2010 International Symposium on Nonlinear Theory and its Applications
NOLTA2010, Krakow, Poland, September 5-8, 2010

Rigorous parameter estimation for noisy mixed-effects models

Alexander Danis† Andrew Hooker‡ and Warwick Tucker†

†Dept. of Mathematics, Uppsala University
Box 480, 751 06 Uppsala, Sweden
‡Department of Pharmaceutical Biosciences, Uppsala University
Box 591, 751 24 Uppsala, Sweden
Email: alexander.danis@math.uu.se, andrew.hooker@farmbio.uu.se, warwick.tucker@math.uu.se

Abstract—We describe how constraint propagation techniques can be used to reliably reconstruct model parameters from noisy data. The main algorithm combines a branch and bound procedure with a data inflation step; it is robust and insensitive to noise. The set–valued results are transformed into point clouds, after which statistical properties can be retrieved. We apply the presented method to a mixed-effects model.

1. Introduction

Finitely parameterized mathematical models are used to describe, explain, summarize, and predict the behavior of physical, biological, and economical systems. A parameter estimation problem is a problem of finding a set of parameter values that makes the model function fit the experimental data. Incomplete approaches to this problem search for one solution in parameter space; complete approaches search for all. In this article, we describe a complete approach – a general-purpose solution strategy based on set–valued computations and global search algorithms that operate reliably under noise. We begin by describing the basic components of the solver which involves set–valued computations, directed acyclic graphs, constraint propagation, and data inflation. We end the article with an example demonstrating the usefulness of our approach.

2. Interval analysis

The foundation of most computer-aided proofs dealing with continuous problems is the ability to compute with set-valued functions. This not only allows for all rounding errors to be taken into account, but – more importantly – all discretization errors too. Here, we will briefly describe the fundamentals of interval analysis. For a concise reference on this topic, see e.g. [Moo66, Neu90].

Let $\mathbb{R}$ denote the set of closed intervals. For any element $\boldsymbol{a} \in \mathbb{R}$, we adopt the notation $\boldsymbol{a} = [\underline{a}, \overline{a}]$, where $\underline{a}, \overline{a} \in \mathbb{R}$. If $\star$ is one of the operators $+, -, \times, \div$, we define the arithmetic on elements of $\mathbb{R}$ by

$$\boldsymbol{a} \star \boldsymbol{b} = \{a \star b \colon a \in \boldsymbol{a}, b \in \boldsymbol{b}\}, \tag{1}$$

except that $\boldsymbol{a} \div \boldsymbol{b}$ is undefined if $0 \in \boldsymbol{b}$. Working exclusively with closed intervals, the resulting interval can be expressed in terms of the endpoints of the operands. Note that the identities (1) reduce to ordinary real arithmetic when the intervals are *thin*, i.e., when $\underline{a} = \overline{a}$ and $\underline{b} = \overline{b}$.

A key feature of interval arithmetic is that it is *inclusion monotonic*, i.e., if $\boldsymbol{a} \subseteq \boldsymbol{x}$, and $\boldsymbol{b} \subseteq \boldsymbol{y}$, then

$$\boldsymbol{a} \star \boldsymbol{b} \subseteq \boldsymbol{x} \star \boldsymbol{y}, \tag{2}$$

where we demand that $0 \notin \boldsymbol{y}$ for division.

One of the main reasons for passing to interval arithmetic is that this approach provides a simple way of enclosing the range of a function f, defined by $R(f; D) := \{f(x) \colon x \in D\}$. Except for the most trivial cases, classical mathematics provides few tools to accurately bound the range of a function. To achieve this latter goal, we extend the real functions to *interval functions* which take and return intervals rather than real numbers. Based on (1) we extend rational functions to their interval versions by simply substituting all occurrences of the real variable x with the interval variable $\boldsymbol{x}$ (and the real arithmetic operators with their interval counterparts). This produces a rational interval function $F(\boldsymbol{x})$, called the *natural interval extension* of f. As long as no singularities are encountered, we have the inclusion

$$R(f; \boldsymbol{x}) \subseteq F(\boldsymbol{x}), \tag{3}$$

by property (2). In fact, this type of range enclosure can be achieved for any reasonable function. A higher-dimensional function $f \colon \mathbb{R}^n \to \mathbb{R}$ can be extended to an interval function $F \colon \mathbb{R}^n \to \mathbb{R}$ in a similar manner.

There exist several open source programming packages for interval analysis, as well as commercial products.

We will now illustrate the use of interval techniques, with a special emphasis on parameter estimation problems.

Example 2.1 *Consider the model $y = f(x; p) = xe^{-px}$, together with the (exact) data point $(x, y) = (2, 1)$, and search region $\boldsymbol{p} = [0, 1]$. A straight-forward interval evaluation of the model function yields:*

$$f(x; \boldsymbol{p}) = f(2; [0, 1]) = 2e^{-[0,1]\times 2} = [2e^{-2}, 2]. \tag{4}$$

This constrains (at $x = 2$) the value of the model function ($y = 1$) to belong to the interval $[2e^{-2}, 2] \approx [0.27, 2]$, which it does. Had we chosen $\boldsymbol{p} = [1, 2]$ as our search

space, we would obtain an inconsistency: $1 \notin f(2,[1,2]) = [2e^{-4}, 2e^{-2}] \approx [0.03, 0.27]$. *This would allow us to discard the entire set* $\boldsymbol{p}$.

This example illustrates how a *divide and conquer* approach can be devised. Starting from a large search space $\boldsymbol{p}$, we adaptively bisect $\boldsymbol{p}$ into smaller subsets, many of which we can discard via inconsistency checks.

3. Noise and data inflation

Given a model function $y = f(x; p)$ together with a finite set of noisy data $(x_1, \tilde{y}_1), \ldots, (x_N, \tilde{y}_N)$, we will attempt to find parameters that make the model *consistent* with the data, i.e., we want to find the set

$$\mathcal{S} = \{p \in \boldsymbol{p} \colon\ f(x_i; p) = \tilde{y}_i \text{ for all } i = 1, \ldots, N\}. \qquad (5)$$

In general, this set will be empty, indicating the presence of noise in the data. In order to improve matters, we inflate each data value into an interval. This can be done in several ways; assigning a width roughly proportional to the value $\tilde{y}_i$ is a good heuristic choice in many situations:

$$\tilde{y}_i \mapsto \boldsymbol{y}_i = \tilde{y}_i(1 + \alpha[-1, +1]) + \beta[-1, +1]. \qquad (6)$$

Here α is a scaling factor, and β is a threshold factor, necessary for situations when $|\tilde{y}_i|$ is very small.

The new requirement for consistency is now formulated in terms of more robust inclusion conditions:

$$\mathcal{S} = \{p \in \boldsymbol{p} \colon\ f(x_i; p) \in \boldsymbol{y}_i \text{ for all } i = 1, \ldots, N\}. \qquad (7)$$

With no data inflation, this reduces to (5). Gradually increasing α (and/or β) will eventually produce a non-empty set of consistent parameters $\mathcal{S}$.

Given a partition $\mathcal{P} = \{\boldsymbol{p}_i\}_{i=1}^{K}$ of the search space $\boldsymbol{p} = \boldsymbol{p}_1 \cup \cdots \cup \boldsymbol{p}_K$, the consistent parameters can be enclosed via $\underline{\mathcal{S}} \subset \mathcal{S} \subset \overline{\mathcal{S}}$, where

$$\begin{aligned} \underline{\mathcal{S}} &= \{\boldsymbol{p}_j \in \mathcal{P} \colon\ f(x_i; \boldsymbol{p}_j) \subset \boldsymbol{y}_i \text{ for all } i = 1, \ldots, N\}, \\ \overline{\mathcal{S}} &= \{\boldsymbol{p}_j \in \mathcal{P} \colon\ f(x_i; \boldsymbol{p}_j) \cap \boldsymbol{y}_i \neq \emptyset \text{ for all } i = 1, \ldots, N\}. \end{aligned}$$

The next example displays how interval-valued data can be contracted, using constraints from both the model function and the search space.

Example 3.1 *Repeating the calculations from Example 2.1, with* $\boldsymbol{p} = [0, 1]$, *but with the data* $(x, \boldsymbol{y}) = (2, [1, 3])$, *we can contract the data range according to*

$$\boldsymbol{y} \mapsto \boldsymbol{y} \cap f(x; \boldsymbol{p}) = [1, 3] \cap [2e^{-2}, 2] = [1, 2].$$

4. Constraint propagation for pedestrians

In this section, we will outline the main ingredients of our parameter estimation procedure. As mentioned in Section 1, our method is global. As such, it attempts to find *all* parameters consistent with the data. Of course, when the data is noisy, there are no consistent parameters, in general. This forces us to inflate the data – a process that compensates for the loss of information caused by the noise. Once sufficiently inflated, the set of consistent parameters is non-empty and bounded. We shrink the bounding set (and the inflated data) by constraint propagation techniques. To be fully effective, these techniques require that the model function be represented in a special form.

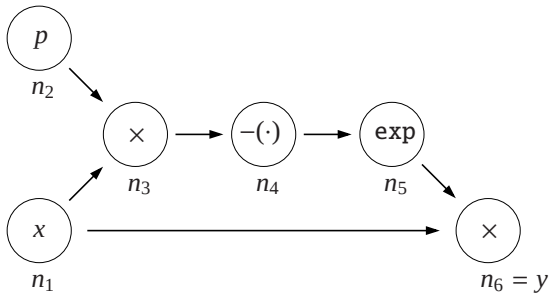

Figure 1: The DAG representation of a forward sweep of $y = xe^{-px}$.

4.1. The DAG representation

We use a directed acyclic graph (DAG) representation of the model function to automate constraint propagations. This representation captures the natural way of decomposing a (possibly complicated) function into its basic building blocks. The graph nodes represent variables, constants or simple functions, while the edges represent dependencies between them.

Example 4.1 *Returning to the model function of Example 2.1,* $f(x; p) = xe^{-px}$, *it can be decomposed into the following code list:*

$$\begin{aligned} n_1 &= x \\ n_2 &= p \\ n_3 &= n_1 \times n_2 \\ n_4 &= -n_3 \\ n_5 &= e^{n_4} \\ n_6 &= n_1 \times n_5. \end{aligned}$$

This list is equivalent to the DAG illustrated in Figure 1.

The DAG representation is used to obtain numeric and symbolic information about various mathematical objects, such as derivatives, slopes, mean value forms, linear or convex relaxations (over- and under-estimators), and convexity information. Representations of equations and inequalities can be included for the purpose of accelerating the constraint propagations. Information on these matters can be found in [GW08, SN05].

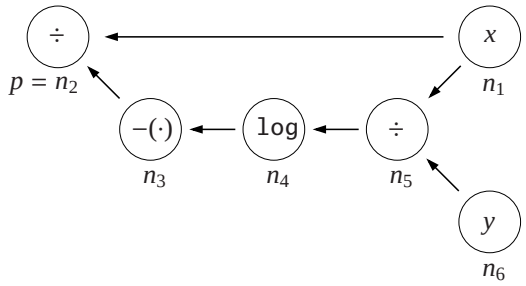

Figure 2: The DAG representation of a backward sweep of $y = xe^{-px}$.

4.2. Constraint propagation on DAGs

To each elementary mathematical operation one associates two operations, *forward* and a *backward* operations. The forward operator evaluates the range based on the range of its arguments and intersect it with the current range. The backward operator evaluates the ranges of its predecessors and intersect it with their current ranges.

Example 4.2 *Once more, we will use the model function of Example 2.1, $f(x;p) = xe^{-px}$. As we saw in Example 4.1, it can be decomposed into a code list as well as a DAG. We will now show how we can use these objects to propagate constraints from data to the parameter. Moving backwards in the code list of Example 4.1, starting from $(x,y) = (n_1, n_6)$, and ending in $p = n_2$, we obtain a new code list:*

$$\begin{aligned} n_5 &= n_6 \div n_1 \\ n_4 &= \log n_5 \\ n_3 &= -n_4 \\ n_2 &= n_3 \div n_1. \end{aligned}$$

This list is equivalent to the DAG illustrated in Figure 2.

Thus, viewing a function in terms of its code list or DAG allows us to compute its formal inverses, without knowing the formulae for these. All we need is the code list for f. Traversing the list backward produces the desired information. This is extremely useful for parameter estimation problems, as we illustrate in the following example.

Example 4.3 *Given the model function $y = f(x;p) = xe^{-px}$, together with the data $(x,y) = (2,1)$, we can generate, and evaluate the code list of Example 4.2:*

$$\begin{array}{rclcl} n_5 &=& n_6 \div n_1 &=& 1 \div 2 \\ n_4 &=& \log n_5 &=& \log \frac{1}{2} \\ n_3 &=& -n_4 &=& \log 2 \\ n_2 &=& n_3 \div n_1 &=& \frac{1}{2}\log 2 \approx 0.34657359. \end{array}$$

The conclusion is that the only parameter that corresponds to the data $(x,y) = (2,1)$ is $p = \frac{1}{2}\log 2$. Of course, for this simple example, we can find the explicit inverse: $p = \frac{1}{x}\log\frac{x}{y}$, which gives the sought result. The point is, however, that we never use this formula; we only use the code list of f.

Continuing Example 3.1, where we have interval-valued data $(x, \boldsymbol{y})$, and a parameter domain $\boldsymbol{p}$ to examine, we can combine the forward and backward sweeps to contract both $\boldsymbol{y}$ and $\boldsymbol{p}$.

Example 4.4 *Again, we work on the model function $y = f(x;p) = xe^{-px}$, but now with the data $(x, \boldsymbol{y}) = (2, [1,3])$, together with the parameter domain $\boldsymbol{p} = [0,1]$. The forward sweep, performed in Example 3.1, contracts the interval data to $\boldsymbol{y} = [1,2]$. Performing a backward sweep, as in Example 4.3, contracts the interval parameter to $\boldsymbol{p} = [0, \frac{1}{2}\log 2]$:*

$$\begin{array}{rclcl} n_5 &=& n_6 \div n_1 &=& [1,2] \div 2 \\ n_4 &=& \log n_5 &=& \log[\frac{1}{2},1] \\ n_3 &=& -n_4 &=& [0, \log 2] \\ n_2 &=& n_3 \div n_1 &=& \frac{1}{2}[0,\log 2] \end{array} \quad \begin{array}{rl} = & [\frac{1}{2},1] \\ = & [-\log 2, 0] \\ & \\ \approx & [0, 0.34657359]. \end{array}$$

Note that, in one forward/backward sweep, we managed to exclude over 65% *of the parameter domain, at the same time reducing the data uncertainty by* 50%*.*

In most cases, the described constraint propagation techniques do not result in a complete contraction to the optimal state. Rather, a stage is reached where no further contraction can be obtained, even though there are inconsistencies present. In order to proceed, some type of partitioning must be employed. The partitioning can be performed at any node of the DAG. Once a node has been selected for partitioning, its domain is split, resulting in two new DAGs, differing only in the domain of the split node. Each DAG is updated through forward/backward sweeps, possibly generating more contraction.

4.3. Data gridding

In order to extract traditional statistics from the set-valued results, we discretize the set $\overline{\mathcal{S}}$ into a collection of points. Recall the the outcome of our parameter estimation is a collection of boxes $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$, whose union $\overline{\mathcal{S}}$ may or may not be a connected set. We form the hull of this collection by taking the smallest box $\hat{\boldsymbol{p}}$ that contains $\overline{\mathcal{S}}$. Next, we introduce m equally spaced nodes along each side of $\hat{\boldsymbol{p}}$. This defines a grid of size m^d where d is the dimension of $\hat{\boldsymbol{p}}$. From this grid, we discard all nodes that are not sufficiently close to the set $\overline{\mathcal{S}}$. This leaves us with a set of points amenable to statistical tests.

5. Methods and examples

We demonstrate our method on a mixed–effects model. A *mixed–effects* model is a model that includes a mixture

of fixed and random factors. In the current setting, we will consider a model function of the form $f(t, \vec{p})$, where t denotes time, and $\vec{p} = (p_1, p_2, p_3)$ is a three-dimensional parameter vector. A *population* parameter is a parameter that is shared among every member of the population. An *individual* parameter is a parameter that is unique to each individual. In what follows, we let p_1 be an individual parameter, whereas p_2 and p_3 act as population parameters. Thus p_1 corresponds to a random factor (sampled from some underlying distribution), whereas p_2 and p_3 correspond to fixed factors.

5.1. Methods

Starting from a given parameter vector $\vec{p} = (p_1, p_2, p_3)$, we perturb the individual parameter according to

$$p_1^i = p_1 + \eta_i \quad \text{where} \quad \eta_i \sim N(0, \sigma^2) \qquad (i = 1, \ldots, N_p). \tag{8}$$

Here $N(a, b)$ denotes the normal distribution with mean a and (positive) variance b. This produces N_p different parameter vectors $\vec{p}^1, \ldots, \vec{p}^{N_p}$ (each corresponding to a different subject), where $\vec{p}^i = (p_1^i, p_2, p_3)$. For each of the N_p subjects, we generate exact data $y_{ij} = f(t_j; \vec{p}^i)$ for $j = 1, \ldots, N_d$. Next, the exact data is perturbed, with maximal intensity $\epsilon > 0$, according to

$$\tilde{y}_{ij} = y_{ij}(1 + \theta_{ij}) \quad \text{where} \quad \theta_{ij} \sim U(-\epsilon, +\epsilon). \tag{9}$$

Here $U(a, b)$ denotes the uniform distribution on the interval $[a, b]$. This produces the data set $(t_j, \tilde{y}_{ij})$, which, together with the model function, is all information we have.

In order to find the typical performance of our method, we repeat the entire estimation process $N_t = 200$ times, and report the average results.

5.2. Example

The following model is used to study the growth of orange tree trunks, see [LB90, DS98].

Example 5.1 *Consider the following function*

$$f(t; \vec{p}) = \frac{p_1}{1 + p_2 e^{p_3 t}}. \tag{10}$$

For this specific example, we will use $N_p \in \{10, 25, 50\}$ subjects, sampled at $N_d = 10$ data sites, evenly spaced within $[100, 1600]$. The parameter values corresponding to the mean population are $\vec{p} = (191.84, 8.153, -0.0029)$; the perturbation parameters are $\sigma = 20$ and $\epsilon \in \{0.1, 0.2, 0.3\}$. We perform the constraint propagation over the parameter region $\boldsymbol{p} = ([0, 300], [0, 9], [-1, 0])$. The results of this procedure are illustrated in Figure 3 and in Table 1.

6. Conclusion

We have described a new approach to solve parameter estimation problems for noisy data. It is a deterministic global search method based on relaxation methods (via data inflation) and the use of set–valued constraint propagation.

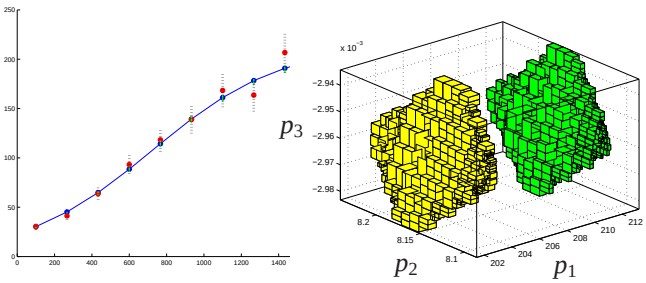

Figure 3: (a) Data inflation and contraction for Example 5.1. The graph of the model function $t \to f(t, \vec{p})$ for one subject (blue line). The data points are marked with red dots. The inflated data sets are shown as striped bars, and the re-contracted data as green bars. (b) The set of consistent parameters for two subjects of Example 5.1.

	$N_p = 10$	$N_p = 25$	$N_p = 50$
$\epsilon = 0.1$	190.266 (20.3)	189.703 (20.1)	189.834 (20.0)
$\epsilon = 0.2$	186.853 (20.4)	185.828 (20.0)	185.740 (20.0)
$\epsilon = 0.3$	185.781 (20.7)	185.851 (20.0)	185.838 (20.0)

Table 1: The results of several experiments for Example 5.1, all using $N_t = 200$ trial runs, $p_1^\sharp = 191.184$, and $\sigma^\sharp = 20.0$. For each pair (ϵ, N_p), we display the pair $\mu(p_1)$ $\mu(\sigma)$ – the average estimates of the distribution parameters for p_1.

References

[DS98] Norman P. Draper and Harry Smith. *Applied Regression Analysis*. John Wiley and Sons, New York, 3rd edition, 1998.

[GW08] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, Philadelphia, PA, 2nd edition, 2008.

[LB90] Mary J. Lindstrom and Douglas M. Bates. Nonlinear mixed effects models for repeated measures data. *Biometrics*, 46(3):673–687, 1990.

[Moo66] Ramon E. Moore. *Interval analysis*. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1966.

[Neu90] Arnold Neumaier. *Interval Methods for Systems of Equations*. Cambridge Univ. Press, 1990.

[SN05] Hermann Schichl and Arnold Neumaier. Interval analysis on directed acyclic graphs for global optimization. *J. of Global Optimization*, 33(4):541–562, 2005.