampled, and a procedure termed *Bellman-operator* (BO) provides the cost-to-go for each given example.

Roughly, the classical SDP algorithm, based on a sampler S and a learner L, is as follows. For each time step t, backwards from the last time step to the first: 1. S samples a finite set of examples $x_1,\ldots,x_n$ at time step t; 2. The BO (detailed below) compute the expected cost-to-go for each of these examples (using the previously learnt expected cost-to-go, i.e. the cost-to-go at time step $t+1$); 3. L learns the expected cost-to-go V_t at time step t from these examples.

The BO-procedure computes the cost-to-go at state x for time step t using simulations of the transition from time step t to time step $t+1$ and using the cost-to-go-function V_{t+1} at step $t+1$ as follows: (i) simulate each possible random outcome (ii) for each random outcome get by simulation the instantaneous cost and the future state y and use the cost-to-go $V_{t+1}(y)$ from y at time step $t+1$ (available thanks to the backward induction) for estimating the future-cost (iii) add these two costs for each random outcome, (iv) average all these results to get the required value, i.e. the expected cost-to-go $V_t(x)$.

The sampler is important in order to reduce the computational cost, a main issue of SDP. We here use tools that are not specific of SDP. More specific approaches for dynamic problems also exist; they are more orthogonal to, than comparable to our approaches below. (Barto et al., 1993) showed the importance of "reducing" the domain, when possible, by using simulations. If only a small subset of the full state space is interesting, this approach restricts the focus of the algorithm to the neighborhood of the visited states. A drawback is that you need an approximate solution before applying simulations, and you need simulations in order to reduce the domain and apply dynamic programming; therefore, this approach leads to complex unstable fixed-point iterations. However, this approach can deal with much bigger domains than the approach sampling the whole domain. (Thrun, 1992) studied how to avoid visiting many times the same area (leading to better guaranteed rates in some theoretical framework), and then reduce the curse of dimensionality.

3.2 Blind active-samplers

We present below point sets in $D=[0,1]^d$. P will be the notation for a set of points $P_1,\ldots,P_n$. $\#E$ is the cardinal of a set E. See e.g. (Tuffin, 1996; Owen, 2003; L'Ecuyer and Lemieux, 2002) for a general introduction to "well chosen" sets of points.

Low discrepancy. The most usual discrepancy is defined as follows:

$$Disc(P)=\sup_{r\in[0,1]}\left|area([0,r])-\frac{1}{n}\#\{i\in[[1,n]];P_i\leq r\}\right|$$

where $[0,r]$ is the set of q such that $\forall i\in[[1,d]]0\leq q_i\leq r_i$ and $area()$ is Lebesgue's-measure. Independent uniform random points have discrepancy roughly decreasing as $O(1/\sqrt{\#P})$. Well chosen deterministic or (non-naive) randomized points achieve $O(\log(\#P)^d/\#P)$. See `http://www.lri.fr/~teytaud/ldsfordplong.pdf` for further elements. We will in the sequel call "low-discrepancy sequence" a classical Niedereiter sequence (see e.g. (Niederreiter, 1992)). We will refer to this sequence as a QR (quasi-random) sequence in the sequel.

Low dispersion. Low dispersion is less widely used than low-discrepancy, but has some advantages (see e.g. discussions in (Lindemann and LaValle, 2003; LaValle and Branicky, 2002)). The most usual criterion (to be minimized) is

$$Dispersion(P)=\sup_{x\in D}\inf_{p\in P}d(x,p) \qquad (1)$$

where d is the euclidean distance. It is related to the following (easier to optimize numerically, except for some values of $\#P$) criterion (to be maximized) :

$$Dispersion_2(P)=\inf_{(x_1,x_2)\in D^2}d(x_1,x_2) \qquad (2)$$

A main difference is that optimizing eq. 2 "pushes" points on the frontier. This effect can be avoided as follows :

$$Dispersion_3(P)=\inf_{(x_1,x_2)\in D^2}d(x_1,\{x_2\}\cup D') \qquad (3)$$

where $D'=\{x\in\mathbb{R}^d;x\notin D\}$. We call "low-dispersion point set" (LD) a point set optimizing equation 2. We call "greedy-low-dispersion point set" (GLD) a point set optimizing equation 2 in a greedy manner, i.e. $P_1=(0.5,...,0.5)$ is the middle point of $[0,1]^d$, P_2 is such that $Dispersion_2(\{P_1,P_2\})$ is maximal, and P_n is such that $Dispersion_2(\{P_1,\ldots,P_{n-1},P_n\})$ is maximal. Our implementation is based on Kd-trees but Bkd-trees are possible (bkdtrees are similar to well-known kdtrees, but also allow fast online adding of

points; see (Procopiuc et al., 2002)). We call "greedy-low-dispersion point set far-from-frontier" (GLDfff) the equivalent point set with $dispersion_3$ instead of $dispersion_2$. We also tested the same dispersion with other L^p-distances than the euclidean one, without success. GLD is very successful, as shown in the sequel; we show a sample in figure 1.

Figure 1: A GLD-sample in dimension 2. Note that the random choice of points among various possible optimal points avoids (probably) too unequilibrated sets. The first points are shown with darker circles. There is randomness in the choice of the next point among a given gray-level.

3.3 A non-blind active-sampler

Many tools for active-learning exist (see introduction) but can not be used in our case as they are classification-specific or known moderately efficient in the regression case. We here propose the use of an evolutionary algorithm. This approach is new in the case of active learning and in the case of SDP, but it is inspired by evolutionary sampling (Liang and Wong, 2001). Its advantages are (i) it is user-friendly (ii) it works better than random and also than derandomized blind point sets at least for a non-negligible set of benchmarks.

Evolutionary algorithms (EA,(Baeck, 1995; Eiben and Smith, 2003)) are a classical tool for robustly optimizing a non-linear function without requiring derivatives. We here use them as a sampling algorithm, as follows: 1. generate an initial population uniformly on the domain; 2. evolve the population until the allowed number of fitness evaluations; 3. use as active sample the union of all offsprings of all epochs. Note that the learning algorithm is not embedded in this approach which is therefore quite general.

We define precisely below our EA, but we point out that any EA could be used as well. The parameters have not been specialized to our problem, we have used a tool that has been designed for optimization purposes and not for our particular application to sampling. Evolutionary algorithms exist for various forms of domains, continuous, discrete or mixed, and therefore our approach is quite general. Estimation of distribution algorithms could be used as well(Larranaga and Lozano, 2001). The EA that we use can be found in the *sgLibrary* (`http://opendp.sourceforge.net`). It implements a very simple genetic algorithm where the mutation is an isotropic Gaussian of standard deviation $\frac{\sigma}{\sqrt[d]{n}}$ with n the number of individuals and d the dimension of space. The crossover between two individuals x and y gives birth to two individuals $\frac{1}{3}x+\frac{2}{3}y$ and $\frac{2}{3}x+\frac{1}{3}y$. Let $\lambda_1,\lambda_2,\lambda_3,\lambda_4$ be such that $\lambda_1+\lambda_2+\lambda_3+\lambda_4=1$; at each generation :

1. we copy the $n\lambda_1$ best individuals (set S_1).
2. we combine the $n\lambda_2$ following best individuals with the individuals of S_1 (rotating among S_1 if $\lambda_1<\lambda_2$).
3. we mutate $n\lambda_3$ individuals among S_1 (again rotating among S_1 if $\lambda_1<\lambda_3$).
4. we randomly generate $n\times\lambda_4$ other individuals, uniformly on the domain.

The parameters are $\sigma=0.08,\lambda_1=1/10,\lambda_2=2/10,\lambda_3=3/10,\lambda_4=4/10$; these parameters are standard ones from the library and have not been modified for the work presented in this paper. The population size is $n=N^\alpha$, where $\alpha\in]0,1]$ is a parameter of the approach. Note that this algorithm is oriented towards sampling small values of the target function. Therefore, it is based on the idea that small values are more interesting. In the case of SDP, the target function is the sum of the instantaneous cost and the expected cost-to-go. Thus, this active sampling is efficient if we are dealing with a problem in which it is likely that trajectories are close to small values of this target function. We will see in our experiments that this is true in some problems, but not e.g. in stock management, leading to poor results in that case. We here see that the non-blind approach developed in this section looks appealing, but has robustness drawbacks. The tuning of α is a possible solution: $\alpha=1$ leads to the pure random sampling; smaller α leads to a more optimistic approach in which the sample is reinforced in parts for which costs are better (smaller), at the price of a weaker ability to avoid very large costs.

3.4 Experiments

The domain to be sampled is made of the continuous state space and a discrete state space. The samplers have then to sample a product $S\times M$, with S the continuous state space and $M=\{m_1,\ldots,m_k\}$ discrete and finite, being the exogenous Markov-Process. The following SM are used:

1. The SMs GLD, QR, LD, GLDfff are the blind approaches defined in section 3.2. The discrete parts of the state-space are sampled in a simple proportional manner: for sampling n points in $S\times M$, where S is the continuous state space and M is a discrete domain $M=\{m_1,\ldots,m_k\}$, a GLD, QR, LD or GLDfff point set is used for sampling S

with $N = n/k$ points $x_1, \ldots, x_N$, and the sample is $((x_1, m_1), \ldots, (x_N, m_1), \ldots, (x_1, m_k), \ldots, (x_N, m_k))$.

2. We have various random samplings. The SM RandomForAll denotes the pure blind (uniform, independent) random sampling of $S \times M$. The SM RandomOnlyForContinuousStates (ROFCS) denotes the pure blind random sampling of S, with a proportional sampling of M (i.e. m_1 appears the same number of times as m_2, ..., as m_k). The SM DeterministicForDim1 is equal to ROFCS, except that for each value m_i, the first coordinate of S is sampled by a regular grid (other coordinates are random). This is therefore deterministic if S is one-dimensional.

3. EAS-α denotes the non-blind approach using EA-sampling, as explained in section 3.3. The parameter is the α-parameter in section 3.3. All the experiments have been performed with the OpenDP library (Gelly and Teytaud, 2005) with the command line options "./runme.sh –nojava –nogui –test OptimizationExperiments –outputFile myResultsFile.txt –testedOptimizers '[[GeneticAlgorithmOptimizer]]' –testedRegressions '[[AutomaticRegression]]' –numExampleWanted X –nbPointsSamplingMethod 500 –nbRuns 44 –nbTryEvaluationsOptimization 70" where X$\in \mathbb{N}$ is the index of the problem. The parameters of the experiments are as follows: 500 points are sampled on the domain (same number of sampled points for each algorithm); all results are on 44 runs; the learner is the "AutomaticRegression" class from (Gelly and Teytaud, 2005), that uses the Gaussian-SVM of SVMTorch (Collobert and Bengio, 2001) and a heuristic rule choice of hyper-parameters. All problems are described in (Gelly et al., 2006) and in (Gelly and Teytaud, 2005) (including free downloads on `http://opendp.sourceforge.net`). The dimensionality of problems is the dimension of the continuous state space, excluding the discrete parts. The objective functions are to be minimized (lower values = better results).

Results about derandomized blind sampling. Results are summarized in table 1. Due to length constraints, all the details with confidence intervals, and results for other methods, are provided in `http://www.lri.fr/~teytaud/ldsfordplong.pdf`. The dimension refers to the dimension of the state space. Columns QR (resp. GLD, GLDfff), refer to the fact that QR-sampling (resp. GLD-sampling, GLDfff-sampling) outperforms the baseline random algorithm, namely ROFCS (in which the continuous part is pure independent random and the discrete part is proportional sampling); we also compared the algorithms to pure random sampling, which is usually worse than ROFCS (which can be seen as a very simple preliminar derandomization). For the column mentionning the fact that GLD is first-ranked, "**y**" stands for "GLD is significantly better than all other algorithms" and "y(=ST)" stands for "GLD is first-ranked and significantly better than all other algorithms except algorithm ST for which there's no statistically significant difference".

Results about non-blind active regression. We note EAS the algorithm with parameter α for non-blind active sampling of the state space defined in 3.3. In moderate dimension, the efficiency of EAS is very moderate; we here present results with high dimensionality and 500 points sampled per time step. We compared EAS, GLD and the best blind sampler on this problem. All results are averaged on 44 runs.

Due to length constraints, details for all problems are provided in `http://www.lri.fr/~teytaud/ldsfordplong.pdf` and we here show only results for the "Arm" and "Multi-Agent" problem; in summary for other problems: for Avoid (dim=8) and Walls (dim=8), GLD outperforms the EAS for any value of α; for Arm (dim=12) and Multi-agent (dim=8), EAS outperforms GLD for all values of α; for Shoot (dim=12) there's not significant difference in results; for Away (dim=8) the EAS outper-

Table 1: Blind sampling methods. First, we see that GLD is often the best blind SM. It outperforms the random-sampling ROFCS in 7 out of 9 experiments. Second, other derandomized samplings are only sometimes better than random-sampling with proportional sampling for discrete parts of the state space; essentially, the difference between these other less-randomized samplings and ROFCS is not significant. The pure naive random sampling, RandomForAll (which includes random sampling of the discrete part also), is not presented in this table as it is meaningless in some cases (when there's no discrete part); its results, when meaningful, are poor. As a conclusion, (i) derandomizing the sampling for the discrete part is strongly better (of course, at least in problems for which there is a discrete part) (ii) GLD is a stable and efficient partially-derandomized sampler of continuous domains.

Problem (dim)	GLD 1st ranked	QR	GLD	GLDfff
WallsX4 (8)	**y**	y	y	n
AvoidX4 (8)	**y**	y	y	y
Stock (4)	**y**	n	y	n
Avoid (2)	y (=GLDfff)	y	y	y
Arm (3)	n	n	n	n
Walls (2)	**y**	y	y	n
Multiag. (8)	n	n	n	y
Shoot (3)	**y**	y	y	y
Away (2)	y (=QR)	n	y	n
Total	7/9	5/9	7/9	4/9

Table 2: Results for the "arm" and "multi-agent" problems. For the "multi-agent" problem, EAS is very efficient; this is probably related to the fact that intuitively, the criterion focusing on "good regions" (where the expected cost-to-go is small) is satisfactory for this problem: large bad regions of the domain are very unlikely thanks to the existence of reasonnable good paths. These large bad regions are explored by blind samplers. The criterion that we plug in EAS is relevant for this case, and should be adapted for other problems in order to get similar good results. EAS has a performance equivalent to ROFCS for the "arm" problem.

Arm dim.x4 = 12		Biagent dim.x4 = 8	
Sampler	Average score	Sampler	Average score
EAS-0.2	10.34±0.21	EAS-0.2	2295.23±16.08
EAS-0.3	10.17±0.20	EAS-0.3	2311.59±13.46
EAS-0.4	10.46±0.23	EAS-0.4	2179.09±15.62
EAS-0.5	10.62±0.21	EAS-0.5	2175.45±16.11
EAS-0.6	10.49±0.20	EAS-0.6	2156.59±15.66
EAS-0.7	10.54±0.23	EAS-0.7	2137.5±13.41
EAS-0.75	10.53±0.22	EAS-0.75	2167.95±14.26
EAS-0.8	10.37±0.20	EAS-0.8	2195±16.62
EAS-0.85	10.17±0.20	EAS-0.85	2184.09±13.91
EAS-0.9	10.44±0.20	EAS-0.9	2170.23±14.22
EAS-0.95	10.38±0.21	EAS-0.95	2245.91±12.39
ROFCS	10.39± 0.20	best-blind	
GLD	12.52 ±0.25	= GLD	2326.59 ±16.17

form GLD moderately significantly or significantly depending on α except for $\alpha = 0.95$.

4 CONCLUSION

This paper studies theoretically and practically active learning, in the case of regression. As pointed out in the introduction, the classification case is very different and non-blind approaches look much more appealing in that case. Our conclusions hold for regression and our experiments are performed in the SDP case in which robustness of sampling is particularly important.

It is sometimes argued that randomness provides robustness (Rust, 1997). We confirm in theorem 2.3 that derandomization should not be complete for the sake of robustness (strict determinism or almost determinism lead to the loss of universal consistency), but we show in theorem 2.4 that a strong derandomization, with only a moderate random part, is enough for UC. In particular, derandomized sequences as used in theorem 2.4 combine (i) known convergence rates (Cervellera and Muselli, 2003) of quasi-random sequences for learning smooth target functions and (ii) robustness of random sampling in terms of UC as in (Devroye et al., 1994).

This conclusion about the best "quantity" of randomness is exemplified in our experiments by, in some cases, the weakness of methods too strongly focusing on parts looking promising (EAS with $\alpha < 1$). On the other hand, these methods are sometimes very efficient for difficult cases (see Table 2). Also, we note that the best blind-method is GLD, often by far, and GLD is a very natural randomized low-dispersion sequence (see (Lindemann and LaValle, 2003; LaValle and Branicky, 2002) on this topic).

We believe in the importance of research about blind or non-blind active learning as active learning *is* a main bottleneck in reinforcement learning or SDP. We have the following positive conclusions for active regression:

1. Our **new blind sampling method**, namely GLD, outperforms all other blind samplers in most cases (see table 1). In particular, it is a natural extension to any number of points and any dimensionality of regular grid-samplings (see figure 1 for small number of points, different (and better than, at least in our experiments) from the low-dispersion approach LD, GLDfff, RandomForAll, QR, etc. This blind approach is the most stable one in our experiments (first ranked in most experiments); it has weaknesses in cases in which the frontier is not relevant. It is known that quasi-random-sequence have to be randomized for various no-bias properties (see e.g. (L'Ecuyer and Lemieux, 2002) about randomized quasi-Monte-Carlo sequences); in this paper we have (i) shown theoretical similar results for active learning pointing out that **randomness is necessary in active sampling** (ii) provided with GLD **a natural randomization** as we simply randomly pick up, for each point, one of the optimal points for the greedy criterion in equation 2 (see figure 1).

2. Our **new non-blind sampling method** is very easy to use; it is **easy to put expert knowledge in it** (increasing α for more carefully exploring the domain, reducing α for focusing on optimistic parts, changing the criterion, adding diversity in the evolutionary algorithm). However, experiments show that the criterion tested here is not universal; it works very efficiently for the "multi-agent" problem, providing strategies with smaller expected cost than all blind techniques tested, but not for some other problems. This is somewhat disappointing, but we guess that non-blind sampling methods do require some tuning. EAS moves continuously, thanks to the α-parameter, from random sampling (or possibly quasi-random sampling, but this has not been studied here) to a focus on areas verifying some used-defined criterion. Results appear robust w.r.t to α, but the criterion is probably a much stronger parameter; here, it focuses on areas with small costs; but for other

problems it is probably much better to sample areas with large costs (e.g. when strong penalties can occur when hitting boundaries). A main advantage of EAS is that it works in **high dimensionalities** in which very few published papers have good active-results.

3. In some cases, GLD is far better than GLDfff, but in one case GLD is the worst algorithm and GLDfff is (very significantly) the best one. As the main difference between these two algorithms is that GLD samples more strongly the frontier, this points out the simple fact that **the frontier of the state-space** can be very important: sometimes it is very relevant (e.g. stock management, in which marginal costs have a first approximation thanks to cost-to-go at corners) and sometimes it is pointless and expensive (GLD puts 2^d points on the corners among the $2^d + 1$ first points!).

REFERENCES

Baeck, T. (1995). *Evolutionary Algorithms in theory and practice*. New-York:Oxford University Press.

Barto, A., Bradtke, S., and Singh, S. (1993). Learning to act using real-time dynamic programming. Technical Report UM-CS-1993-002.

Bertsekas, D. and Tsitsiklis, J. (1996). Neuro-dynamic programming, athena scientific.

Cervellera, C. and Muselli, M. (2003). A deterministic learning approach based on discrepancy. In *Proceedings of WIRN'03, pp53-60*.

Chapel, L. and Deffuant, G. (2006). Svm viability controller active learning. In *Kernel machines for reinforcement learning workshop, Pittsburgh, PA*.

Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1995a). Active learning with statistical models. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems*, volume 7, pages 705–712. The MIT Press.

Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1995b). Active learning with statistical models. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems*, volume 7, pages 705–712. The MIT Press.

Collobert, R. and Bengio, S. (2001). Svmtorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160.

Devroye, L., Gyorfi, L., Krzyzak, A., and Lugosi, G. (1994). the strong universal consistency of nearest neighbor regression function estimates.

Eiben, A. and Smith, J. (2003). *Introduction to Evolutionary Computing*. springer.

Gelly, S., Mary, J., and Teytaud, O. (2006). Learning for dynamic programming, proceedings of esann'2006, 7 pages, http://www.lri.fr/∼teytaud/lfordp.pdf.

Gelly, S. and Teytaud, O. (2005). Opendp, a c++ framework for stochastic dynamic programming and reinforcement learning.

Kearns, M., Mansour, Y., and Ng, A. (1999). A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *IJCAI*, pages 1324–1231.

Larranaga, P. and Lozano, J. A. (2001). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers.

LaValle, S. and Branicky, M. (2002). On the relationship between classical grid search and probabilistic roadmaps. In *Proc. Workshop on the Algorithmic Foundations of Robotics*.

L'Ecuyer, P. and Lemieux, C. (2002). Recent advances in randomized quasi-monte carlo methods.

Lewis, D. and Gale, W. (1994). Training text classifiers by uncertainty sampling. In *Proceedings of International ACM Conference on Research and Development in Information Retrieval*, pages 3–12.

Liang, F. and Wong, W. (2001). Real-parameter evolutionary sampling with applications in bayesian mixture models. *J. Amer. Statist. Assoc.*, 96:653–666.

Lindemann, S. R. and LaValle, S. M. (2003). Incremental low-discrepancy lattice methods for motion planning. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 2920–2927.

Munos, R. and Moore, A. W. (1999). Variable resolution discretization for high-accuracy solutions of optimal control problems. In *IJCAI*, pages 1348–1355.

Niederreiter, H. (1992). *Random Number Generation and Quasi-Monte Carlo Methods*.

Owen, A. (2003). *Quasi-Monte Carlo Sampling, A Chapter on QMC for a SIGGRAPH 2003 course*.

Procopiuc, O., Agarwal, P., Arge, L., and Vitter, J. (2002). Bkd-tree: A dynamic scalable kd-tree.

Rust, J. (1997). Using randomization to break the curse of dimensionality. *Econometrica*, 65(3):487–516.

Schohn, G. and Cohn, D. (2000). Less is more: Active learning with support vector machines. In Langley, P., editor, *Proceedings of the 17th International Conference on Machine Learning*, pages 839–846. Morgan Kaufmann.

Seung, H. S., Opper, M., and Sompolinsky, H. (1992). Query by committee. In *Computational Learning Theory*, pages 287–294.

Sloan, I. and Woźniakowski, H. (1998). When are quasi-Monte Carlo algorithms efficient for high dimensional integrals? *Journal of Complexity*, 14(1):1–33.

Sutton, R. and Barto, A. (1998). *Reinforcement learning: An introduction*. MIT Press., Cambridge, MA.

Thrun, S. B. (1992). Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, Pittsburgh, Pennsylvania.

Tuffin, B. (1996). On the use of low discrepancy sequences in monte carlo methods. In *Technical Report 1060, I.R.I.S.A.*

Vidyasagar, M. (1997). *A Theory of Learning and Generalization, with Applications to Neural Networks and Control Systems*. Springer-Verlag.