MATH3027: Optimization 2022

Computer Lab 1

Prof. Richard Wilkinson
Please send any comments or mistakes to r.d.wilkinson@nottingham.ac.uk

This module is about the theory and *practice* of optimization. Many (if not most) of the optimization problems you'll encounter in your careers cannot be solved analytically (i.e., by pen and paper). Instead, we have to resort to numerical methods and the power of computers to crunch through large numbers of calculations.

In the computer lab sessions we will look at how to implement some of the methods we study in the lectures. Any new material that is presented here may be relevant to the coursework (which counts for 40% of the module mark), but will not be tested in the exam: the exam will only cover things in the lecture notes.

Choice of programming language

You are free to choose which ever programming language you like for this module. I would recommend you use one of

- R
- Python
- MATLAB

(or possibly Julia if you know what you're doing). R and Python have a great advantage over MATLAB, in that are free *open source* languages. MATLAB is very powerful but expensive - you may not have access to it after you graduate. Generally speaking, R is the dominant language for the field of statistics, whereas Python dominates in machine learning. I will demonstrate in R, and can probably help if you get stuck in Python, but I don't know MATLAB and don't have it installed on my computer. Your choice of language isn't going to matter hugely, as in terms of the calculations we are doing the differences between the languages don't matter much.

We will start the semester by running the computer labs as online sessions, as this allows me to demonstrate code and teach concepts more easily than if you are spread across twi huge computer rooms. If the majority prefer, we can revert to doing face-2-face sessions with the help of PhD student demonstrators.

Lab 1

If you have forgotten the basics of your chosen language, please go and do an introductory online tutorial, such as this (link) online tutorial for R. You will need to know how to

- Do basic vector and matrix calculations, e.g., multiply matrices, find inverses and eigenvalues etc.
- Define a function of one or more inputs
- Write for and while loops.
- Plot some functions.

We haven't yet studied methods for doing optimization on a computer, and so the 'computer lab' this involves quite a bit of pen and paper calculation.

1. Consider the Branin function.

$$f(\mathbf{x}) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)\cos(x_1) + s \text{ for } x_1 \in [-5, 10], x_2 \in [0, 15]$$

The parameters a, b, c, r, s, t are fixed constants, which have default values $a = 1, b = \frac{5.1}{4\pi^2}, c = \frac{5}{\pi}, r = 6, s = 10$ and $t = \frac{1}{8\pi}$.

- Write a function that takes x and the parameters (a, b etc) as an input, and returns the value of $f(\mathbf{x})$. Write the function so that the parameter values are by default set to the values above, but so that they can be changed by the user, i.e., `f(x)` should use the default values above, but `f(x, a=2)` should use $a = 2$, with the other parameters set to their defaults. Make sure the function returns an error if $\mathbf{x}$ is outside of the domain of f.
- Plot the function over its domain.
- Calculate (mathematically) the gradient of the function (with respect to $\mathbf{x}$), and the Hessian matrix.
- We will now use a finite difference approximation to compute the gradients, which can be a useful way to check your calculation above. To approximate the first derivative of a function $g(x)$ we can use

$$g'(x) \approx \frac{g(x+h) - g(x-h)}{2h}$$

for small h. Use this approximation to approximate the gradient of f at $\mathbf{x} = (1, 1)^\top$. What happens as h gets smaller? Plot the error in this approximation vs h, using a log-scale for both axes. What happens if h becomes too small? Why does this happen?

Demonstrate why we prefer this central difference method to the forward difference approximation

$$g'(x) \approx \frac{g(x+h) - g(x)}{h}$$

Note that if $\dim(\mathbf{x}) > 1$ we need to perturb one input variable at a time, e.g.,

$$\frac{\partial g}{\partial x_i}(\mathbf{x}) \approx \frac{g(\mathbf{x} + h\mathbf{e}_i) - g(\mathbf{x} - h\mathbf{e}_i)}{2h}.$$

- The numDeriv package in R has functions to compute numerical estimates of derivatives. For example, the gradient and Hessian. Install this package and use it to check your calculations.
- Use one of the default optimization methods in your language of choice to find the minima of f. In R, you can use the command `optim` (the help page can be found in R using `?optim`).

2. Find the global minimum and maximum points of the function $f(x_1, x_2) = 2x_1 - 3x_2$ over the set $S = \{(x_1, x_2) : 2x_1^2 + 5x_2^2 \leq 1\}$. Write a function (in your chosen programming language) to evaluate f, and create a plot of f and S. If you are unsure of what the set looks like, try entering it into desmos.

   Can you use one of the optimizers (such as `optim` to optimize this function)?

3. For each of the following matrices determine whether they are positive/negative semidefinite/definite or indefinite:

   a)
   $$\mathbf{A} = \begin{pmatrix} 2 & 2 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 1 & 3 \end{pmatrix}$$

   b)
   $$\mathbf{A} = \begin{pmatrix} 2 & 1 & 3 \\ 1 & 2 & 1 \\ 3 & 1 & 2 \end{pmatrix}$$

   c)
   $$\mathbf{A} = \begin{pmatrix} -5 & 1 & 1 \\ 1 & -7 & 1 \\ 1 & 1 & -5 \end{pmatrix}$$

   There is a very simple way to do all this in R (or Python etc). Find it and compare your results.

4. Find all the stationary points of

a)

$$f(x_1, x_2) = x_1^4 + 2x_1^2 x_2 + x_2^2 - 4x_1^2 - 8x_1 - 8x_2\,,$$

b)

$$f(x_1, x_2) = x_1^2 + 4x_1 x_2 + x_2^2 + x_1 - x_2\,,$$

and classify them (saddle points, strict/non-strict local/global max/min). Help your intuition by visualizing the function in R.

5. Find the global minimum of

$$f(x, y, z) = 4x^2 + 3y^2 + 4z^2 + 4xy + 6xz + 4yz + 2x + 3y - z.$$

Check your answer using the 'optim' command in R (or any other optimizer).