

Derandomized sampling for learning, with application to stochastic dynamic programming

Olivier Teytaud, Sylvain Gelly and Jérémie Mary and

IA-TAO, Lri, Bât. 490,
Université Paris-Sud, 91405 Orsay Cedex, France

Abstract. We study active learning as a derandomized form of sampling. We show that full derandomization is not suitable in a robust frameworks, 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 in the case of stochastic dynamic programming on a continuous domain. Our main results is that the derandomization works, blindness is robust, and the frontier is important.

1 Introduction to active learning

As pointed out in e.g. [CGJ95b], 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

- a sampler, that chooses points in the domain, and
- a passive learner that takes this 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:

- 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 (e.g. [CM03]); in this case, the process can be: (i) sample the points, (ii) label by the oracle, (iii) learn the target concept/function;
- 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; [LG94] chooses examples with the maximum uncertainty of the learner, [SOS92] chooses examples that reduce maximally the size of the version space. Other approaches include [SC00] (with application to Support Vector Machines (SVM)), [CGJ95a] (with application to Neural Networks).

1.1 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. Also, blind approaches are not always clearly better than pure random sampling. Quasi-random points (blind sampling) are much better than random points for various criteria in small dimensions, even if, when the dimension increases, random points can become better again (see [KMN99, Rus97] for some interesting properties of random sampling). Pessimistic results about the possibility of active learning can be found in e.g. [Vid97], in a worst-case scenario. Mainly, as a conclusion of the state of the art, the best applied results concern moderately large families of functions; classification more than regression, decision trees more than neural networks or SVM. 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.

1.2 Limitations of blind approaches

It is known that the improvement realized by deterministic well-chosen sequences in front of pure random sequences decays as the dimension increases. However, modern quasi-random points sequences are much better than the old versions, in particular thanks to scrambling (see e.g. [Owe03, LL02]), and now become reliable also in high-dimension ([SW98]). For example, some applications of QR (quasi-random) in optimization have been shown efficient in dimension 50, without problem-specific reduction to small dimensions. Best results in the traditional field of quasi-random sequences, namely integration, now come from randomized quasi-random sequences, in particular when dimension increases ([LL02]), whereas former best sequences were strictly deterministic ones. This is a somewhat surprising element in the history of quasi-random sequences. We show here that a similar superiority of randomized, yet non naive, sampling can be shown both theoretically and empirically.

1.3 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 regressions and not classifications.

Note that many works about active sampling of the environment use simulations; see e.g. the pioneer work [BBS93] and many subsequent works. We will here avoid this 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 samplings that sample as efficiently as possible the ‘full’ domain. Examples of active learning which are not simulation-based for dynamic problems have been provided in [MM99] (active discretization of the domain), [CD06] (active learning for SVM, but with a classification approach thanks to the viability approach that reformulates the problem as a classification problem).

1.4 Overview of results

We will here study the following questions, in the case of regression:

- Is non-blind active-learning better than blind active learning? Essentially, we will see some superiority of non-blind approaches on blind approaches, but the improvement is moderate, and for some problems the results are indeed much worse. This is an experimental answer only; we have not found any theoretical element for the intuitive idea of robustness of blind approaches; we have only theoretical results on some robustness for randomized techniques on too-much-deterministic methods. The theoretical analysis of the fact that active learning has some hard limitations has already been done in [Vid97]; but the fact that active non-blind learning can not be as robust as blind learning remains to-be-done.
- Are deterministic approaches better than random approaches? Essentially, we prove robustness (universal consistency) results for random samplers, that are not shared by deterministic samplers. Interestingly, we however show that the quantity of randomness can be strongly reduced, e.g. in a framework preserving both
 - improved convergence rates (as shown in [CM03]) for ”smooth-enough” target-functions;
 - universal consistency (as shown in this paper) for any target function.

2 Some mathematical elements

For any set E , we note E^* the set of finite families of elements of E .

Definition 2.1 *Consider a domain $D = [0, 1]^d$. All definitions below are relative to D .*

*A **learner** A on D is a computable mapping from $(D \times \mathbb{R})^*$ to \mathbb{R}^D .*

*A **sampler** S on D is a computable mapping from $(D \times \mathbb{R})^*$ to D .*

*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 $x_n \leftarrow S(x_0, \dots, x_{n-1}, y_0, \dots, y_{n-1}, A)$;
3. define $y_n \leftarrow f(x_n)$;
4. set $f_n = A(x_0, \dots, x_n, y_0, \dots, y_n)$.
5. go back to step 2.

A sample-learner on D is said **universally consistent (UC)** if, for any measurable f with values in $[0, 1]$, almost surely $\int_D (f(x) - f_n(x))^2 dx \rightarrow 0$ as $n \rightarrow \infty$.

We say that a sampler S on D is **universally consistent (UC)** if for at least one learning algorithm A , the sampler-learner $S + A$ is universally consistent.

A sampler is said **almost-deterministic** if each run of it only uses a finite number of random bits that only depends on the length of its inputs.

A learner is said **almost-deterministic** if each run of it only uses a finite number of random bits that only depends on the length of its inputs.

Theorem 2.2 *The random sampling is universally consistent.*

Proof: Simply use any universally consistent algorithm defined in the statistical learning literature (see e.g. [DGKL94]).

Theorem 2.3 (Moderately randomized samplings are not UC)

Consider S an almost-deterministic sampler. Then, for any almost-deterministic learner A , $S + A$ is not universally consistent. Therefore, S is not universally consistent.

Proof: Consider S and A , respectively an almost-deterministic sampler and an almost-deterministic learner. Then, consider x_0, \dots, x_n, \dots 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 almost-determinism, 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 universally consistent, then on a target $f = g_0$ almost surely f_n converges in norm L^2 to the constant function equal to 0. But f_n is distributed in the same manner as if $f = g_{\frac{1}{2}}$, and f_n should therefore converge in norm L^2 to the constant function equal to $\frac{1}{2}$. This is a contradiction; $S + A$ can not be universally consistent. \square

We have shown in theorem 2.2 that random sampling is universally consistent and in theorem 2.3 that no almost-deterministic sampling can be universally consistent. But where is the limit ? We show in the following theorem that a moderate derandomization, however using a continuous random seed, can lead to uniform 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 ([CM03]).

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

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

- randomly uniformly draw $s \in D$.
- y_0, \dots, y_n, \dots 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

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

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

- 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 $x_n \in D$).

(the sampler is not written exactly as in the definition, but this writing is equivalent)

Then, S is universally consistent.

Proof: We will show (and this is sufficient) that $S + A$ is universally consistent, where A is the 1-nearest neighbor rule for the L^∞ distance.

By lemma 2.5 proved below, we know that:

- for some $\zeta(n) = O(1/n)$ as $n \rightarrow \infty$,

$$\sup_{i \in [[1, n]]} \mu(\{x \in [0, 1]^d; NN_n(x) = x_i\}) \leq \zeta(n) \quad (1)$$

$$\text{for any measurable } E, \text{ almost surely in } s, \#\{i \in [[1, n]]; y_i \in E\}/n \rightarrow \mu(E) \quad (2)$$

where μ is the Lebesgue-measure and with $NN_n(y)$ one of the nearest neighbor of y in $\{y_1, \dots, y_n\}$ for the supremum norm ($NN_n(y)$ is not uniquely determined, what does not matter for the result here).

- for any $\delta > 0$, if $n \geq N(\delta)$, then $\sup_{x \in [0, 1]^d} \inf_{i \leq n} \|x - x_i\| \leq \delta$.

We note $T_n(h)$ the function $x \mapsto h(NN_n(x))$.

- as a corollary of the theorem of Lusin, for any f measurable from $[0, 1]^d$ to $[0, 1]$, there exists a sequence f_n of continuous function from $[0, 1]^d$ to $[0, 1]$ that converges almost everywhere to f . We note $n(\epsilon)$ an integer such that $n \geq n(\epsilon) \Rightarrow \|f_n - f\|_1 \leq \epsilon$ (such a $n(\epsilon)$ exists by Lebesgue's dominated convergence theorem).

- by the theorem of Heine, each f_n is uniformly continuous (as $[0, 1]^d$ is a compact Hausdorff space); we note $\delta_{\epsilon, n}$ such that $\|x - y\| \leq \delta_{\epsilon, n}$ ensures that $|f_n(x) - f_n(y)| \leq \epsilon$.
- if $n \geq n(\epsilon)$ and $n' \geq N(\delta_{\epsilon, n})$, then

$$\|f_n - f\|_1 \leq \epsilon \quad (3)$$

$$\|T_{n'}(f_n) - f_n\|_\infty \leq \epsilon \quad (4)$$

- combining equations 3 and 4 above yield

$$\|f - T_{n'}(f_n)\|_1 \leq 2\epsilon \quad (5)$$

- equation 3 leads to

$$\mu(\{x; |f_n(x) - f(x)| \geq A\}) \leq \epsilon/A \quad (6)$$

- equation 2 applied to $E = \{x; |f(x) - f_n(x)| \geq A\}$ and equation 6 leads to

$$\frac{1}{n'} \#\{i \in [1, n']; |f(x_i) - f_n(x_i)| \geq A\} \leq \epsilon/A + o(1) \quad (7)$$

as $n' \rightarrow \infty$, almost surely in s ; $o(\cdot)$ is for $n' \rightarrow \infty$.

- $T_{n'}(f)$ and f are equal at the x_i for $i \leq n'$, therefore equation 7 yields

$$\frac{1}{n'} \#\{i \in [1, n']; |T_{n'}(f)(x_i) - f_n(x_i)| \geq A\} \leq \epsilon/A + o(1) \quad (8)$$

as $n' \rightarrow \infty$.

- equations 8 and 1 lead to

$$\mu(\{x \in [0, 1]^d; |T_{n'}(f)(x) - T_{n'}(f_n)(x)| \geq A\}) \leq (\epsilon/A + o(1)) \underbrace{n' \zeta(n')}_{O(1)}$$

which in the case $A = \sqrt{\epsilon}$ leads to

$$\|T_{n'}(f) - T_{n'}(f_n)\|_1 \leq O(2\sqrt{\epsilon} + o(1)) \quad (9)$$

as in all this proof, $o(\cdot)$ is for $n' \rightarrow \infty$.

- Equations 5 and 9 lead to

$$\|T_{n'}(f) - f\|_1 \leq O(\epsilon) + O(\sqrt{\epsilon}) + o(1) = O(\sqrt{\epsilon})$$

almost surely in s , for any fixed ϵ . In particular, this is true almost surely for all $\epsilon = 1/2^i$ for $i \in \mathbb{N}$ (as this set is countable). This is exactly the almost sure convergence of $T_{n'}(f)$ to f for the $\|\cdot\|_1$ norm and therefore for the $\|\cdot\|_p$ norm for any $p \geq 1$. \square

We now show the following lemma about uniform low-dispersion sequences. Note that the requirement about discrepancy is much more moderate than in so-called low-discrepancy sequences (in which the discrepancy is $O(\log(n)^d/n)$ and not $o(1)$ as here).

Lemma 2.5 (Nice property of shifted well-distributed sequences)

Note μ the Lebesgue-measure. Consider y_1, \dots, y_n a sequence of elements in $[0, 1]^d$ such that the dispersion is low i.e.

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

and the discrepancy goes to 0, i.e.

$$\sup_{r \in [0, 1]^d} |\#\{i \in [[1, n]]; \forall j x_{i,j} \leq r_j\} / n - \pi_{i \in [[1, d]]} r_i| \rightarrow 0 \text{ as } n \rightarrow \infty \quad (10)$$

where $x_{i,j}$ is the j^{th} component of x_i . We consider s a random variable uniformly distributed in $[0, 1]^d$. We define for all $i \geq 0$, $x_i = y_i + s$. Then

- for some $\zeta(n) = O(1/n)$ as $n \rightarrow \infty$,

$$\forall n \in \mathbb{N}, \forall i \leq n, \mu(\{x \in [0, 1]^d; NN_n(x) = x_i\}) \leq \zeta(n) \quad (11)$$

and for any measurable E ,

$$\text{almost surely in } s, \frac{1}{n} \#\{i \in [[1, n]]; x_i \in E\} \rightarrow \mu(E) \quad (12)$$

with $NN_n(x)$ one of the nearest neighbor of x in $\{x_1, \dots, x_n\}$ ($NN_n(x)$ is not uniquely determined, but this does not matter for the result here).

- for any $\delta > 0$, then for some $N(\delta)$, if $n \geq N(\delta)$, then $\sup_{x \in [0, 1]^d} \inf_{i \leq n} \|x - x_i\| \leq \delta$.

Proof: In this proof we will call "rectangles" rectangles with sides parallel to axis, e.g. the rectangle with parameters a and b in $[0, 1]^d$ is $\{x \in [0, 1]^d; a \leq x \leq b\}$.

First, it is easy to see (and classically known in discrepancy literature) that equation 10 leads to

$$\sup_{(r,t) \in ([0, 1]^d)^2} |\#\{i \in [[1, n]]; \forall j, t_j \leq y_{i,j} \leq r_j\} / n - \pi_i(r_i - t_i)| \rightarrow 0 \text{ as } n \rightarrow \infty \quad (13)$$

where $y_{i,j}$ is the j^{th} component of y_i . This means that the measure of rectangles is well-approximated by the sequence of points, as soon as equation 10 (about rectangles including 0) holds. This is proved by writing the area of a rectangle as a weighted sum with weights -1 and 1 of areas of rectangles including 0.

The existence of $N(\delta)$ and equation 11 are immediate. Equation 12 is proved as follows.

- Any Lebesgue measurable set E is equal to a Borel set F within a set of null measure N :

$$E \setminus N = F \setminus N$$

with $\mu(N) = 0$ and F Borel-measurable.

- With probability 1, all the x_i are in $D \setminus N$ as N has null measure; therefore, almost surely, equation 12 is equivalent to its counterpart with the Borel-measurable F instead of the Lebesgue-measurable E .
- Thanks to this two points, it is sufficient to show that equation 12 holds for E Borel-measurable.
- For any $\epsilon > 0$, one can write

$$\cup_{i \in I} R_i \subset F \subset D \setminus \cup_{i \in J} R'_i$$

where both I and J are finite and

$$\mu(D \setminus \cup_{i \in J} R'_i \setminus \cup_{i \in I} R_i) \leq \epsilon$$

and the R_i and the R'_i are disjoint rectangles. Therefore, for n sufficiently large, equation 13 ensures that within precision ϵ , $\frac{1}{n} \#\{i \in [[1, n]]; x_i \in E\}$ is equal to $\mu(E)$ within precision 2ϵ .

Hence the expected result. \square

3 Application to stochastic dynamic programming with a new sampling method

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 Function value learning and active learning

We introduce reinforcement learning and stochastic dynamic programming in a very short manner ; see [BT96, SB98] 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 to the expectancy of the cost conditionally to this state. This cost-to-go has to be learnt from examples. These examples can be chosen, and a sometimes expensive oracle provides noise-free values for a given example (this oracle for time step t is : simulate each possible random outcome, for each random outcome simulate the instantaneous cost and use the estimated cost-to-go at time step $t+1$ for estimating the future cost, add these two costs for each random outcome,

average all these results to get the required value). Roughly, the process is as follows :

For each time step t , backwards from the last time step to the first :

- choose a finite set of examples x_1, \dots, x_n at time step t ;
- 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 s) ;
- learn the expected cost-to-go at time step t from these examples.

The choice of the finite set of x -values is important in order to reduce the computational cost, which is the main trouble in stochastic dynamic programming.

We will use approaches and theoretical analysis that are not specific of stochastic dynamic programming. More specific approaches for dynamic problems are the followings:

- [BBS93] has shown the importance of reducing the domain, when possible, by using simulations; if you are interested in a small subset of the full state space, you can use it in your dynamic-programming algorithm, e.g. by restricting your attention to the neighborhood of the visited states (simulate, see which states are visited, restrict your attention to these 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.
- [Thr92] studies 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. In the case of stochastic simulation, see also [Gly89] ; this can be related to the derandomization of random process sampling ([Hic98]): in both cases, the idea is "do not let randomness choose the exploration".

3.2 Some point sets for blind active-learning

We present below point sets in $D = [0, 1]^d$. P will be the notation for a set of points P_1, \dots, P_n . $\#E$ is the cardinal of a set E . See e.g. [Tuf96, Owe03, LL02] for a general introduction to "well chosen" sets of points.

3.2.1 Low discrepancy

The most usual discrepancy is defined as follows :

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

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 points achieve $O(\log(\#P)^d/\#P)$. For large values of d the exponent becomes a trouble. Many solutions have been proposed for dealing with large dimensionality if the distribution to be approximated has strong dependencies between coordinates (see e.g. [Hic98]) or when a reasonable grouping is possible ([Owe03]), but for $[0, 1]^d$ the only results are (see [WW97, SW98] and references therein) :

- a deterministic sequence for which it is proved that the exponential dependency in d is removed, but the discrepancy decreases as $O(1/(\#P)^\alpha)$ with $\alpha < \frac{1}{2}$, i.e. the sequence is worst than a random independent sequence.
- an existence proof, without any construction, of a sequence that achieves $O(1/(\#P)^\alpha)$ with $\alpha > \frac{1}{2}$, i.e. there exists something that (i) is better than random (ii) is better than standard low-discrepancy in large dimension, but we are not able to build it.

The interested reader is referred to [Nie92, Tuf96, Owe03] for more information. Other discrepancies have been defined, in particular using a L^p norm of $area([0, r]) - \frac{1}{n} \#\{i \in [[1, n]]; P_i \leq r\}$ instead of a L^∞ norm, or using other shapes than rectangles $[0, r]$.

The main methods for the generation of low-discrepancy sequences are algebraic. Numerical optimization of $Disc(.)$ has been tried but is very difficult. We will in the sequel call "low-discrepancy sequence" a classical Niederreiter sequence (see e.g. [Nie92]). We will refer to this sequence as a QR (quasi-random) sequence in the sequel.

3.2.2 Low dispersion

Low dispersion is less widely used than low-discrepancy, but has some advantages (see e.g. discussions in [Tuf96]). The most usual criterion (to be minimized) is

$$Dispersion(P) = \sup_{x \in D} \inf_{p \in P} d(x, p) \quad (14)$$

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) :

$$Dispersion_2(P) = \inf_{(x_1, x_2) \in D^2} d(x_1, x_2) \quad (15)$$

A main difference is that optimizing eq. 15 "pushes" points on the frontier. This effect can be avoided as follows :

$$Dispersion_3(P) = \inf_{(x_1, x_2) \in D^2} d(x_1, \{x_2\} \cup D') \quad (16)$$

where $D' = \{x \in \mathbb{R}^d; x \notin D\}$.

We call "low-dispersion point set" (LD) a point set optimizing equation 15. We call "greedy-low-dispersion point set" (GLD) a point set optimizing equation 15 in a greedy manner, i.e. $P_1 = (0.5, \dots, 0.5)$ 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 $Dispersion_2(\{P_1, \dots, P_{n-1}, P_n\})$ 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 [PAAV02]). We call "greedy-low-dispersion point set far-from-frontier" (GLD-fff) 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.

3.3 A non-blind active-learning algorithm

Evolutionary algorithms are a classical tool for robustly optimizing a non-linear function without requiring derivative. We here use them as a sampling algorithm, as follows:

- generate an initial population uniformly on the domain;
- evolve the population until the allowed number of fitness evaluations;
- use as active sample the union of all offspring of all epochs.

Note that the learning algorithm is not embedded in this approach, which is therefore quite general.

We define precisely below our evolutionary algorithm, but we point out that any evolutionary algorithm could be used as well. In particular, 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. The reader is referred to [Bae95, ES03] for more information on evolutionary algorithms; estimation of distribution algorithms could be used as well [LL01].

The genetic algorithm 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 $\frac{\sigma}{\sqrt[n]{n}}$ 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 :

- we copy the $n\lambda_1$ best individuals (set S_1).
- we combine the $n\lambda_2$ following best individuals with the individuals of S_1 (rotating among S_1 if $\lambda_1 < \lambda_2$).
- we mutate $n\lambda_3$ individuals among S_1 (again rotating among S_1 if $\lambda_1 < \lambda_3$).
- 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 chosen for the work presented in this paper.

The population size is $n = N^\alpha$, where α is a parameter of the approach. α is typically equal to 0.8.

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. This is the case in stochastic dynamic programming only if we are dealing with a problem in which it is likely that trajectories are close to good values. 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;
- α smaller leads to a more optimistic approach in which the sample is reinforced in parts for which rewards are better, at the price of a weaker behavior for avoiding very negative rewards.

3.4 Experiments

The samplers have to sample a product $D \times S$, with S continuous and $D = \{D_1, \dots, D_k\}$ discrete and finite. The following sampling methods are used:

- the sampling methods GLD, QR, LD, GLDfff are the blind approaches defined in section 3.2 for sampling the state domain - the states of the discrete Markov-processes are sampled in a simple proportional manner. Precisely, for sampling n points in the product $S \times D$, where S is the continuous state space and D is a discrete domain $D = \{D_1, \dots, D_k\}$, a GLD, QR, LD or GLDfff point set is used for sampling S with $N = n/k$ points x_1, \dots, x_N , and the sample is

$$\begin{aligned} & (D_1, x_1), (D_1, x_2), \dots, (D_1, x_N), \\ & (D_2, x_1), (D_2, x_2), \dots, (D_2, x_N), \\ & \dots, \dots, \dots, \\ & (D_k, x_1), (D_k, x_2), \dots, (D_k, x_N). \end{aligned}$$

- the sampling method RandomForAll denotes the pure blind (uniform, independent) random sampling of $D \times S$.
- the sampling method RandomOnlyForTrueStates denotes the pure blind random sampling of S , with a proportional sampling of D (i.e. D_1 appears the same number of times as D_2 , as D_3 , ..., as D_k).
- the sampling method DeterministicOnlyForTrueStates is equal to RandomOnlyForTrueStates, except that for each value D_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.

- `SamplingMethodOptimization` denotes the non-blind approach using evolutionary sampling, as explained in section 3.3. The parameter is the α -parameter in section 3.3; values 0.5, 0.6, 0.7, 0.8, 0.9 have been tested.

The parameters of the experiments are as follows:

- 70 points used for each non-linear optimization computing one value function; the optimizations are performed by the same evolutionary algorithm as the one used for the sampling but with population size $\lfloor \sqrt{70} \rfloor$;
- 5000 points sampled on the domain (same number of sampled points for each algorithm);
- all results have been averaged on 11 runs.
- the learnings are performed by the "automaticRegression" method from [GT05], that uses the Gaussian-SVM of SVMTorch ([CB01]) and a cross-validation-based choice of parameters.

All the experiments have been performed with the OpenDP library ([GT05]) with the command line options `./runme.sh -nojava -nogui -test OptimizationExperiments -outputFile myResultsFile.txt -testedOptimizers '[[GeneticAlgorithmOptimizer]]' -testedRegressions '[[AutomaticRegression]]' -numExampleWanted X -nbPointsSamplingMethod 5000 -nbRuns 11 -nbTryEvaluationsOptimization 70` where X is the number of the problem.

For comparison, we include the results of the "stupid" method that randomly generates actions uniformly in the action-domain and the "greedy" method that only optimizes the immediate rewards. The objective functions are to be minimized (lower values = better results).

All problems are described in [GMT06] and in [GT05] (including free downloads on <http://opendp.sourceforge.net>). The dimensionality of problems (the dimension of the continuous state space, excluding the discrete parts) is between parenthesis.

3.4.1 Results *DynamicSystemAway* (2)

SamplingMethodOptimization0.8	1.00 ± 1.22
SamplingMethodDeterministicOnlyForTrueStates	1.04 ± 1.25
SamplingMethodQR	1.04 ± 1.23
SamplingMethodGLD	1.09 ± 1.31
SamplingMethodRandomForAll	1.09 ± 1.30
SamplingMethodRandomOnlyForTrueStates	1.09 ± 1.30
SamplingMethodOptimization0.9	1.13 ± 1.36
SamplingMethodLD	1.18 ± 1.36
SamplingMethodOptimization0.5	1.18 ± 1.45
SamplingMethodOptimization0.6	1.22 ± 1.45
SamplingMethodOptimization0.7	1.22 ± 1.55
SamplingMethodGLDff	1.59 ± 2.3002
Stupid	10 ± 1.41
Greedy	11.5 ± 3.53

3.4.2 Results *DynamicSystemMultiagent* (8)

SamplingMethodGLDff	192.72 ± 14.20
SamplingMethodOptimization0.8	197.27 ± 17.93
SamplingMethodOptimization0.9	201.81 ± 11.67
SamplingMethodDeterministicOnlyForTrueStates	204.54 ± 8.20
SamplingMethodOptimization0.6	204.54 ± 15.07
SamplingMethodRandomOnlyForTrueStates	205.45 ± 17.52
SamplingMethodLD	208.18 ± 14.01
SamplingMethodQR	208.18 ± 17.78
SamplingMethodOptimization0.7	209.09 ± 13.75
SamplingMethodRandomForAll	210.00 ± 15.49
SamplingMethodGLD	211.81 ± 18.87
SamplingMethodOptimization0.5	220 ± 17.88
Greedy	490 ± 0.00
Stupid	610 ± 0.00

3.4.3 Results *DynamicSystemArm* (3)

SamplingMethodRandomOnlyForTrueStates	9.23 ± 0.832
SamplingMethodRandomForAll	9.58 ± 1.09
SamplingMethodLD	10.73 ± 1.74
SamplingMethodDeterministicOnlyForTrueStates	10.81 ± 1.78
SamplingMethodQR	10.82 ± 1.7
SamplingMethodGLD	11.60 ± 1.27
SamplingMethodOptimization0.8	11.71 ± 1.07
SamplingMethodGLDfff	12.12 ± 1.35
Greedy	12.38 ± 8.45
SamplingMethodOptimization0.9	12.43 ± 1.21
SamplingMethodOptimization0.7	12.48 ± 1.20
SamplingMethodOptimization0.6	12.56 ± 1.68
SamplingMethodOptimization0.5	13.27 ± 0.79
Stupid	19.06 ± 4.19

3.4.4 Results *DynamicSystemShoot* (3)

SamplingMethodGLD	28.52 ± 0.159
SamplingMethodRandomOnlyForTrueStates	28.54 ± 0.19
SamplingMethodRandomForAll	28.54 ± 0.15
SamplingMethodOptimization0.8	28.55 ± 0.18
SamplingMethodDeterministicOnlyForTrueStates	28.57 ± 0.18
SamplingMethodOptimization0.9	28.59 ± 0.21
SamplingMethodQR	28.60 ± 0.19
SamplingMethodOptimization0.6	28.62 ± 0.23
SamplingMethodGLDfff	28.63 ± 0.23
SamplingMethodLD	28.66 ± 0.20
SamplingMethodOptimization0.7	28.66 ± 0.24
SamplingMethodOptimization0.5	28.68 ± 0.23
Greedy	29.00 ± 0.00
Stupid	29.00 ± 0.00

3.4.5 Results *DynamicSystemStockAndDemand* (4)

SamplingMethodGLD	1841.39 ± 14.54
Stupid	2507.41 ± 1002.02
SamplingMethodRandomOnlyForTrueStates	2777.5 ± 124
SamplingMethodRandomForAll	2902.12 ± 122.52
SamplingMethodQR	2917.52 ± 166.91
SamplingMethodDeterministicOnlyForTrueStates	2939.09 ± 167.11
SamplingMethodLD	2984.68 ± 90.24
Greedy	3000.56 ± 753.02
SamplingMethodGLDfff	3022.52 ± 17.10
SamplingMethodOptimization0.6	3022.52 ± 17.10
SamplingMethodOptimization0.7	3022.52 ± 17.10
SamplingMethodOptimization0.8	3022.52 ± 17.10
SamplingMethodOptimization0.9	3022.52 ± 17.10
SamplingMethodOptimization0.5	3022.52 ± 17.10

3.4.6 Results *DynamicSystemTrivial* (2)

SamplingMethodGLD	550 ± 1e-15
SamplingMethodOptimization0.7	550 ± 1e-15
SamplingMethodOptimization0.8	550 ± 1e-15
SamplingMethodOptimization0.9	550 ± 1e-15
SamplingMethodOptimization0.5	550 ± 1e-15
SamplingMethodDeterministicOnlyForTrueStates	559.09 ± 30.15
SamplingMethodRandomForAll	559.09 ± 20.22
SamplingMethodQR	563.63 ± 23.35
SamplingMethodLD	568.18 ± 33.71
SamplingMethodRandomOnlyForTrueStates	581.81 ± 51.34
Greedy	1000.00 ± 0.00
SamplingMethodGLDfff	1000.00 ± 0.00
SamplingMethodOptimization0.6	1000.00 ± 0.00
Stupid	1200.00 ± 0.00

3.4.7 Results *DynamicSystemWalls (2)*

SamplingMethodGLD	65 ± 1e-15
SamplingMethodQR	72.72 ± 13.66
SamplingMethodLD	81.81 ± 17.50
SamplingMethodRandomForAll	85 ± 17.32
SamplingMethodDeterministicOnlyForTrueStates	90.00 ± 14.66
SamplingMethodRandomOnlyForTrueStates	92.72 ± 12.72
Greedy	100.00 ± 0.00
SamplingMethodGLDff	100.00 ± 1e-15
SamplingMethodOptimization0.6	100.00 ± 0.00
SamplingMethodOptimization0.7	100.00 ± 0.00
SamplingMethodOptimization0.8	100.00 ± 0.00
SamplingMethodOptimization0.9	100.00 ± 0.00
SamplingMethodOptimization0.5	100.00 ± 0.00
Stupid	100.00 ± 0.00

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 stochastic dynamic case in which robustness is particularly important).

The theoretical part concludes that in the case of active regression:

- derandomization should be moderate for the sake of robustness (strict determinism or almost determinism lead to the loss of universal consistency);
- but moderate derandomization is possible (results like those in [CM03] concluding to faster convergence rates for smooth target functions are preserved in our framework, i.e. with a simple random shift of a deterministic well distributed sequence, and thanks to the random part universal consistency is preserved). We conjecture that more sophisticated randomized quasi-random samples (see e.g. [LL02]) also have this property.

This suggests the use of derandomized point sets, but with moderate derandomization. This conclusion is exemplified in our experiments by the weakness in some cases of methods too strongly focusing on parts that look promising ($\alpha < 1$), whereas these methods are sometimes very efficient. We note that even values of α that look somewhat careful ($\alpha = 0.9$) sometimes lead to (intuitively unexpected a priori) disasters. In that sense, theory is in accordance with practice.

More precisely, our experiments lead to the following results:

- for derandomized technics (QR, GLDff, GLD): TODO
- for our new evolutionary technic for sampling: TODO

One can say that our conclusions are not very surprising. However, we believe in the importance of such a research as (i) active learning *is* a main bottleneck in reinforcement learning or stochastic dynamic programming and (ii) we have the following conclusions for active regression:

- our new non-blind sampling method is very easy to use, we see clearly how to put expert knowledge in it (increasing α for more carefully exploring the domain, reducing α for focusing on optimistic parts); also, it moves continuously from random sampling (or possibly quasi-random sampling) to a focus on areas verifying some criteria chosen by the user (in our experiments we focus on areas with good rewards but other criteria, such as areas with bad reward for e.g. problems in which strong penalties can occur when hitting boundaries could also be used); TODO
- we emphasize both theoretically (by theorems in section 2) and practically, with a much wider range of problems than many published works about dynamic optimization, that too strongly derandomizing the sampling necessarily leads to drawbacks in some cases. However, moderate derandomization actually works: QR-sampling is better than random sampling in all cases, and our new technique, when not too optimistically parametrized, is efficient TODO.

To conclude in four sentences:

- We propose a new technique that is easy-to-use and has intuitive parameters that moves continuously from random uniform sampling to a very deterministic sampling.
- "Strongly" active sampling is sometimes very efficient, but can lead to very poor results.
- Derandomized blind point sets are less impressive than strongly active methods in some cases, but they are much more robust and never disappointing in front of pure random sampling.
- Think about the frontier; it is not very expensive to sample it, and it can strongly enhance the results in some cases. This is unfortunately very problem dependent.

References

- [Bae95] T. Baeck. *Evolutionary Algorithms in theory and practice*. New-York:Oxford University Press, 1995.
- [BBS93] A.G. Barto, S.J. Bradtke, and S.P. Singh. Learning to act using real-time dynamic programming. Technical Report UM-CS-1993-002, , 1993.
- [BT96] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-dynamic programming*, athena scientific. 1996.
- [CB01] R. Collobert and S. Bengio. Svmtorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

- [CD06] Laetitia Chapel and Guillaume Deffuant. Svm viability controller active learning. In *Kernel machines for reinforcement learning workshop, Pittsburgh, PA*, 2006.
- [CGJ95a] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 705–712. The MIT Press, 1995.
- [CGJ95b] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 705–712. The MIT Press, 1995.
- [CM03] Cristiano Cervellera and Marco Muselli. A deterministic learning approach based on discrepancy. In *Proceedings of WIRN'03, pp53-60*, 2003.
- [DGKL94] L. Devroye, L. Györfi, A. Krzyżak, and G. Lugosi. the strong universal consistency of nearest neighbor regression function estimates, 1994.
- [ES03] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. springer, 2003.
- [Gly89] P.W. Glynn. Importance sampling for stochastic simulations. *Management Science*, 35(11):1367–92, 1989.
- [GMT06] S. Gelly, J. Mary, and O. Teytaud. Learning for dynamic programming, proceedings of esann'2006, 7 pages, <http://www.lri.fr/~teytaud/lfordp.pdf>. 2006.
- [GT05] S. Gelly and O. Teytaud. Opendp, a c++ framework for stochastic dynamic programming and reinforcement learning, 2005.
- [Hic98] Fred J. Hickernell. A generalized discrepancy and quadrature error bound. *Mathematics of Computation*, 67(221):299–322, 1998.
- [KMN99] M.J. Kearns, Y. Mansour, and A.Y. Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *IJCAI*, pages 1324–1231, 1999.
- [LG94] D. Lewis and W. Gale. Training text classifiers by uncertainty sampling. In *Proceedings of International ACM Conference on Research and Development in Information Retrieval*, pages 3–12, 1994.
- [LL01] P. Larranaga and J. A. Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2001.
- [LL02] P. L'Ecuyer and C. Lemieux. Recent advances in randomized quasi-monte carlo methods, 2002.
- [MM99] Remi Munos and Andrew W. Moore. Variable resolution discretization for high-accuracy solutions of optimal control problems. In *IJCAI*, pages 1348–1355, 1999.
- [Nie92] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. 1992.
- [Owe03] A.B. Owen. *Quasi-Monte Carlo Sampling, A Chapter on QMC for a SIGGRAPH 2003 course*. 2003.
- [PAAV02] O. Procopiuc, P. Agarwal, L. Arge, and J. Vitter. Bkd-tree: A dynamic scalable kd-tree, 2002.
- [Rus97] J. Rust. Using randomization to break the curse of dimensionality. *Econometrica*, 65(3):487–516, 1997.
- [SB98] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. MIT Press., Cambridge, MA, 1998.
- [SC00] Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In Pat Langley, editor, *Proceedings of the 17th International Conference on Machine Learning*, pages 839–846. Morgan Kaufmann, 2000.

- [SOS92] H. S. Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Computational Learning Theory*, pages 287–294, 1992.
- [SW98] I.H. Sloan and H. Woźniakowski. When are quasi-Monte Carlo algorithms efficient for high dimensional integrals? *Journal of Complexity*, 14(1):1–33, 1998.
- [Thr92] S. B. Thrun. Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, Pittsburgh, Pennsylvania, 1992.
- [Tuf96] B. Tuffin. On the use of low discrepancy sequences in monte carlo methods. In *Technical Report 1060, I.R.I.S.A.*, 1996.
- [Vid97] M. Vidyasagar. *A Theory of Learning and Generalization, with Applications to Neural Networks and Control Systems*. Springer-Verlag, 1997.
- [WW97] G. Wasilkowski and H. Wozniakowski. The exponent of discrepancy is at most 1.4778. *Math. Comp*, 66:1125–1132, 1997.