Quasi-Random resamplings, with applications to rule-samplng, cross-validation and (su-)bagging

Olivier Teytaud, Sylvain Gelly, Stéphane Lallich, Elie Prudhomme

Quasi-random resamplings, with applications to rule extraction, cross-validation and (su-)bagging

Olivier Teytaud, Sylvain Gelly, Stphane Lallich, Elie Prudhomme

May 17, 2006

1 Introduction

Resampling (typically, but not necessarily, bootstrapping) is a well-known stochastic technique for improving estimates in particular for small samples. It is known very efficient in many cases. Its drawback is that resampling leads to a compromise computational cost / stability through the number of resamplings. The computational cost is due to the study of multiple randomly drawn resamples. Intuitively, we want some more properly distributed resamples to improve the stability of resampling-based algorithms. Quasi-random numbers are a well-known technique for improving the convergence rate of data-based estimates. We here consider quasi-random version of resamplings. We apply this technique to BSFD, a data-mining algorithm for simultaneous-hypothesis-testing, to cross-validation, and to (su-)bagging, an ensemble method for learning. We present quasi-random numbers in section 2. We present bootstrap and a quasi-random version of bootstrap-sampling in section 3. We present experimental results in section 4.

2 Introduction to quasi-random numbers

If you are unlucky, random points can be distributed in a very non-uniform manner. Figure 1 (upper-left) is a pseudo-random independent sample (uniformly drawn in $[0,1]^2$). Usually, you can not get by chance something as regular as other plots in figure 1. Therefore, in many areas of computer science, better-than-random points have been studied (integration [1], optimization [2], path planning [3], learning [4]). In order to generate and check uniform point-sets, a measure of uniformity is useful. Consider a point set $x_1, \ldots, x_n$ in $D = [0,1]^d$. An intuitive measure is $\sup_{x \in D} \inf_{i \in [[1,n]]} d(x, x_i)$ (to be minimized), where d is some distance (e.g. L^∞ distance). Then, generating points as the lattice in figure 1 is easy, and it has been pointed out that for the criterion above, this is optimal for many values of the number of points ([3]). However, this point set is not satisfactory. For example, it would be nice that the projection on any axis of a good point set is a good point set. This is not the case for the

lattice in figure 1: the projection on some well-chosen axis leads to accumulations. Therefore, other criterions have been defined; the most well known is discrepancy. Among various discrepancies, the most well known is the following: $\sup_{r\in D}\left|\frac{1}{n}Card\{i\in[[1,n]];\forall j(x_i)_j\leq r_j\}-\pi_{j\in[[1,d]]}r_j\right|$. This formula has an immediate interpretation: it is the largest absolute difference between the area of a rectangle including 0 and the proportion of points in this rectangle. It is much more stable with respect to projection on an axis. However, it has various drawbacks (see [5]): (1) it only deals with rectangles with axis parallel to the canonical axis; (2) it only deals with rectangles; (3) it is not symmetric in the sense that the discrepancy of $x_1,\ldots,x_n$ is not the discrepancy of $1-x_1,\ldots,1-x_n$; (4) it is a worst case on r. The two first drawbacks are unclear drawbacks. Considering variables, in a non-rotation invariant manner, can be meaningful. The fourth drawback is probably the main trouble. Fortunately, extensions have already been defined. The main tool is the L^2-star-discrepancy: $\sqrt{\int_{r\in D}\left(\frac{1}{n}Card\{i\in[[1,n]];\forall j(x_i)_j\leq r_j\}-\pi_{j\in[[1,d]]}r_j\right)^2}$. This form of discrepancy (as well as others) verify inequalities similar to Koksma's inequality (see [5] on this topic). Many algebraic methods have been defined for generating sequences of points with low discrepancy ([1, 3, 6, 7, 8]). We here consider scrambled-Halton-sequences. We now define this quasi-random sequence. Consider p a prime number. The following sequence generates the n^{th} element $x_{n,p}\in[0,1]$ of the Van Der Corput sequence [9] in basis p:

- write n in basis p: $n=p_kp_{k-1}\ldots p_1$, i.e. $n=\sum_{i=1}^k p_ip^i$ with $p_i\in[[0,p-1]]$;
- $x_{n,p}=0.p_1p_2\ldots p_k$ in basis p, i.e. $x_{n,p}=\sum_{i=1}^k p_i/p^i$.

A classical improvement, termed *scrambling* consists in replacing this by $x_{n,p}=0.\pi(p_1)\pi(p_2)\ldots\pi(p_k)$ where π is some permutation of $0,1,\ldots,p-1$. The Halton sequence [10] generalizes the Van Der Corput sequence to dimension d. Consider p_i the i^{th} prime number. Then, x_n, the n^{th} element of a Halton sequence in dimension d, is $x_n=(x_{n,p_1},x_{n,p_2},\ldots,x_{n,p_d})\in[0,1]^d$. The scrambled-Halton sequence is the use of a randomly drawn permutation for each $i\in[[1,d]]$.

3 The applications of bootstrap in data mining and quasi-random bootstrap

Bootstrap has been first defined by Efron [11] and uniform versions appeared with the work of Giné and Zinn [12]. These works and many others are summarized in [13].

Often, we would like to know P, an unknown law of probability, but we only have a database which can be assumed i.i.d according to P. If this database is $x_1,\ldots,x_n$, the traditional notation is $\hat{P}$ for the empirical measure, i.e. $\widehat{P}=\frac{1}{n}\sum_{i=1}^n\delta_{x_i}$, where δ_x is the Dirac measure located at x. A bootstrap sample consists in randomly drawing n elements, with replacement, among the x_i: for

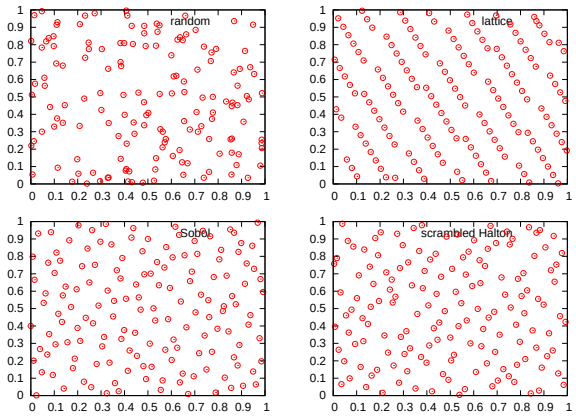


Figure 1: Random points, lattice points, Sobol points, scrambled-Halton points.

$i \in [[1,n]]$, x'_i is equal to x_{j_i}, where the $(j_i)_{i\in[[1,n]]}$ are independent and uniform on $[[1,n]]$. We then note $\widehat{\widehat{P}} = \frac{1}{n}\sum_{i=1}^{n} \delta_{x'_i}$.

So, bootstrap is based on the replacement of a random variable that can not be simulated ($\widehat{P}$, the random part of which being the sampling) by one that can be simulated ($\widehat{\widehat{P}}$, the random part of which being the resampling). Simulating $\widehat{\widehat{P}}$ is usually done in an i.i.d manner. However, there is a straightforward application of quasi-random numbers to this simulation. Precisely, we have to quasi-randomly draw the $(x'_1, \ldots, x'_n)$. Each x'_i is equal to x_{j_i} where the $j_1, \ldots, j_n$ are i.i.d uniformly in $[[1,n]]$. Naively sampling the bootstrap distribution is therefore equivalent to:

- randomly draw $z \in]0,1]^n$;
- for $i \in [[1,n]]$, define $x'_i = x_{j_i}$ where $j_i = \lceil n \times z_i \rceil$.

With this formalism, a quasi-random version is straightforward: we only have to quasi-randomly draw the z in dimension n. Therefore, we can use a quasi-random sequence in dimension n, where n is the number of instances. Unfortunately, this naive approach does not work. The reason is that strongly redundant results can appear, as the result of the second line above is the same when ordinates of z are permuted. Well distributed z's do not lead to well distributed samples. Therefore, we define a better approach. The indices j_i are multinomially distributed. It is known that the multinomial distribution is still multinomial after the first components are observed. In algorithmic terms, this means that the following procedure works for generating correct bootstrap samples:

- randomly draw $z \in]0,1]^n$;
- set $r = n$ (number of elements to be drawn in the database);
- for $i \in [[1,n]]$,

- define z'_i minimal such that $P(binomial(r, 1/(n+1-i)) \leq z'_i) \geq z_i$;
- set $r \leftarrow r - z'_i$.

- for $i \in [[1, n]]$, define the bootstrap sample such that x_i appears in the x'_j exactly z'_i times.

This generates independent bootstrap samples. We now just have to replace the random-drawing at the first line by a quasi-random sequence. Traditionally, the efficiency of quasi-random sequences is justified by Koksma-Hlawka's inequality. This inequality ensures a convergence in $O(V \log(b)^n/b)$, where n is the dimension, b is the number of quasi-random samples, and V is the total variation in the sense of Hardy & Krause. In many cases, the quasi-random method indeed works, whenever V is infinite. Here, we are in a nice case: V is finite. Let's formalize this claim:

Theorem: finiteness of the Hardy&Krause total variation *Consider f any mapping $[[0, n]]^n \to \mathbb{R}$. Consider the following application $\pi_f : z \mapsto f(z')$, $\pi_f : [0, 1]^n \to \mathbb{R}$, where z' is defined as follows:*

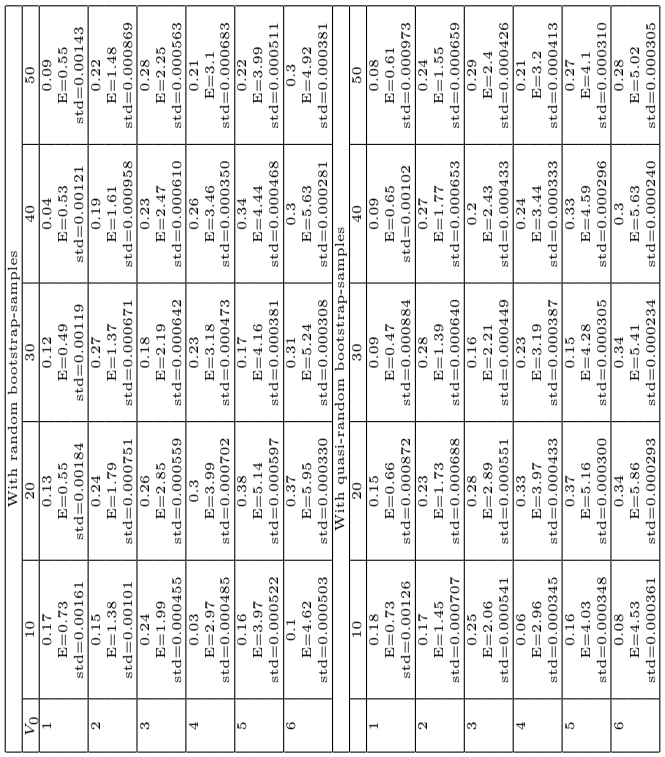
- *set $r = n$ (number of elements to be drawn in the database);*
- *for $i \in [[1, n]]$,*
  - *define z'_i minimal such that $P(binomial(r, 1/(n+1-i)) \leq z'_i) \geq z_i$;*
  - *set $r \leftarrow r - z'_i$.*

Then, π_f has finite total variation in the sense of Hardy&Krause. **Proof:** π_f is constant in each of finitely many hyperrectangles partitionning $[0, 1]^n$. This is sufficient to imply the finiteness of the total variation.□

Quasi-random points are often much more efficient than random points in low dimension, but quasi-random points must be adapted for huge dimension, in spite of strong improvements due in particular to scrambling. We need a *dimension reduction*. We perform this dimension reduction as follows for cases in which the dimensionality of the quasi-random sequence would be too high:

- group examples in the learning set in k clusters by k-means (including the label as supplementary ordinate);
- by quasi-random, choose the number of random draws in each of the k clusters, n_1 examples in cluster 1, ..., n_k examples in cluster k;
- then, randomly distribute the n_i examples among the examples in cluster i.

This method is inspired by [14] in a different context. It's likely that more sophisticated methods could be defined also. See [15] for particular cases of derandomization in huge dimension that might be helpful.



With random bootstrap-samples										
V_0	10		20		30		40		50	
1	0.17	E=0.73 std=0.00161	0.13	E=0.55 std=0.00184	0.12	E=0.49 std=0.00119	0.04	E=0.53 std=0.00121	0.09	E=0.55 std=0.00143
2	0.15	E=1.38 std=0.00101	0.24	E=1.79 std=0.000751	0.27	E=1.37 std=0.000671	0.19	E=1.61 std=0.000958	0.22	E=1.48 std=0.000869
3	0.24	E=1.99 std=0.000455	0.26	E=2.85 std=0.000559	0.18	E=2.19 std=0.000642	0.23	E=2.47 std=0.000610	0.28	E=2.25 std=0.000563
4	0.03	E=2.97 std=0.000485	0.3	E=3.99 std=0.000702	0.23	E=3.18 std=0.000473	0.26	E=3.46 std=0.000350	0.21	E=3.1 std=0.000683
5	0.16	E=3.97 std=0.000522	0.38	E=5.14 std=0.000597	0.17	E=4.16 std=0.000381	0.34	E=4.44 std=0.000468	0.22	E=3.99 std=0.000511
6	0.1	E=4.62 std=0.000503	0.37	E=5.95 std=0.000330	0.31	E=5.24 std=0.000308	0.3	E=5.63 std=0.000281	0.3	E=4.92 std=0.000381
With quasi-random bootstrap-samples										
V_0	10		20		30		40		50	
1	0.18	E=0.73 std=0.00126	0.15	E=0.66 std=0.000872	0.09	E=0.47 std=0.000884	0.09	E=0.65 std=0.00102	0.08	E=0.61 std=0.000973
2	0.17	E=1.45 std=0.000707	0.23	E=1.73 std=0.000688	0.28	E=1.39 std=0.000640	0.27	E=1.77 std=0.000653	0.24	E=1.55 std=0.000659
3	0.25	E=2.06 std=0.000541	0.28	E=2.89 std=0.000551	0.16	E=2.21 std=0.000449	0.2	E=2.43 std=0.000433	0.29	E=2.4 std=0.000426
4	0.06	E=2.96 std=0.000345	0.33	E=3.97 std=0.000433	0.23	E=3.19 std=0.000387	0.24	E=3.44 std=0.000333	0.21	E=3.2 std=0.000413
5	0.16	E=4.03 std=0.000348	0.37	E=5.16 std=0.000300	0.15	E=4.28 std=0.000305	0.33	E=4.59 std=0.000296	0.27	E=4.1 std=0.000310
6	0.08	E=4.53 std=0.000361	0.34	E=5.86 std=0.000293	0.34	E=5.41 std=0.000234	0.3	E=5.63 std=0.000240	0.28	E=5.02 std=0.000305

Figure 2: Results of quasi-random-BSFD versus standard-BSFD. The variance is much lower in all but one case.

4 Experimental results

More detailed results are available in `http://www.lri.fr/~teytaud/qrboutput.pdf`. One can find there results about quasi-random-cross-validation also.

4.1 Results for information extraction

We refer to the url above for full results; we here only present in Figure 2 one of the tables of results, in the following setting: each variable is independent, normally distributed with variance one and mean 0. BSFD is applied for extracting variables with mean > 0; therefore the best answer is the empty answer (no detection). The table presents for each dimensionality and for 16 examples the empirical probability of more than V_0 false detections, when the bootstrap-based evaluation (thanks to BSFD [16]) of the selection threshold is based on 32 bootstrap-samples. The user provides $\delta = 0.25$ and V_0; the goal of BSFD is to be as close as possible to a risk δ of selecting more than V_0 type-I errors.

4.2 Results for (su-)bagging with dimension-reduction

Bagging and subagging were applied in a setting detailed in `http://www.lri.fr/~teytaud/qrboutput.pdf` (18 examples, 25 bootstraps). A few results are presented in figure 3. The complete results in the url above show that QRBagging > Bagging > Weak learner, and QRSuBagging > SuBagging > Weak learner.

5 Conclusion

We experimented quasi-random-sequences of resamples. The standard quasi-random points in $[[0,1]]^k$ could be used thanks to the decomposition of the multinomial law into k binomial laws. The convergence is faster and more stable, both theoretically and in various applications. As far as we know, it's the first time that quasi-random-sequences of bootstrap-samples and quasi-random-sequences of subsets are defined and applied.

References

[1] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, P. SIAM, Ed., 1992.

[2] A. Auger, M. Jebalia, and O. Teytaud, "Xse: quasi-random mutations for evolution strategies," in *Evolutionary Algorithms*, 2005.

[3] B. Tuffin, "On the use of low discrepancy sequences in monte carlo methods," 1996. [Online]. Available: citeseer.ist.psu.edu/tuffin96use.html

[4] C. Cervellera and M. Muselli, "A deterministic learning approach based on discrepancy," in *Proceedings of WIRN'03, pp53-60*, 2003. [Online]. Available: citeseer.ist.psu.edu/633350.html

[5] F. Hickernell, "A generalized discrepancy and quadrature error bound," *Math. Comp. 67, 299-322*, 1998.

[6] A. Owen, *Quasi-Monte Carlo Sampling, A Chapter on QMC for a SIGGRAPH 2003 course*, 2003.

[7] I. M. Sobol', "On the systematic search in a hypercube," vol. 16, no. 5, pp. 790–793, Oct. 1979.

[8] P. Bratley and B. Fox, "Algorithm 659: Implementing sobol's quasirandom sequence generator," *ACM Transactions on Mathematical Software Volume 14, Number 1, pages 88-100*, 1988.

[9] J. G. van der Corput, "Verteilungsfunktionen," *Proc. Ned. Akad. v. Wet., 38:813?821*, 1935.

Method (mean standard deviation of output)	Error rate ± standard deviation
199 runs, 320 examples (1/5-4/5), dim 2	
SuBaggingDecStump (0.137097)	0.110062 ± 0.065750
SuBaggingQRDecStump8 (0.123100)	0.097637 ± 0.048096
SuBaggingQRDecStump4 (0.125160)	0.103565 ± 0.058860
SuBaggingQRDecStump2 (0.125882)	0.101170 ± 0.054242
BaggingDecStump (0.079022)	0.164514 ± 0.121832
BaggingQRDecStump5 (0.074899)	0.154719 ± 0.123209
BaggingQRDecStump4 (0.073276)	0.159332 ± 0.122726
BaggingQRDecStump2 (0.073361)	0.158154 ± 0.119545
BaggingQRDecStump8 (0.074814)	0.159293 ± 0.121817
199 runs, 320 examples (1/5-4/5), dim 4	
SuBaggingDecStump (0.136094)	0.148673 ± 0.066784
SuBaggingQRDecStump8 (0.128807)	0.136267 ± 0.061232
SuBaggingQRDecStump4 (0.127693)	0.137092 ± 0.060400
SuBaggingQRDecStump2 (0.128570)	0.140036 ± 0.063278
BaggingDecStump (0.078661)	0.164632 ± 0.121136
BaggingQRDecStump5 (0.075725)	0.158704 ± 0.120155
BaggingQRDecStump4 (0.074121)	0.159685 ± 0.121453
BaggingQRDecStump2 (0.074604)	0.165162 ± 0.120381
BaggingQRDecStump8 (0.075216)	0.161334 ± 0.122183
199 runs, 1280 examples (1/5-4/5), dim 2	
SuBaggingDecStump (0.098132)	0.034690 ± 0.018896
SuBaggingQRDecStump8 (0.091579)	0.033336 ± 0.017868
SuBaggingQRDecStump4 (0.091416)	0.031559 ± 0.015501
SuBaggingQRDecStump2 (0.093533)	0.032025 ± 0.016090
BaggingDecStump (0.070920)	0.035691 ± 0.078878
BaggingQRDecStump5 (0.066053)	0.022584 ± 0.060712
BaggingQRDecStump4 (0.064879)	0.029012 ± 0.071060
BaggingQRDecStump2 (0.066117)	0.026701 ± 0.067174
BaggingQRDecStump8 (0.065024)	0.028399 ± 0.068832
199 runs, 1280 examples (1/5-4/5), dim 4	
SuBaggingDecStump (0.099501)	0.041349 ± 0.027841
SuBaggingQRDecStump8 (0.092640)	0.038277 ± 0.024346
SuBaggingQRDecStump4 (0.092366)	0.036368 ± 0.018317
SuBaggingQRDecStump2 (0.092609)	0.036520 ± 0.017937
BaggingDecStump (0.071598)	0.030308 ± 0.072339
BaggingQRDecStump5 (0.064187)	0.025577 ± 0.065089
BaggingQRDecStump4 (0.067099)	0.029601 ± 0.072861
BaggingQRDecStump2 (0.070020)	0.024110 ± 0.062809
BaggingQRDecStump8 (0.066850)	0.025391 ± 0.064964

Figure 3: Results for bagging and subagging.

[10] J. Halton, "On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals," *Numerische Mathematik*, vol. 2:84–90, 1960.

[11] B. Efron, "Bootstrap methods: Another look at the jacknkife. annals of statistics 7, 1-26," 1979.

[12] E. Gine, , and Z. J., "Bootstrapping general empirical measures. annals of probability 18, 851-869," 1984.

[13] A. V. D. Vaart and J.-A. Wellner, "Weak convergence and empirical processes. springer series in statistics," 1996.

[14] N. Diamantidis, D. Karlis, and E. Giakoumakis, "Unsupervised stratification of cross-validation for accuracy estimation," *Artificial Intelligence 116, 1-16*, 2000.

[15] F. J. Hickernell, "A generalized discrepancy and quadrature error bound," *Mathematics of Computation*, vol. 67, no. 221, pp. 299–322, 1998. [Online]. Available: citeseer.ist.psu.edu/hickernell97generalized.html

[16] S. Lallich, O. Teytaud, and E. Prudhomme, "Statistical inference and data mining: false discoveries control," in *proceedings of the 17th COMPSTAT Symposium of the IASC*, 2006.