Engsci 721: Assignment 2a: Least squares estimation for nonlinear models

Oliver Maclaren
oliver.maclaren@auckland.ac.nz

Due: **Sunday 25th September 11.59 pm** (submit via Canvas). This part is 2/2 of your Assignment 2 and worth 5% out of the total 10%.

This part of your Assignment 2, Part 2b, will look at basic parameter estimation in a simple *potentially nonlinear* (in some parameters!) model. For simplicity we will consider relatively stable problems, in the sense that we will look at scenarios with more data than parameters (metaphorically, 'tall', though no longer linear). We will look at constructing both *point* and *interval* estimates – **see Lecture 8 from my (Oliver's) part of the course for more background on various types of estimate (though this is not required to complete the assignment).**

Problems like the one considered here lead to natural generalisations in which parameters are no longer scalar values but e.g. spatially varying *functions*. These problems then require regularisation or related strategies.

[Note: your answers should be given in the form of Python/Matlab etc code and output produced from this, along with any required comments or derivations. You should include your code in your submission.]

Question 1

This question is based on experimental work by Hodgkin and Keynes in the 1950s on giant squid axons. This was closely related to the earlier work by Hodgkin and Huxley that led them to win the 1963 Nobel Prize in Physiology or Medicine. (Confession: I stole the idea for this problem from 'Partial Differential Equations for Scientists and Engineers' by Farlow, which is a great general reference on PDEs.)

So! Suppose a biologist is trying to determine how fast potassium ions diffuse in an exoplasm solution. This helps us understand how nerve impulses are transmitted along axons. The problem is that the diffusion coefficient is essentially impossible to measure directly. What we can do, however, is find a theoretical expression for how the spatiotemporal potassium concentration, denoted by $u(x,t)$, depends on the diffusion coefficent D (assumed constant), via a 'forward model'. We can then solve the corresponding *inverse problem*: measure $u(x,t)$ and estimate D from this.

Our general forward model for this problem will be the advection-diffusion equation on an infinite domain:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} - v\frac{\partial u}{\partial x}, \qquad -\infty < x < \infty$$

with initial condition

$$u(x,0) = A\exp\left(-x^2/a\right), \qquad -\infty < x < \infty$$

where D is the (unknown!) diffusion coefficient, v is the (known) fluid velocity, $u(x,t)$ is the potassium concentration, and A and a are parameters controlling the shape of the initial condition. (The infinite domain allows us to avoid the boundary conditions etc). We will assume that a is known but A is unknown. We will also assume that all these quantities are non-dimensionalised or expressed in appropriate units.

In many inverse problems we need to solve the equation *numerically*, since a simple analytical solution typically doesn't exist. However, it can be shown that the above problem has the analytical solution:

$$u(x,t) = \frac{A\sqrt{a}}{\sqrt{a+4Dt}}\exp\left(\frac{-(x-vt)^2}{a+4Dt}\right),$$

which will make solving our inverse problem slightly easier. Note, however, that we could carry out the same steps below when $u(x,t)$ is obtained from a PDE solver rather than analytically!

Our primary goal here is to estimate the *target parameter* D. As part of this, we need to estimate the so-called *nuisance parameter* A. (This is called a nuisance parameter because we aren't directly interested in its value but we still need to deal with it somehow to get to our main goal.)

While we could aim to estimate all parameters simultaneously, Hodgkin and Keynes used a fairly sensible two-stage process: they first fit the initial condition to data they had available at $t = 0$ at a series of x locations along the axon. They then 'plugged-in' the estimated initial condition parameters to the above equation, giving a function of D only. Finally, they fit this resulting equation to data they collected at a single location, $x = x_0$, for a series of times t. Let's follow them!

a) **Implement** a function (or set of functions) in Python/Matlab/Julia etc that returns the solution $u(x,t)$ at any location x and time t for given a, A, D, v.

b) **Verify** your function results in curves that look like the those in the figure below for both the initial condition setting and solution at fixed x location solution. These settings are: (initial condition) $t = 0$, x a grid of values between -1 and 1, $a = 0.2, A = 10, D = 1.5, v = 1$; and (solution at fixed x location) $x = 0$, t a grid of values between 0 and 1 (other parameters same again).

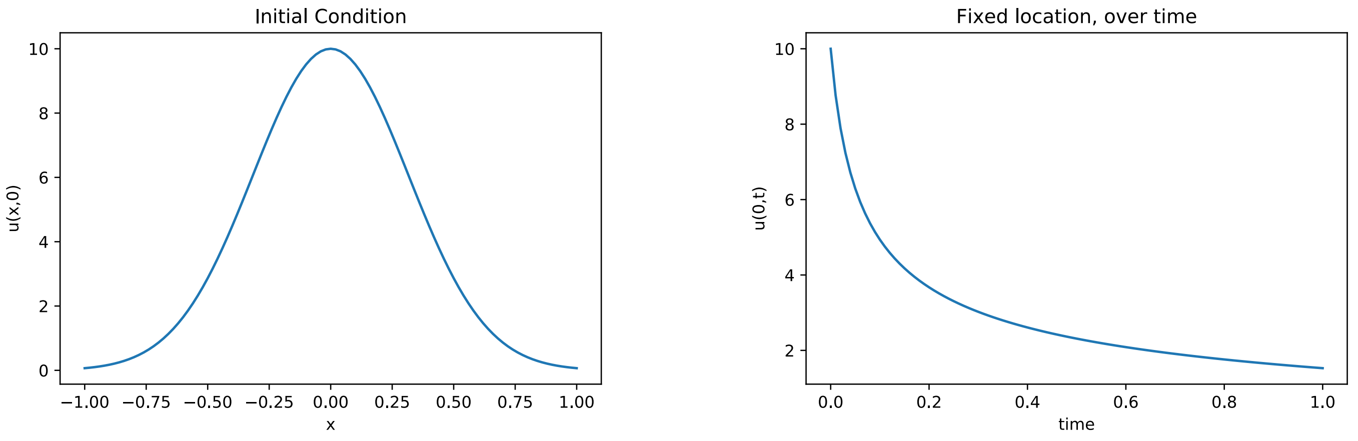

Figure 1: (L) Example initial condition. (R) Example solution for fixed x, varying t.

c) Now let's first assume that we obtain experimental data on the initial condition under the 'statistical model' (see Lecture 8, though this is not necessary for your assignment)

$$u_{\text{obs}}(x,0)_i = u_{\text{obs}}(x_i,0) = A\exp\left(-x_i^2/a\right) + e_i$$

for each observation $i = 1, ..., m$, and for e_i independent, identically distributed random variables representing noise having *known* (and normal) distribution (e.g. based on past experience with the measuring device):

$$e_i \sim \mathcal{N}(0,1).$$

We can consider each e_i as an entry in an error vector e, and similarly for the other observations such as $u_{\text{obs}}(x,0)_i$, where $u_{\text{obs}}(x,0)$ is shorthand for the vector formed from evaluating the function for each x_i value. We assume the vector x, with entries/measurement locations x_i, is fixed and known. First assume our unknown parameter is A. We can think of the above as a statistical model for random $u_{\text{obs}}(x,0)$ vector realisations, i.e. our given data $u_{\text{obs}}(x,0)$ is considered a realisation of the statistical model, and $A \mapsto A\exp\left(-x_i^2/a\right)$ as a deterministic model for the mean.

We are first going to use least squares to get a 'best estimate' of our A parameter before considering our parameter of interest D.

For fixed a and parameter to estimate A, define the sum of squares error function, SSE, in terms of the L_2 norm of the error vector:

$$\text{SSE}(A; u_{\text{obs}}(x,0)) = \|e\|^2 = \|u_{\text{obs}}(x,0) - A\exp\left(-x^2/a\right)\|^2,$$

where as above $\exp\left(-x^2/a\right)$ represents a vector formed from evaluating the function for each x_i in x.

Write down an optimisation problem to solve that would give the (unregularised) least squares solution for A. You do not (yet!) need to solve this problem!

d) Now let's solve the optimisation problem! Use the following measurements to **compute the least squares solution for** A by calling an optimisation library (Python, Matlab etc have optimisation libraries!):

$$x_i = [-0.6, -0.5, -0.3, -0.25, -0.1, 0.0, 0.1, 0.25, 0.3, 0.5, 0.6],$$

$$u_{\text{obs}}(x_i, 0) = [1.78, 2.90, 6.27, 8.06, 10.64, 10.14, 10.00, 6.48, 5.63, 1.82, 0.91]$$

where $a = 0.2, D = 1.5, v = 1$.

e) Next we will assume that A is known and equal to 10, and focus on estimating D from the model for u at a 'fixed location over time,' i.e. $u(x_0, t)$. (Note that in reality A will be estimated using a process like the above and hence our estimates will have additional uncertainty, but we will ignore this for now.) We will take $x_0 = 0$. We will again assume a simple statistical model of the form:

$$u_{\text{obs}}(0, t_i) = u(0, t_i) + e_i, \qquad e_i \sim \mathcal{N}(0, 1).$$

We are going to continue to base our estimation approach on (nonlinear) least squares. Furthermore, we are going to form so-called *confidence intervals* based on some magical approximate/asymptotic results from statistical theory on the 'sampling distribution' of the least squares estimator. To do this we need to compute the *relative* sum of squares function:

$$\text{SSE}_R(D; u_{\text{obs}}(0, t)) = \text{SSE}(D; u_{\text{obs}}) - \min_D\{\text{SSE}(D; u_{\text{obs}})\}$$

where the SSE function, now for the unknown parameter D, is defined as in your minimisation problem for estimating A but now as a function of D and uses $u_{\text{obs}} = u_{\text{obs}}(0, t)$, **the vector of all observations at $x = 0$ evaluated for a vector of known *times* t** (careful to use the right model and data combination!).

Since D is a simple scalar parameter, we can calculate this relative SSE function by **directly evaluating the 'unnormalised' SSE** over a grid of D values for fixed data and other parameters. We can then compute the relative SSE by subtracting off the minimum of the (finite) number of grid values (e.g. by using np.min in Python).

Do this for a grid of 100 D values from 0.1 to 5 and plot the resulting relative SSE function. Use the data:

$$t = [0, 0.01, 0.05, 0.1, 0.15, 0.2, 0.3, 0.35],$$

$$u_{\text{obs}}(0, t) = [10.50, 8.23, 5.40, 4.17, 5.77, 5.61, 3.37, 1.60]$$

f) We can use this relative SSE function to construct *confidence intervals*. These work by taking our *interval estimate* to be the set of all parameters with relative SSE (essentially 'relative fit') within a given threshold:

$$\{D \mid \text{SSE}_R(D; u_{\text{obs}}) \leq c\}$$

for some threshold c.

In order to 'calibrate' these intervals in a 'frequentist' sense (Lecture 8), we need to come up with a way of choosing c so that the 'interval guessing procedure' will trap the true value 95% of the time (regardless of what the true value is!). Long story short (see e.g. Pawitan's 'In All Likelihood' book), for a scalar parameter like D (and measurement standard deviation $\sigma = 1$), an approximate 95% confidence interval procedure based on SSE_R can be constructed by choosing c = 3.84. Magic (...using the Chi-squared distribution...)! Note: for $\sigma \neq 1$ you would need to divide the SSE by σ^2 to apply the same threshold.

Use this approach to **determine the confidence interval for D given the data above**.

g) Next we will investigate the theoretical performance of this general *method* of forming confidence intervals (i.e. our 'confidence interval procedure' – see Lecture 8) by carrying out a *coverage experiment*. In particular, assume that the true value of D is 2. Furthermore, continue to assume $A = 10, a = 0.2, v = 1$, and that we observe noisy versions of $u(0, t_i)$ with noise model $e_i \sim \mathcal{N}(0, 1)$. We will assume the t_i values are the same as above but *generate new noise realisations for each 'trial' in our experiment.*

Use the above to **determine the empirical coverage of the above confidence interval procedure**, i.e. determine an empirical estimate of

$$\mathbb{P}\left([L(u_{\text{obs}}), U(u_{\text{obs}})] \ni D; (D, a, A) = (2, 0.2, 10)\right),$$

where L and U are the lower and upper confidence interval limits determined by the SSE cut-off criterion. Do this by repeatedly generating new data vectors $u_{\text{obs}}(0, t)$, each vector realisation consisting of observations at the same times t_i for $i = 1, .., 8$, under the (fixed) true parameters $(D, a, A) = (2, 0.2, 10)$ and the noise model above, and counting how often the corresponding (random) interval $[L(u_{\text{obs}}), U(u_{\text{obs}})]$ contains the (fixed) true value of D.

Do this for e.g. 1000 or more repetitions, and for each repetition consisting of $m = 8$ measurements with t_i values as above.