A recycling estimator of Sobol's sensitivity index

Art Owen
Stanford University

September 2012

Abstract

A new estimator of Sobol's global sensitivity index is given in Owen (2012). That estimator attains a better rate of convergence in a certain small effects limit than other estimators. It costs 4 function evaluations per Monte Carlo observation, using parts from three different uniformly distributed random vectors. With some variable reuse, greater efficiency can potentially be obtained. This note develops two methods that recycle the parts of the vectors and reuse some function evaluations. In numerical examples, efficiency gains from about 1.5 fold to about 3 fold are observed.

Introduction

This note retains the notation of Owen (2012). That article also has the motivating context and references to the literature.

The estimator studied there takes the form

$$\frac{1}{n}\sum_{i=1}^{n}(f(\boldsymbol{x}_i)-f(\boldsymbol{z}_{i,u}{:}\boldsymbol{x}_{i,-u})(f(\boldsymbol{x}_{i,u}{:}\boldsymbol{y}_{i,-u})-f(\boldsymbol{y}_i)). \tag{1}$$

It constructs one product of differences from 4 function evaluations per Monte Carlo sample. If we compute it for multiple subsets u, then they can all share function evaluations $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{y}_i)$ but each new subset potentially costs up to 2 more function evaluations per sample.

Of the sample points $\boldsymbol{x}_i$, $\boldsymbol{y}_i$, and $\boldsymbol{z}_i$, the portions used are as depicted here

	$\boldsymbol{x}_{-u}$	$\boldsymbol{y}_{-u}$	$\boldsymbol{z}_{-u}$
$\boldsymbol{x}_u$	●	●	
$\boldsymbol{y}_u$		●	
$\boldsymbol{z}_u$	●		

where a bullet shows a combination used in equation (1) and an empty spot shows a combination not used.

By making use of other combinations of these variables we are able to average more pairs of the form (1) with only a small increase in the number of function evaluations per $(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ triple. One approach is to run all 9 combinations. A second approach is to merge all three of $\boldsymbol{x}_u$, $\boldsymbol{y}_u$ and $\boldsymbol{z}_u$ with $\boldsymbol{x}_{-u}$ and $\boldsymbol{y}_{-u}$ generating 6 combinations and letting $\boldsymbol{z}_{-u}$ remain unused.

Nine fold

Let $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ be independent $\mathbf{U}[0,1]^d$ random variables. We may form 9 distinct points $\boldsymbol{a}_u{:}\boldsymbol{b}_{-u}$ where $\boldsymbol{a}$ and $\boldsymbol{b}$ are in $\{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}\}$. Taking $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{d}$ and $\boldsymbol{e}$ in $\{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}\}$ subject to $\boldsymbol{a}$, $\boldsymbol{c}$ and $\boldsymbol{e}$ distinct as well as $\boldsymbol{b}\ne\boldsymbol{d}$, we find that

$$\begin{aligned}&\mathbb{E}\bigl(g(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c},\boldsymbol{d},\boldsymbol{e})\bigr)=\underline{\tau}_u^2\quad\text{where}\\&\qquad g(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c},\boldsymbol{d},\boldsymbol{e})=(f(\boldsymbol{a}_u{:}\boldsymbol{b}_{-u})-f(\boldsymbol{c}_u{:}\boldsymbol{b}_{-u}))(f(\boldsymbol{a}_u{:}\boldsymbol{d}_{-u})-f(\boldsymbol{e}_u{:}\boldsymbol{d}_{-u})).\end{aligned}$$

There are 36 ways to assign $\boldsymbol{a}$ through $\boldsymbol{e}$ but they only yield 18 different formulas because the product inside the expectation evaluates to the same quantity if $(\boldsymbol{b},\boldsymbol{c})$ is swapped with $(\boldsymbol{d},\boldsymbol{e})$. We introduce the ordering $\boldsymbol{x}\preceq\boldsymbol{y}\preceq\boldsymbol{z}$ on the (labels of) the triple of vectors. Then we immediately get:

Proposition 1 *The expected value of*

$$\frac{1}{18}\sum_{\substack{\boldsymbol{a},\boldsymbol{c},\boldsymbol{e}\in\{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}\}\\ \boldsymbol{a},\boldsymbol{c},\boldsymbol{e}\,\text{distinct}}}\ \sum_{\substack{\boldsymbol{b},\boldsymbol{d}\in\{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}\}\\ \boldsymbol{b}\preceq\boldsymbol{d}}} g(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c},\boldsymbol{d},\boldsymbol{e})$$

is $\underline{\tau}_u^2$.

Each of the $g(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c},\boldsymbol{d},\boldsymbol{e})$ on its own attains the better convergence rate when $\boldsymbol{x}_u$ is unimportant (in that $\overline{\tau}_u^2$ is small). Therefore averaging these 18 quantities also attains that rate.

This method generates 18 bilinear terms per triple. It costs 9 function evaluations of which 3, namely $f(\boldsymbol{x}_i)$, $f(\boldsymbol{y}_i)$ and $f(\boldsymbol{z}_i)$ can be reused with different u while up to 6 new evaluations may be needed for a new subset u.

When many sets u are being considered we get essentially 18 bilinear terms for 6 function evaluations, or an average cost of $1/3$. The original method has a corresponding cost of 2. The new method could potentially be 6 times as efficient, but this is unlikely because the multiple bilinear terms for a given triple can be correlated. Indeed for product functions many of the 18 are identical.

Six fold

This method is similar to the nine fold strategy except that $\boldsymbol{z}_{-u}$ is not to be used. In the notation above, that forces $\boldsymbol{b}=\boldsymbol{x}$ and $\boldsymbol{d}=\boldsymbol{y}$. The result is that we get 6 bilinear terms per triple. We make 6 function evaluations per triple of which 2 can be reused ($f(\boldsymbol{x})$ and $f(\boldsymbol{y})$) while the other 4 may have to be new evaluations.

Proposition 2 *The expected value of*

$$\frac{1}{6}\sum_{\substack{\boldsymbol{a},\boldsymbol{c},\boldsymbol{e}\,\in\{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}\}\\ \boldsymbol{a},\boldsymbol{c},\boldsymbol{e}\,\text{distinct}}} g(\boldsymbol{a},\boldsymbol{x},\boldsymbol{c},\boldsymbol{y},\boldsymbol{e})$$

is $\underline{\tau}_u^2$.

A crude analysis does not favor this method. It gets 6 new bilinear terms for 4 new evaluations, so the cost is 2/3 or double what the nine fold method costs. Like the nine fold method it is very efficient when $\overline{\tau}_u^2$ is small.

Large indices

All of the methods considered here work very well when $\overline{\tau}_u^2$ is small. It would be interesting to have a method that works well whether or not $\overline{\tau}_u^2$ or $\underline{\tau}_u^2$ is small.

Suppose instead that $\boldsymbol{x}_u$ is very important. At the extreme we may suppose that $\boldsymbol{x}_{-u}$ is irrelevant. Then the nine fold method has 18 bilinear terms but they are identical in groups of three corresponding to the three ways to make the now irrelevant choice of $(\boldsymbol{d},\boldsymbol{e}) \in \{(\boldsymbol{x},\boldsymbol{y}),(\boldsymbol{x},\boldsymbol{z}),(\boldsymbol{y},\boldsymbol{z})\}$. In that limit the nine fold method scores 6 bilinear terms but still requires up to 6 new function evaluations for a cost of 1. As a result the six fold method may outperform the nine fold method when $\boldsymbol{x}_u$ is very important.

Examples

Equation (1) was compared with the six fold and nine fold strategies on a few functions on $[0,1]^2$:

$$\begin{aligned}
f_1(x_1,x_2) &= x_1,\\
f_2(x_1,x_2) &= x_2,\\
f_3(x_1,x_2) &= \cos(x_1/(10^{-100}+x_2)),\\
f_4(x_1,x_2) &= x_1x_2,\\
f_5(x_1,x_2) &= x_1+x_2, \quad \text{and}\\
f_6(x_1,x_2) &= 2(x_1-1/2)(x_2-1/2),
\end{aligned}$$

with $u=\{1\}$ and $u^c=\{2\}$. The 10^{-100} in f_3 is to prevent difficulties with x_2 rounded down to 0. That is quite unlikely with a good random number generator in Monte Carlo sampling, but might occur in some other quadrature techniques.

The table following this paragraph presents cost times variance for three methods on these 6 functions. The costs are 2 for the original method, 6 for nine fold and 4 for six fold. These are the incremental costs required to add another subset u to an existing list of sets. They are based on $R=10{,}000$

independent simulations. The nine fold method generates 18 estimates which were turned into 18 variance estimates then averaged to get the variance for the original method.

$C \times V$	Original	Nine	Six
f_1	0.0531	0.02532	**0.0169**
f_2	0	0	0
f_3	0.5191	**0.1727**	0.1948
f_4	0.0065	0.0035	**0.0032**
f_5	0.0531	0.0253	**0.0169**
f_6	0.0075	**0.0025**	0.0051

On this small set of examples the original method was never more efficient than either of the methods that employed recycling. For f_2 all three methods are exact. Functions f_1 and f_5 give identical results because the model is additive (no x_1–x_2 interaction) and the estimators filter out the main effect of x_2, leaving a function of x_1 alone. Indeed f_1 may stand in for any linear function of x_1 and x_2 having a nonzero coefficient on x_1. Function f_6 is a pure interaction having no main effects for x_1 or x_2.

Efficiency ratios for functions other than f_2 where it is not well defined and f_5 (which matches f_1) are tabulated below taking the original method to be of unit efficiency.

Efficiency	f_1	f_3	f_4	f_6
Nine	2.10	3.01	1.84	3.04
Six	3.14	2.66	2.03	1.47

Estimates for u^c

The estimator

$$\frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{x}_i)(f(\boldsymbol{x}_{i,u}{:}\boldsymbol{y}_{i,-u}) - f(\boldsymbol{y}_i)) \qquad (2)$$

is widely used for $\underline{\tau}_u^2$. See Sobol et al. (2007). This estimator does not do as well as (1) in the limit of very small $\overline{\tau}_u^2$ but it has some desirable properties. Adding a new set u to the list of desired sets requires only one additional function evaluation. Furthermore if the estimate is available for u, the same function values can be used to estimate $\underline{\tau}_{-u}^2$ by interchanging $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{y}_i)(f(\boldsymbol{x}_{i,u}{:}\boldsymbol{y}_{i,-u}) - f(\boldsymbol{x}_i)). \qquad (3)$$

In the very common case where $\underline{\tau}_{\{j\}}^2$ and $\overline{\tau}_{\{j\}}^2 = \sigma^2 - \underline{\tau}_{-\{j\}}^2$ are both needed for $j = 1, \dots, d$ this feature essentially cuts the needed number of function evaluations in half.

Their estimator uses the function values in this matrix

	$\boldsymbol{x}_{-u}$	$\boldsymbol{y}_{-u}$	$\boldsymbol{z}_{-u}$
$\boldsymbol{x}_u$	●	●	
$\boldsymbol{y}_u$		●	
$\boldsymbol{z}_u$			

Switching u to $-u$ corresponds to transposing the matrix of bullets, which then becomes lower triangular. Interchanging $\boldsymbol{x}$ and $\boldsymbol{y}$ amounts to a permutation applied to both rows and columns bringing the bullet back above the diagonal.

The function values for (1) cannot be re-arranged to get an analogous doubly centered estimator for $\underline{\tau}^2_{-u}$, but the elements of the nine fold estimator can be so arranged. To see this, revisit the 3×3 matrix on page 1 containing 5 bullets. If we have all 9 function values then we can interchange u and $-u$ for any pattern of bullets and still find we have all the necessary function evaluations.

For the original method, the 4 function values we need lie inside a 3×2 submatrix. To interchange the roles of u and u^c we transpose the matrix of bullets and blanks. The bullets now lie within a 2×3 submatrix. No permutation applied to both rows and columns will yield three non-empty rows for this matrix. Similarly, the six fold method cannot be reversed to yield an analogous estimator for $\underline{\tau}^2_{-u}$.

The original matrix does however contain a triangle of bullets that allow the singly centered estimator (2) to be computed for u^c from the same data used to compute (1). Alternatively, given the sampling pattern for (2) if one then wants to apply the estimator (1) it requires one more function value per sample.

References

Owen, A. B. (2012). Better estimation of small Sobol' sensitivity indices. *ACM transactions on mathematical software*, (to appear).

Sobol, I. M., Tarantola, S., Gatelli, D., Kucherenko, S. S., and Mauntz, W. (2007). Estimating the approximation error when fixing unessential factors in global sensitivity analysis. *Reliability Engineering & System Safety*, 92(7):957–960.