Using GEM-SA for Probabilistic Sensitivity Analysis

Jeremy Oakley

CHEBS, University of Sheffield

j.oakley@sheffield.ac.uk

July 5, 2005

1 Introduction

GEM-SA is a general-purpose tool written by Marc Kennedy for building Gaussian process emulators for deterministic models, and can be downloaded from www.shef.ac.uk/st1mck/code.html These accompanying R functions are designed to be used in conjunction with GEM-SA so that very large numbers of model runs can be obtained from a computationally expensive health economic model.

The emulator is a computationally cheap surrogate for the economic model. Once the emulator has been built, you should proceed as normal in conducting your PSA, but with the emulator used in place of the economic model. The emulator in GEM-SA is designed for a single output variable, and you will need to construct one emulator for costs and a second for effectiveness. GEM-SA currently allows a maximum of 30 input variables, and 400 runs of the model.

2 Terminology

You should note that GEM-SA uses terminology from the computer experiments, which differs from that in health economics. In GEM-SA, the process of obtaining a distribution of the model output given probability distributions for the model

inputs is known as *uncertainty analysis* (i.e. PSA in the health economics community), and *sensitivity analysis* refers to identifying which uncertain inputs have the greatest contribution to the output uncertainty. Specifically, GEM-SA is designed to conduct a *variance-based* sensitivity analysis, in which the contribution to the output variance from each input parameter is determined. (Currently, GEM-SA requires the input distributions to be either independent normal or uniform to do this, but can fit the emulator regardless of the input distributions).

3 Building and using the emulator - a walkthrough

This illustrated with a simple test example. We have a simple function of 5 uncertain input parameters, $X_1, \ldots, X_5$, with

$X_1 \sim N(0,1)$, $\log X_2 \sim N(0,1)$, $X_3 \sim Gamma(3,2)$, $X_4 \sim Beta(10,20)$, $X_5 \sim U[0,5]$.

1. Open the script `inputsetup.R` in a text editor.

2. Edit line 4 to specify the distribution types (normal, lognormal, gamma etc.) in the vector `distributions`:
   `distributions<-c('n','l','g','b','u')`.

3. Edit line 15 to specify the parameters of each distribution in the matrix `parameters`:
   `parameters<-matrix(c(0,1,0,1,3,2,10,20,0,5),nrow=np,ncol=2,byrow=T)`

4. Specify the number of model runs as the first argument in `setupdesign` in line 19. E.g. for 100 model runs:
   `inputs<-setupdesign(100,distributions,parameters)`.

5. R will write the design points to the file `c:/inputs.txt`. Edit the path in line 23 if you wish to change the location of the inputs file.

6. Save your changes to the script `inputsetup.R` and run this script in R. You may find it helpful to save `distributions` and `parameters` as R objects for later use.

7. N.B. The inputs are chosen using a maximin LHS scheme in the function `setupdesign.R`. This function generates a Latin Hypercube sample and then, iteratively, randomly permutes the arrangement of design points in order to increase the minimum distance between any two points. The number of iterations in line 50 of `setupdesign.R` is set at 100, but increasing this will improve the design (at the cost of more computing time).

8. You must now run your economic model at the input values specified in `c:/inputs.txt`. Each row corresponds to one model run, with the inputs in order as specified in `distributions`. If your model produces two (or more) outputs (e.g. cost and effectiveness), you must treat each output separately and build one emulator per output. The output must be saved in a plain text file, in a single column with one row per model run. For this example, we will use the example `testfunction.R` included in the Zip file. In R, type

   ```
   outputs<-testfunction(inputs)
   write(outputs,"c:/outputs.txt",ncolumns=1)
   ```

   to produce a vector of output values and write them to the file `c:/outputs.txt`.

9. Now that you have the input and output files, start up GEM-SA. Refer to the on-line user manual (in particular "Creating a project") for help on building the emulator. Note that it is not necessary to specify input distributions within GEM-SA for our purposes. On the options tab you should also uncheck the "calculate main effects" and "sum effects" options. Once the project has been run, you can exit GEM-SA.

10. Returning to R, open the script `gpsetup.R` in a text editor.

11. Edit the path in line 2 as necessary depending on where GEM-SA has been installed.

12. Edit line 5 as necessary to change the location of the inputs file, the number of model runs (`nrow`) and the number of input parameters (`ncol`).

13. Edit line 6 as necessary to change the location of the outputs file and the number of model runs (`nrow`).

14. Save any changes and run the script `gpsetup.R`.

15. You can now estimate the model output at any set of new inputs `x` using the function

    `emulate<(x,inputs,betahat,B,r)`

    where each row of `x` is one set of input parameters. A large monte carlo set of `n` inputs can be generated within R by typing

    `x<-mcinputgenerate(n,distributions,parameters)`

4 Testing the accuracy of the emulator

This can be done within GEM-SA using cross validation: a model run is left out of the data for building the emulator, and the emulator prediction at the omitted input value is then compared with the known corresponding output. Further details can be found in the "Cross validation" section in the GEM-SA online manual.