Optimisation

Search competition

```
$  import dill
$  with open('path/to/mystery_function.dill', 'rb') as ff:
       mystery_function = dill.load(ff)
$  mystery_function(0.1, 10)
    >   9981.97
```

```
Who can find the input which gives the smallest output?

You have 5 minutes!
```

CAMBRIDGE SPARK

One strategy: guess many points and plot!

```
$  mn = -3
$  mx = 3
$  vals = [1, 10, 100, 500, 1000, 2000]
$  pts = np.linspace(mn, mx, 100)
$  xx, yy = np.meshgrid(pts, pts)
$  contours = plt.contour(xx, yy, mystery_function(xx, yy), vals)
$  plt.clabel(contours, fontsize=9, inline=1)
$  plt.show()
$  idx = np.unravel_index(np.argmin(mystery_function(xx, yy)))
$  xx[idx], yy[idx], mystery_function(xx[idx], yy[idx])
Bonus: 3D surface!
```

CAMBRIDGE SPARK

What would you do if you *knew* the function?

- Suppose that:
  - The function we are minimising has a closed form
  - ...and we actually know the equation
- How would you proceed now?
- Here's the mystery function

```
def mystery_function(x, y):
   a = 1.5
   b = 100
   return (a - x)**2 + b*(y - x**2)**2
```

Can you find the minimum? You have 2 minutes!

CAMBRIDGE SPARK

Differentiation

$$f(x) = x^2$$
$$\frac{\partial f}{\partial x} = 2x$$

$$a = 1.5$$
$$b = 100$$
$$f(x, y) = (a - x)^2 + b(y - x^2)^2$$
$$\frac{\partial f}{\partial y} = 2b(y - x^2)$$
$$\frac{\partial f}{\partial x} = -2(a - x) - 2bx(y - x^2)$$

$$2b(y - x^2) = 0$$
$$\Rightarrow y = x^2$$
$$-2(a - x) - 2bx(y - x^2) = 0$$
$$\Rightarrow 2bx^3 - 2bxy + 2x - 2a = 0$$
$$2bx^3 - 2bx(x^2) + 2x - 2a = 0$$
$$2x - 2a = 0$$
$$\Rightarrow x = a, y = a^2$$

CAMBRIDGE SPARK

What is optimisation?

- Typical situation:
  - We have a function
  - We want to find the minimum
  - We can simply 'solve' -- too many minima anyway!
  - We can evaluate the function for a given input
  - We can might be able to get some other information e.g.
  - the derivative of the function for a given input (i.e. get the gradient at a given point)
- Optimisation is the process of finding the input which produces the minimum
- i.e finding $\theta^* = \text{argmin } f(\theta)$
- Thinking in 3D, imagine a 2D function $h = f(\theta_1, \theta_2)$
- h describes how high you are on the hill given the birds-eye-view coordinate (θ_1, θ_2)
- You've got a blindfold on, but I'm next to you, and will tell you the gradient
- How will you get down?

Gradient descent

- A simple approach is simply to step in the direction of steepest descent…
- ...and keep doing that!*

$$x^{(t+1)} \leftarrow x^{(t)} - \gamma \frac{\partial}{\partial x} f(x^{(t)})$$

$$\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} - \gamma \nabla f(\boldsymbol{x}^{(t)})$$

$$\nabla f(\boldsymbol{x}) = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_D}]$$

$$\gamma = \text{"learning rate" or "step size"}$$

*Not recommended in real life - c.f. Cliffs and human mortality

CAMBRIDGE SPARK

Hands-on session

01-optimisation-skeleton.ipynb
30 mins

Further reading

Khan academy differentaiation introduction:
https://www.khanacademy.org/math/old-differential-calculus#derivative-intro-dc

...lots of textbooks cover calculus in general!