D-optimal Designs for Logistic Models using Metaheuristics

Will Gertsch

June 3, 2021

Outline

1. Logistic regression model and information matrix
2. Optimal design background
3. Particle swarm and genetic algorithm
4. D-optimal designs for a linear predictor
5. D-optimal designs for a quadratic predictor

Logistic Model

Suppose $y_i \sim \text{Bernoulli}(p_i)$ for $i = 1, \ldots, n$. The logit link function is defined as

$$p_i = \frac{1}{1 + e^{-\eta_i}} = \frac{e^{\eta_i}}{1 + e^{\eta_i}}$$

where $\eta_i = \beta_0 + \beta_1 x_i$ and $0 \leq p_i < 1$
We may also use a quadratic model $\eta_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2$.

Information Matrix

The information matrix for the logistic model can be derived as

$$M(\beta) = \sum_{i=1}^{k} p_i(1 - p_i) f(x_i) f(x_i)' = X'WX$$

where W is a diagonal weight matrix with entries $p_i(1 - p_i)$. The design matrix X has rows $f(x_i)' = (1, x_i)$ if η_i is linear. If η_i is quadratic, then $f(x_i)' = (1, x_i, x_i^2)$.

Design Notation

We denote a design ξ using weight notation. If we have N samples total and n_i samples for each design point $x_1 \ldots, x_k$, let

$$\xi = \begin{pmatrix} x_1 & \ldots & x_k \\ w_1 & \ldots & w_k \end{pmatrix}$$

where $w_i = n_i/N$ are weights.

Information Matrices

In optimal design, a design ξ implies a corresponding information matrix

$$M = M(\xi, \beta) = \sum_{i=1}^{k} w_i p_i (1 - p_i) f(x_i) f(x_i)' = X'WX$$

where W now has diagonal entries $w_i p_i (1 - p_i)$

Locally Optimal Designs

For a given value of the parameter vector β, we can find a locally optimal design by minimizing functions of the information matrix. Below are some common examples:

- **D-optimality:** $-\log\left(\det(M)\right)$
- A-optimality: $\mathrm{tr}(M^{-1})$
- E-optimality: $-\max_M \min_{Mv=\lambda v} \lambda$

In this presentation, we will focus on D-optimality.

Equivalence Theorem for D-optimality

We can test if a design is D-optimal by computing the sensitivity function

$$ch(x) = g(x)f(x)'M(\xi, \beta)^{-1}f(x) - p$$

where p is the number of parameters in the model and

$$g(x) = \frac{\exp(\eta)}{(1 + \exp(\eta))^2}$$

$\eta = \beta_0 + \beta_1 x$ or $\eta = \beta_0 + \beta_1 x + \beta_2 x^2$

Plotting ch(x)

If $ch(x) = g(x)f(x)'M(\xi,\beta)^{-1}f(x) - p \leq 0$ with equality at the design points, then the design ξ is D-optimal. In other words, we can plot $ch(x)$ to see if a design is optimal.

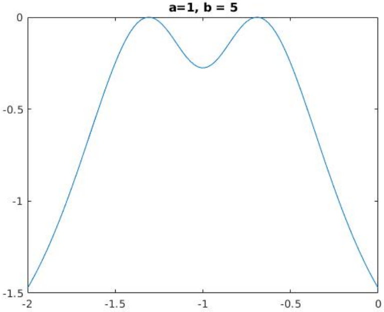


Figure: ch(x) for a D-optimal design

Review of Previous Results

Papers on the logistic model D-optimality:

- Mathew, Sinha (2001) derive D-optimal designs for linear η.
- Sebastiani, Settimi (1992) derive designs on different design intervals for linear η.
- Lall et al. (2018) use a Fedorov algorithm to find designs on $[-1, 1]$.
- Fornius (2005) found designs numerically for quadratic η.

My Contributions

I wrote software that uses metaheuristic optimization algorithms to

- Find D-optimal designs for linear and quadratic logistic models.
- Find designs on any design interval.
- Change number of design points.

Choice of algorithm

I used 2 algorithms to find designs:

- Particle swarm optimization (PSO)
- Genetic algorithm (GA)
- Using implementation in PlatEMO with default parameters.
- PSO may work better than GA or vice-versa

These algorithms were chosen because they worked the best out of all the other single-objective algorithms in PlatEMO for this problem.

Particle Swarm Optimization

Particle swarm optimization (PSO) (Kennedy, Ebernhart 1995) is a nature-inspired algorithm to find the optimal value of an objective function. It is based on fish and bird schooling behavior.

Algorithm:

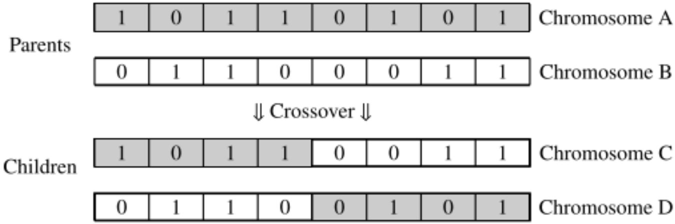
1. Initialize N particles x_i with velocities v_i.
2. Find particle with best objective value.
3. Update velocity v_i^{t+1} for all particles such that they are drawn towards the current best global value and towards the current best for particle i.
4. Update location $x_i^{t+1} = x_i^t + v_i^{t+1}$ for all particles.
5. Repeat steps 2-4.

Genetic Algorithm

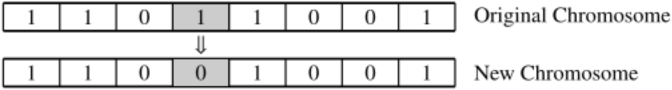
The genetic algorithm (GA) (Holland 1975) is based on natural selection.It has 3 main steps: crossover, mutation, and selection.

- x_i's are encoded as strings called chromosomes.
- Crossover and mutation steps operate on these chromosomes.
- Diagrams from *Nature-Inspired Optimization Algorithms* by Xin-She Yang.

Crossover


Parents
1 0 1 1 0 1 0 1 Chromosome A
0 1 1 0 0 0 1 1 Chromosome B
⇓ Crossover ⇓
Children
1 0 1 1 0 0 1 1 Chromosome C
0 1 1 0 0 1 0 1 Chromosome D

Mutation



Genetic Algorithm

Algorithm:

1. Initialize N solutions.
2. Crossover: swap characteristics of 2 parent solutions with prob. p_c to produce 2 child solutions.
3. Mutation: randomly alter a characteristic of a child solutions with prob. p_m.
4. Selection: Accept child solutions if fitness increases.
5. Best N solutions go on to next generation.
6. Repeat steps 2-5.

This is the basic format of a genetic algorithm. Details may vary with implementation.

Results from Mathew, Sinha

Mathew, Sinha show that the locally *D*-optimal design for a logistic model with $\eta = \beta_0 + \beta_1 x$ has two equally weighted design points

$$x_1, x_2 = \frac{\pm 1.5434 - \beta_0}{\beta_1}$$

Results from Mathew, Sinha

Algorithms were run with swarm size of 1000 and for 1000000 objective function evaluations. Both algorithms had the same initial values.

β	Theoretical	PSO	GA
0,1	$\begin{bmatrix} -1.5434 & 1.5434 \\ 0.5 & 0.5 \end{bmatrix}$	$\begin{bmatrix} -1.5434 & 1.5434 \\ 0.5000 & 0.5000 \end{bmatrix}$	$\begin{bmatrix} -1.5452 & 1.5435 \\ 0.5000 & 0.5000 \end{bmatrix}$
0.3,0.4	$\begin{bmatrix} -4.6085 & 3.1085 \\ 0.5 & 0.5 \end{bmatrix}$	$\begin{bmatrix} -4.6085 & 3.1085 \\ 0.5000 & 0.5000 \end{bmatrix}$	$\begin{bmatrix} -4.6091 & 3.1083 \\ 0.5000 & 0.5000 \end{bmatrix}$
2,-5	$\begin{bmatrix} 0.0913 & 0.7087 \\ 0.5 & 0.5 \end{bmatrix}$	$\begin{bmatrix} 0.0913 & 0.7087 \\ 0.5000 & 0.5000 \end{bmatrix}$	$\begin{bmatrix} 0.0923 & 0.7109 \\ 0.5000 & 0.5000 \end{bmatrix}$

PSO tends to converge quicker for this problem but GA comes very close to theoretical solution.

Results from Lall et al.

Lall et. al use a modified Fedorov algorithm to find D-optimal designs on $[-1, 1]$. PSO and GA are run with the same parameters as last slide.

β_0	β_1	$x_1(F)$	$x_2(F)$	$x_1(PSO)$	$x_2(PSO)$	$x_1(GA)$	$x_2(GA)$
0.1	0.5	-1	1	-1	1	-1	1
1	1	-1	1	-1	1	-1	1
1	4	-0.636	0.136	-0.636	0.136	-0.636	0.136

PSO and GA both are able to confirm the solution.

Results Sebastiani

Sebastiani derives theoretical results for all possible combinations of design intervals. The resulting designs are modifications of the optimal design on $(-\infty, \infty)$. To illustrate, I selected $\beta_0 = 0, \beta_1 = 1$. All resulting designs were equally weighted.

	$[-\infty, \infty]$	$[-1, \infty]$	$[-\infty, 1]$	$[-1, 1]$	$[10, 20]$
$x_1(T)$	-1.543	-1	-1.796	-1	?
$x_2(T)$	1.543	1.796	1	1	?
$x_1(PSO)$	-1.543	-1	-1.796	-1	10^*
$x_2(PSO)$	1.543	1.796	1	1	10^*
$x_1(GA)$	-1.545	-1	-1.796	-1	10
$x_2(GA)$	1.544	1.796	1	1	12

$*$ PSO had trouble converging to the correct solution in this case.

Note also that the last case is not covered in Sebastiani.

Results from Fornius

Fornious (2005) derives D-optimal designs for the quadratic model with $\eta_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2$. Designs have either 3 or 4 support points depending on shape of response.

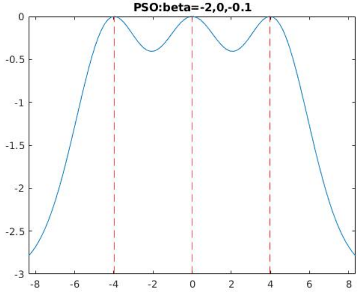
Nominal values	Design
$2, 0, -0.1$	$\begin{bmatrix} -5.7185 & -2.7017 & 2.7017 & 5.7185 \\ 0.3138 & 0.1862 & 0.1862 & 0.3138 \end{bmatrix}$
$2, 0, -4$	$\begin{bmatrix} -0.9042 & -0.4272 & 0.4272 & 0.9042 \\ 0.3138 & 0.1862 & 0.1862 & 0.3138 \end{bmatrix}$
$-2, 0, -0.1$	$\begin{bmatrix} -3.9819 & 0 & 3.9819 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$
$-2, 0, -4$	$\begin{bmatrix} -0.6296 & 0 & 0.6296 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$

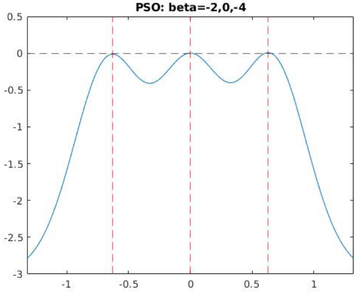
$\beta_1 \neq 0$ shifts the design.

What about on different design intervals?

PSO Results 1

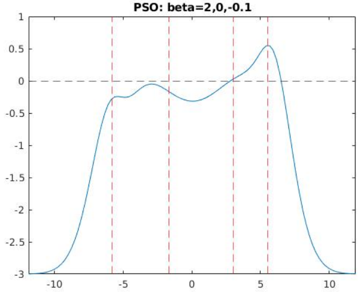
PSO can find optimal designs for the 3 point cases.

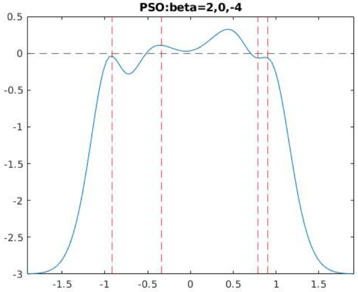




PSO Results 2

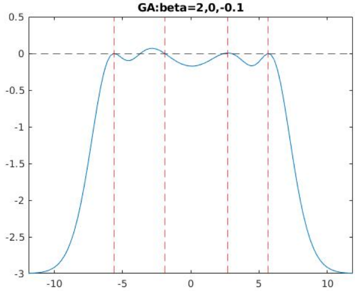
PSO has trouble finding the optimal design when the design has 4 points.

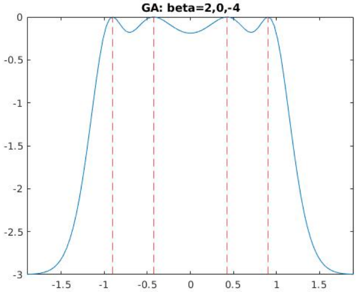




GA Results 1

GA has much better performance on the 4 point designs.

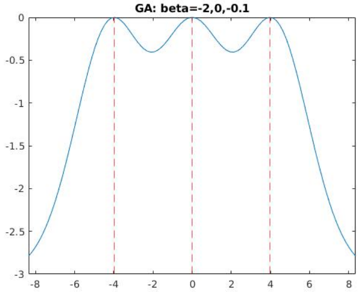


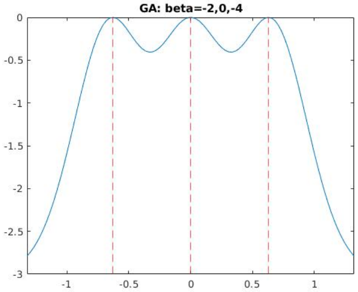


In the case when $\beta = (2, 0, -0.1)^T$, the design produced by GA is not still not quite optimal. Parameter tuning?

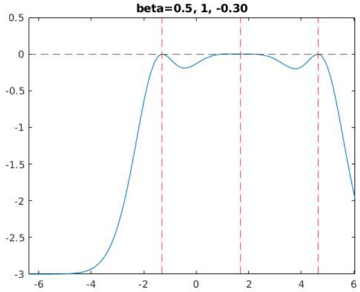
GA Results 2

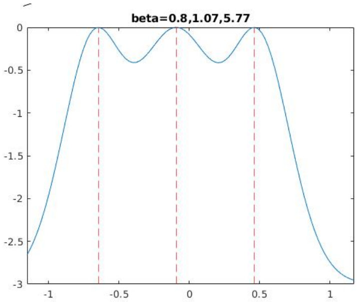
GA can also find the optimal 3 point designs.





Other Designs





Optimal designs for quadratic model on any interval.

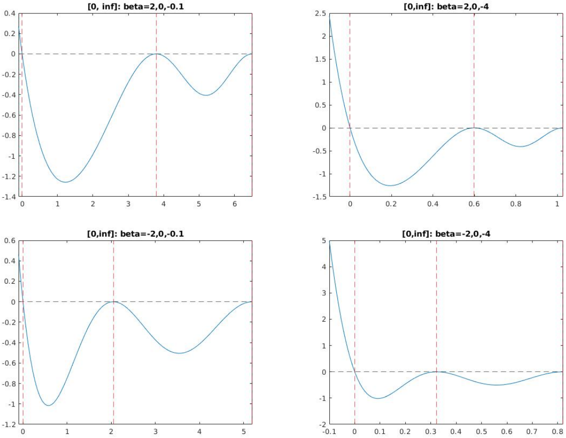
The previous results were all for the case where $x \in (-\infty, \infty)$. What if the design interval was one of the following?

- $[0, \infty)$
- $[-1, 1]$
- $[a, b]$ where a, b are larger than the optimal design points on $(-\infty, \infty)$.
- $[0, 1]$

I will use GA to find these optimal designs as GA seems to have better performance compared to PSO.

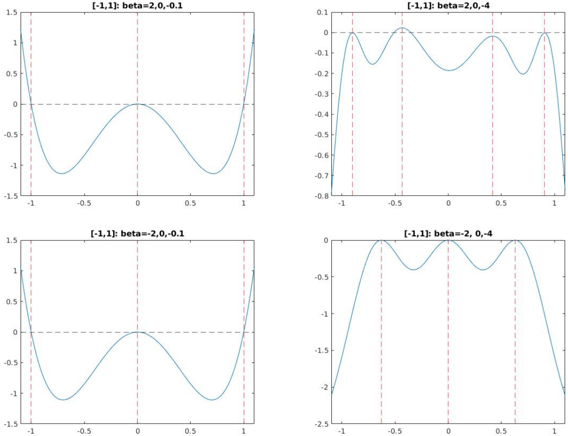
$[0, \infty)$

Strictly positive designs have 3 support points.



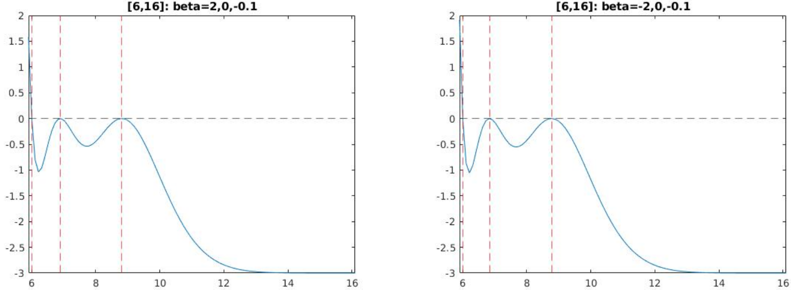
$[-1, 1]$

When beta = (2, 0,-4), the global optimal design points are already within the interval, so we get the 4 point design.



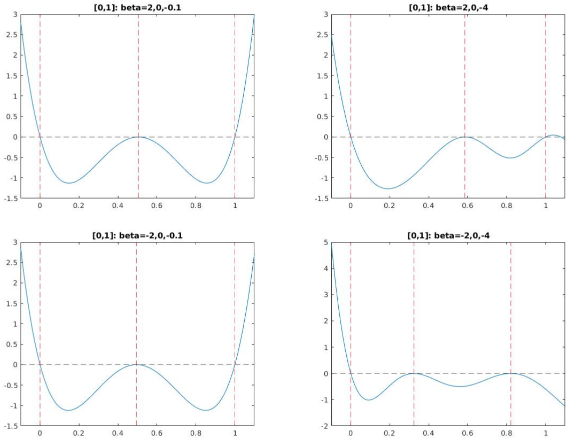
[6, 16]

beta=(2,0,-4) has issues. 1 point at 6 is close. Same with beta=(-2,0,-4). The two other nominal values are better.

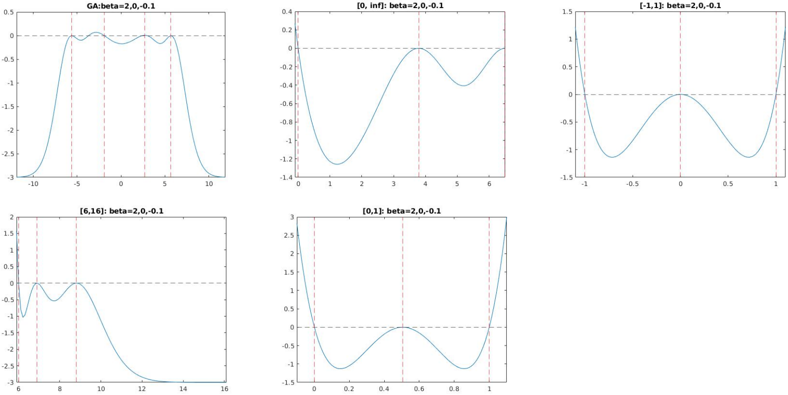


[0, 1]

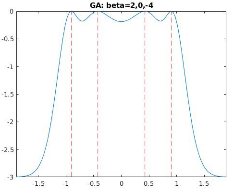
Using the unit interval produces 3 point designs for out set of nominal values.

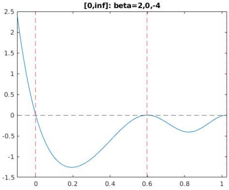


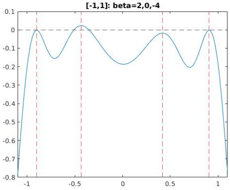
beta = (2, 0, -0.1)

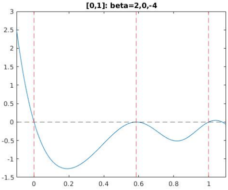

GA:beta=2,0,-0.1
[0, inf]: beta=2,0,-0.1
[-1,1]: beta=2,0,-0.1
[6,16]: beta=2,0,-0.1
[0,1]: beta=2,0,-0.1

beta= (2,0,-4)

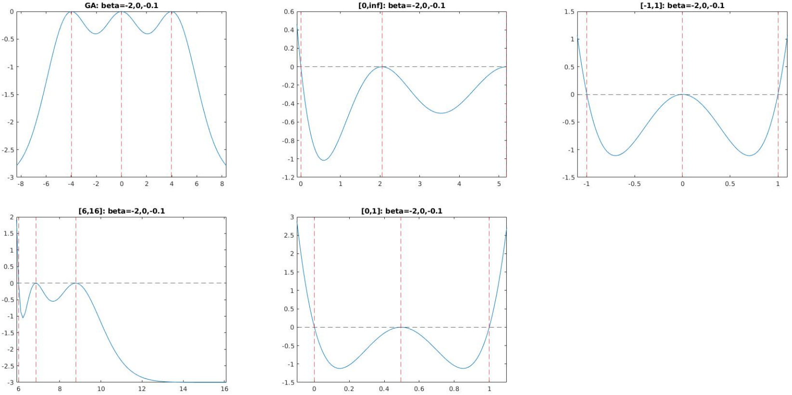

GA: beta=2,0,-4
0
-0.5
-1
-1.5
-2
-2.5
-3
-1.5 -1 -0.5 0 0.5 1 1.5


[0,inf]: beta=2,0,-4
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0 0.2 0.4 0.6 0.8 1


[-1,1]: beta=2,0,-4
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
-0.8
-1 -0.5 0 0.5 1

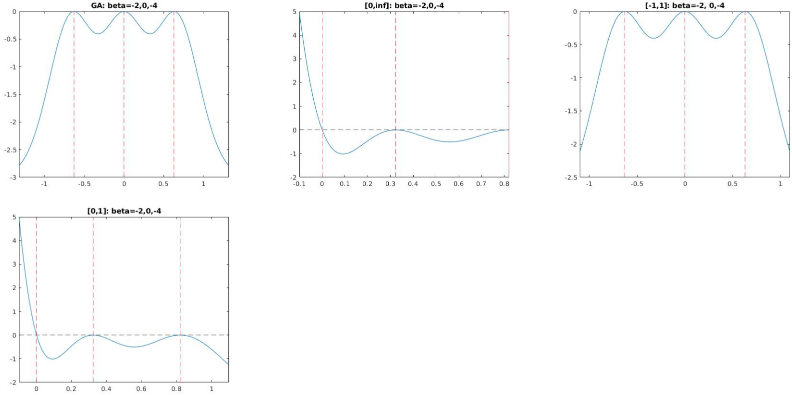

[0,1]: beta=2,0,-4
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0 0.2 0.4 0.6 0.8 1

beta=(-2,0,-0.1)



GA: beta=-2,0,-0.1
[0,inf]: beta=-2,0,-0.1
[-1,1]: beta=-2,0,-0.1
[6,16]: beta=-2,0,-0.1
[0,1]: beta=-2,0,-0.1

beta=(-2,0,-4)



Conclusions

- My code is able to replicate previous results for locally D-optimal designs.
- Also is able to find designs on user-specified design intervals.
- Code can easily be extended to any degree polynomial.
- Future work: fractional polynomials.
- Demonstrate PlatEMO code.

References

- Paola Sebastiani, Raffaella Settimi (1997), *A note on D-optimal designs for a logistic regression model*, Journal of Statistical Planning and Inference
- Shwetank Lall, Seema Jaggi, Eldho Varghese, Cini Varghese & Arpan Bhowmik (2018): *An algorithmic approach to construct D-optimal saturated designs for logistic model*, Journal of Statistical Computation and Simulation
- Thomas Mathew, Bikas Kumar Sinha (2001), *Optimal designs for binary data under logistic regression*, Journal of Statistical Planning and Inference
- Ellinor Fornius (2005), *D-optimal Designs for Quadratic Logistic Regression Models*