A Design Search Algorithm for Generalized Linear Models

D.C. Woods*
Southampton Statistical Sciences Research Institute
University of Southampton

Introduction

The search algorithm presented here is capable of finding either exact or continuous designs for generalized linear models, either locally D-optimal designs under a given model or robust designs which compromise across a class of models. Background and details of the statistical methods are given in Woods, Lewis, Eccleston and Russell (2006), available at `http://www.maths.soton.ac.uk/staff/woods/glm_design`.

Simulated annealing

A simulated annealing algorithm (see, for example, Spall (2003, ch.8) and Haines (1987)) is used to solve the optimization problem. Simulated annealing is modelled on the cooling of materials in metallurgy. It is a *probabilistic* optimization technique, as opposed to a greedy algorithm, meaning that changes to the current *state* (design) are accepted according to a *transition probability*. In the original formulation, for a minimization problem, transitions which decrease the *energy function* (objective function) are accepted with probability 1 but transitions that increase the energy function also have non-zero probability of being accepted, allowing the algorithm to move away from local optima in the search space. The transition probabilities are determined according to the Boltzmann distribution, as used in the Metropolis-Hastings algorithm (Metropolis et al., 1953). Under this formulation, the probability of accepting an "uphill" transition decreases with the increase in energy between the initial state and the transition and also decreases as the *temperature* of the system decreases. The initial temperature is user-controlled and often a geometric cooling scheme is employed with the temperature decreasing by a fixed factor after a set number of iterations. A nice description of simulated annealing can be found at `http://en.wikipedia.org/wiki/Simulated_annealing`.

In a design of experiments setting with continuous variables, a simulated annealing algorithm makes random perturbations to the design points, compares the existing design to the perturbed design and accepts the perturbation with probability inversely proportional to the increase in the objective function (in a minimization problem). Changes which improve the design are always accepted; changes which result in a singular design, that is, a design from which the model cannot be estimated, are always rejected. As the system cools, i.e. the temperature parameter decreases, the probability of accepting poor moves also decreases. When the temperature is 0, a greedy algorithm results.

The algorithm allows the user to specify the initial acceptance probability, the geometric cooling parameter, the parameter controlling the geometric decrease in the step size, the number of iterations between changes in step size (perturbation size), the maximum number of iterations and the minimum step size. The stopping rule of the algorithm is either (i) the maximum number of iterations has been reached or (ii) the minimum step size has been obtained.

*Email: `D.C.Woods@maths.soton.ac.uk`
WWW: `http://www.maths.soton.ac.uk/staff/woods`

Compiling and running

The algorithm is written in C++ and was developed under Linux using the g++ compiler. It requires the GNU Scientific Library (GSL) (`http://www.gnu.org/software/gsl/`). A typical commandline to compile and link the code on a Linux system with GSL installed in its default location would be

```
g++ -o cont_search.run cont_glm_search.cpp -lm -lgsl -lgslcblas
```

Input and output

By default, input is read from a file called cont_search.in. See cont_search.inREADME for a description of an example input file. The default output file is output.out. These can be altered by changing the relevant lines near the beginning of cont_glm_search.cpp.

Continuous designs

To find continuous (or approximate) designs, where each support point has a corresponding design weight indicating the proportion of experimental effort to be allocated to that point, it is necessary to specify an extra variable (placed last in the input file) with range [0,1]. When specifying the terms in the model, only include this variable as a "0". Specify "approx" as the type of design. See cont_search.inCONT for an example.

For various examples using continuous designs, the optimality of the designs from the algorithm has been confirmed using the necessary and sufficient conditions that can be established; see Woods and Lewis (2005).

C++ classes

Eight different C++ classes are used in the algorithm code and are included in the zip file glm_search_SA.zip.

- matrix_class - a class of 2D arrays that interfaces with GSL functions for matrices
- matrixoperations_class - a class of static utilities for use with matrix_class
- random_class - a static utility class for the generation of pseudo-random numbers from various distributions using the GSL functions
- cont_design_class - objects of this class hold designs under a particular model
- model_class - these objects hold the link function and model parameters
- criteria_class - static utility class with methods for evaluating the objective functions for the *D*- and *A*- optimality criteria
- update_class - static utility class with methods for the updating formula for *D*-optimality under exact designs
- cont_search_class - static utility class with methods for simulated annealing design search

Table 1: Performance of the SA algorithm for examples 1 and 2, based on 10 random starts.

Example	Runs	Step-size	Update	Average Run Time	Average Obj. fun. value
1	16	1.1	No	1m35.56	2.3010
1	16	1.1	Yes	1m26.49	2.2406
1	16	1.01	No	9m21.11	2.2793
1	16	1.01	Yes	5m29.10	2.2558
1	24	1.1	No	3m32.80	97.4659
1	24	1.1	Yes	2m32.90	97.0874
1	24	1.01	No	22m15.32	98.6193
1	24	1.01	Yes	9m45.66	98.7167

Performance and tuning

The algorithm was timed finding exact designs for Example 1 from Section 5 of Woods et al (2006). Average timings for 10 design searches are given in Table 1, both with and without the use of an updating formula based on that given for linear models by Fedorov (1972, p.162). All runs of the algorithm were performed on a desktop PC with a 3.2Ghz Intel Pentium IV processor. The computational cost of complete objective function evaluations is dependent upon the design size n whereas evaluation of the objective function via the updating formula is independent of n. Hence, greater computational savings accrue from using the updating formula for problems with large design sizes.

The "step size" column gives the settings for the geometric decrease in the step size, which only occurs when the average number of accepted transitions is between 0.4 and 0.6. For these timings, this average was taken over 20 iterations. The temperature was cooled continually, at a rate of 0.9. Choice of these parameters determines the length of the design search and, in some cases, the thoroughness of the search. Therefore a trade-off is required. Large examples with many variables usually require the most intensive search of the design space and therefore require considerable search time. An important technique is then to decrease the annealing parameters but run several searches in parallel using, for example, a Beowulf cluster.

Acknowledgements

This work took place as part of the Comb*e*chem e-Science project (EPSRC grant GR/R67729).

References

Fedorov, V.V. (1972), *Theory of Optimal Experiments.* Academic Press, New York.

Haines, L.M. (1987). The application of the annealing algorithm to the construction of exact optimal designs for linear-regression models. *Technometrics*, **29**, 439-447.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, **21**, 1087-1092.

Spall, J.C. (2003). *Introduction to Stochastic Search and Optimization: Estimation, Simulation and Control.* Wiley, New York.

Woods, D.C. and Lewis, S.M. (2005). Continuous optimal designs under model uncertainty. *Technical Report 389, School of Mathematics, University of Southampton.*

Woods, D.C., Lewis, S.M., Eccleston, J.A. and Russell, K.G. (2006). Designs for generalized linear models with several variables and model uncertainty. *Technometrics*, 48, 284-292.