

Optimal Design Foundations

John Sherrill

Spring 2015

1 Motivation

Algorithmic design is primarily concerned with the generation of experimental designs by computational means. Particular design characteristics are usually desired and these characteristics are often made with respect to a given model:

$$\mathbf{y} = X\beta + \epsilon$$
$$\epsilon_i \stackrel{iid}{\sim} (0, \sigma^2)$$

where this model is considered to have p parameters. A specific design region, R , is usually specified such that the design properties of X are only considered over points $\mathbf{x} \in R$. While a great many characteristics of designs exist (blocking, completeness, balance, orthogonality, confoundedness, etc.) those of present interest concern the variance of predicted values of a model.

Much of what is available in evaluation and comparison of response surface methodology designs as well as computer-generated design were the work of Kiefer (1959, 1961) and Keifer and Wolfowitz (1959). Much of their work uses measure theoretic approach in which a design is viewed in terms of design measure. While this analytic approach has become useful in certain contexts it is of limited use generally. As is common in the statistical sciences, as computing power increased (particularly in the 1970s and 1980s) design optimality became a practical component of the discipline as designs were put forth as being efficient in terms of criteria inspired by Keifer and his coworkers ([2], Section 8.2).

2 Scaled Prediction Variance Function Derivation

Assuming that there are p parameters in the above model, if \mathbf{b} is the ordinary least squares estimate of β then the variance of the predicted values¹ is given by

$$\begin{aligned}\text{Var}[\hat{\mathbf{y}}(\mathbf{x})|X] &= \text{Var}[\mathbf{x}\mathbf{b}] \\ &= \text{Var}[x_1b_1 + x_2b_2 + \cdots + x_pb_p] \\ &= \sum_{i,j} \text{Cov}(x_ib_i, x_jb_j) \\ &= \sum_{i,j} x_i \text{Cov}(b_i, b_j) x_j \\ &= \mathbf{x}' \text{Var}[\mathbf{b}|X] \mathbf{x} \\ &= \mathbf{x}' E[(\mathbf{b} - E(\mathbf{b}))(\mathbf{b} - E(\mathbf{b}))'|X] \mathbf{x} \\ &= \mathbf{x}' E[(\mathbf{b} - \beta)(\mathbf{b} - \beta)'|X] \mathbf{x}\end{aligned}$$

Note that $\mathbf{b} = (X'X)^{-1}X'\mathbf{y} = (X'X)^{-1}X'(X\beta + \epsilon) = \beta + (X'X)^{-1}X'\epsilon$

$$\begin{aligned}&= \mathbf{x}' E[(\beta + (X'X)^{-1}X'\epsilon - \beta)(\beta + (X'X)^{-1}X'\epsilon - \beta)'|X] \mathbf{x} \\ &= \mathbf{x}' E[(X'X)^{-1}X'\epsilon)((X'X)^{-1}X'\epsilon)'|X] \mathbf{x} \\ &= \mathbf{x}' E[(X'X)^{-1}X'\epsilon\epsilon'X(X'X)^{-1}|X] \mathbf{x} \\ &= \mathbf{x}'(X'X)^{-1}X'E[\epsilon\epsilon'|X]X(X'X)^{-1}\mathbf{x}\end{aligned}$$

¹See http://en.wikipedia.org/wiki/Mean_and_predicted_response

Assuming that $\epsilon \stackrel{iid}{\sim} N(0, \sigma^2)$,

$$\begin{aligned} &= \mathbf{x}'(X'X)^{-1}X'\sigma^2IX(X'X)^{-1}\mathbf{x} \\ &= \mathbf{x}'\sigma(X'X)^{-1}\mathbf{x} \end{aligned}$$

To be able to make comparisons between designs of different size, the “Scaled Prediction Variance” (SPV) function will be used. All things being equal, designs of larger sizes will have smaller prediction variances and multiplying by the design size provides a measure of prediction variance on a per-observation basis. Also, dividing by σ^2 makes the SPV scale-free. Thus we have

$$SPV_X(\mathbf{x}) = \frac{N\text{Var}[\hat{\mathbf{y}}(\mathbf{x})]}{\sigma^2} = N\mathbf{x}'(X'X)^{-1}\mathbf{x}$$

Because this function provides a penalty for an increased sample size, cost comparisons between designs can be made ([2], Section 7.4.4). Moreover, division by σ^2 means that the SPV function is *not* a function of σ and therefore it isn't necessary for data to have been collected to estimate $\hat{\sigma} = \frac{SSE}{n-p}$.

3 Criteria

Naturally, a desirable property of a design would be low levels of prediction variance. Therefore, by taking note of the form of the SPV function, we seek to minimize $(X'X)^{-1}$ in some way (assuming the design region and design size are fixed). There are several ways, referred to as design criteria, in which this can be achieved that are supported in the `OptimalDesign` package:

3.1 D Criterion

Minimize over all designs X

$$\text{D Criterion} = |(X'X)^{-1}|$$

This minimizes the volume of the confidence hyperellipsoid for parameter estimates. This is because $|(X'X)^{-1}|$ is proportional to the square of the volume of the confidence hyperellipsoid.

3.2 A Criterion

Minimize over all designs X

$$\text{A Criterion} = \text{tr}((X'X)^{-1})$$

Minimizing the A Criterion results in minimizing the average variance of the estimates of the regression coefficients.

3.3 I Criterion

Minimize over all designs X

$$\text{I Criterion} = \int_{\mathbf{x} \in R} SPV_X(\mathbf{x}) d\mathbf{x}$$

Minimizing the I criterion results in minimizing the average prediction variance over the entire design region R .

3.4 G Criterion

Minimize over all designs X

$$\text{G Criterion} = \max_{\mathbf{x} \in R} SPV_X(\mathbf{x})$$

That is, minimize the maximum prediction variance over the entire design region R .

4 Search Algorithms

There are several search algorithms for finding optimal designs. Examples include the Sequential Search, Exchange, DETMAX, and Fedorov algorithms which are the searches supported by the OPTTEX procedure in SAS. Currently only a modifiable Fedorov algorithm is supported in `OptimalDesign`. Support for a design search utilizing a genetic algorithm is currently under construction (May 2015).

4.1 Fedorov Algorithm (Deluxe Monte Carlo Edition)

The Fedorov Algorithm takes an initial design and then searches for a point over the design region, R to replace one of the design points. The point chosen to be added to the design is the point that results in the largest decrease in the design criterion. `OptimalDesign` does not use the full Fedorov Algorithm as it is slow with current computing power. To speed things up, `OptimalDesign` selects a random point in the design region to potentially add to the design. If this exchange results in lowering the search criterion, then the exchange is implemented and another, criterion-decreasing exchange is searched for. This algorithm is essentially the 'Monte Carlo' algorithm used in the R package `AlgDesign`.

To compute the D, A, I, and G criteria directly, a matrix inversion is required. There are, however, more efficient means of calculating the change to the criteria, given a particular exchange. The methods for the D, A, and I criteria are from identities provided in the `AlgDesign` documentation [3].

4.1.1 D Criterion

To more efficiently calculate the D criterion, the Woodbury Matrix Identity can be used: let A be a p -by- p matrix, U is p -by- n , C is n -by- n , V is n -by- p . Then

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

Let Y be the model matrix of a design after the replacement of design point \mathbf{x} with candidate point \mathbf{y} . $(Y'Y)^{-1}$ can be computed in a computationally simple manner by utilization of the above identity. If $(X'X)^{-1}$ is known,

$$\begin{aligned} (Y'Y)^{-1} &= ((X'X) + [\mathbf{x}, \mathbf{y}]I_2[-\mathbf{x}, \mathbf{y}]')^{-1} \\ &= (X'X)^{-1} - (X'X)^{-1}[\mathbf{x}, \mathbf{y}](I_2 + [-\mathbf{x}, \mathbf{y}]'(X'X)^{-1}[\mathbf{x}, \mathbf{y}])^{-1}[-\mathbf{x}, \mathbf{y}]'(X'X)^{-1} \end{aligned}$$

and the change in the D criterion is

$$\begin{aligned} \Delta_D &= (X'X)^{-1} - (Y'Y)^{-1} \\ &= (X'X)^{-1}[\mathbf{x}, \mathbf{y}](I_2 + [-\mathbf{x}, \mathbf{y}]'(X'X)^{-1}[\mathbf{x}, \mathbf{y}])^{-1}[-\mathbf{x}, \mathbf{y}]'(X'X)^{-1} \end{aligned}$$

Let $d_{uv} = u'(X'X)^{-1}$. It can be shown that

$$\Delta_D = d_{\mathbf{y}\mathbf{y}} - d_{\mathbf{y}\mathbf{y}} \cdot d_{\mathbf{x}\mathbf{x}} + d_{\mathbf{y}\mathbf{x}}^2 - d_{\mathbf{x}\mathbf{x}}$$

which is a computationally cheap calculation.

4.1.2 A Criterion

It should be noted that the trace of a matrix is linear. Thus, for a design X , the A criterion is a linear functional of the dispersion matrix $(X'X)^{-1}$:

$$\text{tr}((Y'Y)^{-1}) = \text{tr}((X'X)^{-1}) - \text{tr}((X'X)^{-1}[\mathbf{x}, \mathbf{y}](I_2 + [-\mathbf{x}, \mathbf{y}]'(X'X)^{-1}[\mathbf{x}, \mathbf{y}])^{-1}[-\mathbf{x}, \mathbf{y}]'(X'X)^{-1})$$

and, similarly to that above for the D criterion, the change in the A criterion is

$$\begin{aligned}\Delta_A &= \text{tr}((X'X)^{-1}) - \text{tr}((Y'Y)^{-1}) \\ &= \text{tr}((X'X)^{-1}[\mathbf{x}, \mathbf{y}](I_2 + [-\mathbf{x}, \mathbf{y}]'(X'X)^{-1}[\mathbf{x}, \mathbf{y}])^{-1}[-\mathbf{x}, \mathbf{y}]'(X'X)^{-1})\end{aligned}$$

It can be shown (laboriously) that if $\phi_{uv}^A = \text{tr}((X'X)^{-1}uv'(X'X)^{-1})$ then

$$\Delta_A = \frac{(1 - d_{xx})\phi_{xx}^A + d_{yx}(\phi_{xy}^A + \phi_{yx}^A) - (1 + d_{yy})\phi_{xx}^A}{1 + \Delta_D}$$

4.1.3 I Criterion

Because integration is also a linear functional then there should be no surprise that there is a identity for the I Criterion similar to that for the A criterion. Let X_c denote the matrix containing the N set of candidate points approximating the design space R . If

$$B = \frac{1}{N}X_c'X_c, \quad \text{and} \quad \phi_{uv}^I = v'(X'X)^{-1}B(X'X)^{-1}u,$$

then it can be show (laboriously) that,

$$\Delta_I = \frac{(1 - d_{xx})\phi_{xx}^I + d_{yx}(\phi_{xy}^I + \phi_{yx}^I) - (1 + d_{yy})\phi_{xx}^I}{1 + \Delta_D}$$

4.1.4 G Criterion

The G criterion is currently the most difficult criterion to calculate. This is because it requires a calculation of the SPV over the entire set of candidate points. To illustrate the difficulty, consider a search for a 5 factor, linear design (with no interaction). If for each factor we consider 21 different levels (-1 to 1 at .1 increments), then the candidate space contains $21^5 = 4084101$ points. For each iteration in the Fedorov search a matrix multiplication involving a matrix of this size will be required, slowing the search immensely.

The G criterion is relatively unstable as a function of different designs. That is, a small change in the design can lead to large changes in the G criterion. This implies that large candidate sets are required to find G-optimal designs and it's not practical to search for G-optimal designs in this manner ([1], page 777).

The best that one can do is:

Let X denote a potentially new design matrix. Let X_c denote the candidate set matrix. Let X have singular value decomposition (SVD) $X = UDV'$ and X_c have SVD $X_c = U_cD_cV_c'$. We have

$$SVD_{thing} = U_cVD^{-1}$$

The 'leverage' of each point in the candidate space is the corresponding row-norm (Frobenius norm) in of the SVD_{thing} matrix. These leverages are the SPV value at each respective point. The maximum of these leverages are found and this is the G criterion of the potentially new design. If the new design has a lower G criterion, then the old design is exchanged for the new.

While this is more efficient than computing the SVD for each and every candidate point, it's not that much more efficient because the real bottle neck is the size of the candidate set which is not reduced by this technique.

One possible way to speed up this calculation is using an "approximation" of the D matrix in the SVD for X . Often, several singular values are near-zero and could be set to zero. While this could speed up the computation, it would only result in a doubling of the speed at a maximum, which is not a solution to the inefficient calculation.

4.2 Genetic Algorithm

Under construction...

References

- [1] *SAS OnlineDoc: Version 8*, 2015.
- [2] Raymond Myers, Douglas Montgomery, and Christine Anderson-Cook. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. Wiley, 2009.
- [3] Robert E. Wheeler. *Comments on Algorithmic Design*, 2009.