Inplementation details of Hilbert sampling

April 14, 2015

Abstract

This program contains the implementation of the Hilbert sampling developed in He and Owen (2014). To drive the Hilbert sampling, we need two ingredients: one-dimensional randomized quasi-Monte Carlo (RQMC) inputs and Hilbert's curve. We use scrambled van der Courput (VDC) sequences as the QMC inputs. Also, we use Butz' algorithm (see Butz, 1971) to approximate the Hilbert's curve with a desired precision.

1 Hilbert generator

The class *HilberSampling* is the core contribution of this work, which is based on the source code in Lawder (2000). One of the constructors in *HilberSampling* is given as

HilbertGenerator(m, d),

where m is the order of the Hilbert's curve, namely $H_m(x)$, and d is the dimension. Another constructor used to drive the fast algorithm is given as

HilbertGenerator(m, m', d),

where m' is the order of the Hilbert' curve to be reserved and $m' \leq m$. The main functions

convert(P, x) and fastConvert(P, x)

are actually the random mapping function $P = H_m(x) + \boldsymbol{u}$, where $x \in [0, 1]$ is one of the scrambled VDC points, $\boldsymbol{u} \sim \mathbb{U}([0, 2^{-m}]^d)$, and $H_m(x)$ represents the first m bits of $H(x)$. The later one uses the stored bits of $H_{m'}$ to speed up the algorithm. If the first m' bits were stored, the *fastConvert* function only needs to compute the later $m - m'$ bits by the Butz' alogrithm. In such way, a lot of computation can be saved. Otherwise, the function will store the m' bits of $H_{m'}$ and the necessary ingredients which are used to generate the next bit. The total storage is $O(d2^{m'})$.

If the sample size is $n = 2^K$, then we set $m = \lceil K/d \rceil$ and $m' = \lfloor K/d \rfloor$. Note that this program is only implemented for $1 \leq d \leq 31$ and $0 < m' \leq m \leq 31$. If $m' = 0$, the *fastConvert* function will be redirected to the *convert* function.

2 RQMC generator

To make comparisons with RQMC, we generate scrambled Sobol' points (see Sobol', 1967) by using the C++ library of T. Kollig and A. Keller (http://www.uni-kl.de/AG-Heinrich/SamplePack.html). Also, we use the first dimesion of scrambled Sobol' sequence as the scrambled VDC sequence. We use *drand48()* as the pseudo-random numbers generator. One can replace it with other generator in the header file *Drand48.h*.

References

Butz, A. R. (1971). Alternative algorithm for Hilbert's space-filling curve. *IEEE Transactions on Computers*, 20(4):424–426.

He, Z. and Owen, A. B. (2014). Extensible grids: uniform sampling on a space-filling curve. *arXiv preprint arXiv:1406.4549*.

Lawder, J. K. (2000). Calculation of mappings between one and n-dimensional values using the hilbert space-filling curve. *Research Report JL1/00, University of London*.

Sobol', I. M. (1967). On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(7):784–802.