Project 1: Solving a linear inverse problem

Technical Computation in the Earth Sciences

21st September 2020

If you've taken any courses involving statistics or data analysis, or read such papers from the literature, you've probably heard terms like "inversion", or "linear regression", or "least-squares estimation". However, these terms are fairly often introduced without explaining them in linear algebra terms. If you're programming in a language where matrices and vectors are well-supported though, linear algebra is probably actually the most convenient and intuitive approach to these topics.

Introduction.1: Matrix multiplication as linear combination

Say we have a column vector x representing some independent variable – be it time, distance, or so on – containing some number of values, for instance:

```
julia> x = 1:10. # This step range will behave like a 10x1 array (i.e., a column vector)
1.0:1.0:10.0
```

We may then consider constructing a matrix A, each column of which contains some different function of x, for example:

```
julia> A = [x.^0 x.^1 x.^2]
10×3 Array{Float64,2}:
 1.0   1.0    1.0
 1.0   2.0    4.0
 1.0   3.0    9.0
 1.0   4.0   16.0
 1.0   5.0   25.0
 1.0   6.0   36.0
 1.0   7.0   49.0
 1.0   8.0   64.0
 1.0   9.0   81.0
 1.0  10.0  100.0
```

Multiplying this matrix A by a column vector of parameters ϕ, for instance

```
julia> ϕ = [1,2,3] # n.b.: [1,2,3] gives you a column vector, [1 2 3] gives you a row vector
3-element Array{Int64,1}:
 1
 2
 3

julia> y = A * ϕ
10-element Array{Float64,1}:
   6.0
  17.0
  34.0
```

```
 57.0
 86.0
121.0
162.0
209.0
262.0
321.0
```

gives us a *linear combination* of the columns of A, in this case equivalent to the function

$$y = 1x^0 + 2x^1 + 3x^3$$

Introduction.2: Inversion

So that's nice and all, but what if we want to go the other way? Start with the independent variable x and dependent variable y and *invert* for the parameters ϕ? Well, if the forward equation is

$$y = A * \phi \tag{1}$$

then we ought to be able to rearrange things a bit to get ϕ on it's own. We have to be a bit careful though, since we're dealing with matrices, and there is strictly speaking no such thing as matrix division, and even matrix multiplication isn't commutative. However, if we could find a matrix A_l^{-1} such that

$$A_l^{-1} * A = \mathbb{1}$$

(in other words, a *left inverse* of A), then we could simply multiply both sides by A_l^{-1}:

$$\begin{aligned} A_l^{-1} * y &= A_l^{-1} * A * \phi \\ A_l^{-1} * y &= \mathbb{1} * \phi \\ A_l^{-1} * y &= \phi \end{aligned}$$

It turns out that as long as the columns of A are *linearly independent*, and A has more rows than columns, then there *is* such a matrix A_l^{-1}, specifically (Equation 11.5 of Boyd & Vandenberghe):

$$A_l^{-1} = (A^T A)^{-1} A^T \tag{2}$$

where A^T is the transpose of A (or conjugate transpose if A were complex). This partiular left-inverse for a (square or tall) matrix with linearly-independent columns is sometimes also called the *pseudo-inverse*. One point to note is that this equation still involves a matrix inversion $(A^T A)^{-1}$, but since $A^T A$ is square, that's comparatively easy. In any case, given equation 2, we can readily calculate this left-inverse in Julia and use it to *invert* for our original ϕ

```
julia> Aᴵ = (A' * A)^-1 * A'
3×10 Array{Float64,2}:
  0.9         0.5          0.183333    -0.05       …   0.0333333    0.3
 -0.304545   -0.125758     0.0113636    0.106818      -0.0409091   -0.195455
  0.0227273   0.00757576  -0.00378788  -0.0113636      0.00757576   0.0227273

julia> Aᴵ * y
3-element Array{Float64,1}:
 0.99999999999946
 2.000000000000071
 2.999999999999992
```

Pretty close, minus some floating-point rounding error. But what if our y isn't entirely accurate? Say ϕ is unknown, and y is coming from noisy experimental observations? It turns out that in this more general case, what $A_l^{-1} * y$ really inverts for is the ϕ that *minimizes* the square of the distance between y and $A * \phi$,

$$\|A * \phi - y\|^2 \tag{3}$$

In other words, $(A^T A)^{-1} A^T y$ gives us the *ordinary least squares* solution for the parameters ϕ (c.f. Equation 12.5 and thereabouts of Boyd & Vandenberghe).

In the event of a less-than-perfect fit, the residual given by Equation 3 may be used as a measure of the misfit of the least-squares solution, though it is more useful if analytical uncertainties are known (which they may not always be).

There are also some generalizations which (though not directly required for the rest of the project) could be useful to know about when working with data with known uncertainties or (even better) covariances:

(1) Weighted least-squares

$$\phi = (A^T W A)^{-1} W A^T y \tag{4}$$

where W is a diagonal matrix of weights equal to the inverse of the squared analytical uncertainties, i.e. `W = diagm(1./σ.^2)` where σ is the (one-sigma) uncertainty of the observations y.

(2) Generalized least squares

$$\phi = (A^T \Omega^{-1} A)^{-1} \Omega^{-1} A^T y \tag{5}$$

where Ω is the covariance matrix of the observations y.

1

Linear inversion with a real dataset: partition coefficients

Petrologists often use mineral/melt *partition coefficients* to model the expected concentrations of trace elements in various minerals. For some trace element El, the partition coefficient is defined as

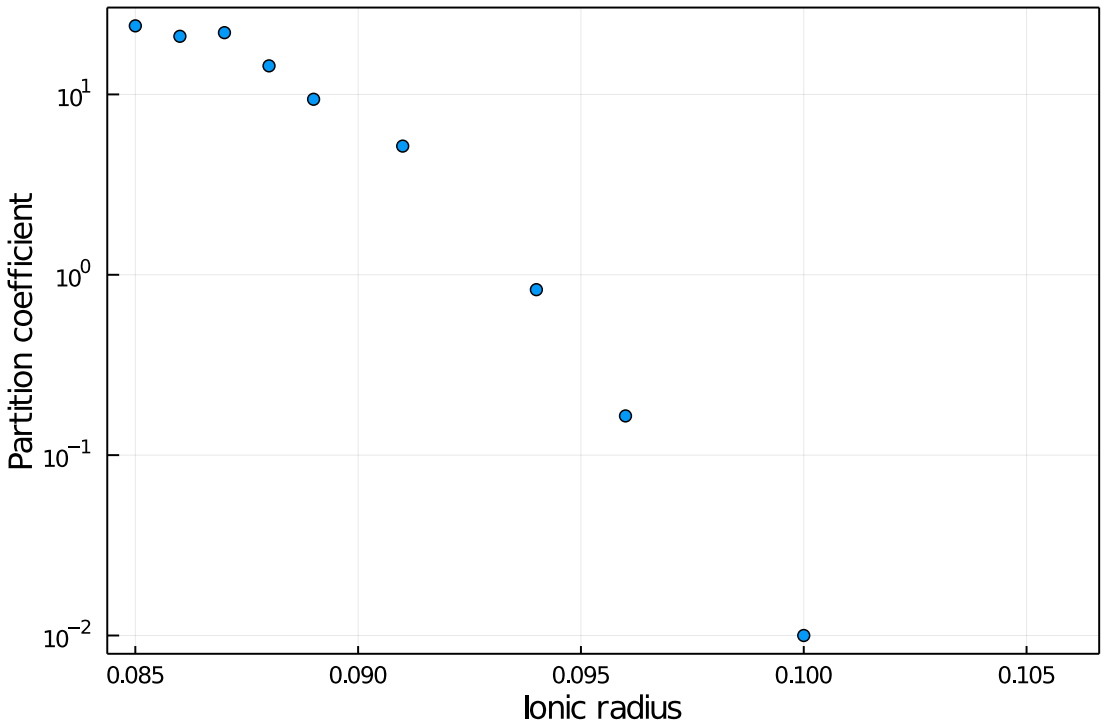
$$D_{El}^{mineral/melt} = \frac{[El]_{mineral}}{[El]_{melt}}$$

the concentration of E in the mineral divided by the concentration of E in the surrounding melt from which that mineral is crystallizing. At equilibrium, this coefficient is approximately constant for a given element in a given mineral, with some dependence on temperature and magma composition. In particular, for a set of trace elements of varying ionic radius but constant ionic charge, mineral/melt partition coefficients are well-modelled at constant temperature as functions of the form $log(D) = a + b * r^2 + c * r^3$ where r is ionic radius.

Consider the following partition coefficient dataset, for the 3+ rare earth elements in the mineral zircon *at 1000 K*, an experimental study by Claiborne et al., 2018.

```
REE3 = ["La","Pr","Nd","Sm","Gd","Tb","Dy","Ho","Er","Tm","Yb","Lu",] # 3+ rare earth elements
r = [0.106,0.101,0.100,0.096,0.094,0.092,0.091,0.089,0.088,0.087,0.086,0.085,] # Ionic Radii
D = [NaN, NaN, 0.01, 0.165, 0.827, NaN, 5.17, 9.4, 14.4, 22.0, 21.0, 24.0] # Partition coefficients
```

```
using Plots
p = plot(r, D, seriestype=:scatter, xlabel="Ionic radius", ylabel="Partition coefficient",
yscale=:log10, label="", framestyle=:box)
savefig(p, "Figure1.pdf")
display(p)
```



Write a program to use linear least-squares inversion to estimate the parameters a, b, c at this temperature, and predict the partition coefficients of the missing elements (the ones for which D is `NaN`). The `isnan` function may be useful, in addition to the material we have talked about above.

Optional: In more detail, such mineral/melt partition coefficients can be related to more physically-meaningful parameters: a mineral- (and magma)-specific D_0, an optimal radius r_0 and an activation energy E (related in turn to bulk modulus), as shown in Equation 2 of Blundy and Wood, 1994. Given the parameters a, b, and c for which we have just inverted, you should be able to solve for D_0,r_0, and E in terms of a, b, and c using ordinary algebra, keeping in mind that $N_A = 6.022e23$ atoms/mol and $R = 8.3145$ J/(mol K). These equations also make it a bit easier to extrapolate observations of partition coefficients at one temperature to other temperatures. *Since this paper gets a bit technical, it's totally optional, but it's good fun if you have time.*

2

Linear inversion with a real dataset: choose-your-own-dataset edition

Pick a real-world dataset of your choice for which linear least-squares inversion can help you determine some parameter of interest. Do the columns of A necessarily have to be powers of x or can we use other functions as well, within some limits?

Be sure to think carefully about, and describe in your report (below):

- Why the function you're inverting describes your dataset
- What assumptions you're making
- Any deeper context for the parameters you've inverted (e.g., anything analagous to the context of the Blundy & Wood paper for the partition coefficient example above)?

Some places to look for relevant data might include:

- https://github.com/awesomedata/awesome-public-datasets (see climate and earth sections)
- https://www.iedadata.org/
- https://data.usgs.gov/datacatalog/

among many others

Make sure you have picked out your real dataset before the start of the next class!

3

Report

In addition to your code, please describe your results and their significance in the format of a brief lab report with one or more figures. While you may collaborate with others while coding (as long as this is clearly noted in comments within your code), the report should be purely your own work.

Due 9/27 by midnight your time