CT5141 Lab Week 11

James McDermott

Solutions in `sol11.pdf`.

Graph layout

1. Try the `graph_layout.py` code for several graphs and look at the plots that are produced during the optimisation. To try out a different graph, just uncomment a line in the `__main__` block.
2. What does a local optimum look like? Can you imagine a locally optimal (not globally optimal) graph layout of (say) 4 nodes? What prevents it from improving?
3. Can you imagine a graph of 3 nodes that results in a local optimum? Could there be a graph of 3 nodes which fails to achieve its desired edge lengths? (This is not the same question.)
4. In the MDS (multi-dimensional scaling) dimensionality reduction algorithm, we have m data points in n dimensions, and we wish to choose m positions for them in 2D such that the distances are preserved. What would we need to do to implement MDS using our `graph_layout.py` code?

Gradient descent with various algorithms

5. The Rastrigin function $An + \sum_i x_i^2 - A\cos(2\pi x_i)$ is a common test function.

We have the code:

```
def rastrigin(x):
    n = 2
    A = 10
    return A*n + np.sum(x**2 - A*np.cos(2*np.pi * x))
```

For $n = 2$, try out each of the following algorithms, inspect the output to understand the minimum it finds and the number of fitness evaluations it used:

- `scipy.optimize.minimize` (gradient descent, finds local optimum only)
- `scipy.optimize.basinhopping` (like iterated hill-climbing, but with gradient descent)
- `cma` (CMA-ES – global search).

Here is some more useful code:

```
!pip install cma

from scipy.optimize import basinhopping
basinhopping(rastrigin, x0)
from scipy.optimize import minimize
minimize(rastrigin, x0)
from cma import CMAEvolutionStrategy
es = cma.CMAEvolutionStrategy(x0, sigma0)
es.optimize(rastrigin)
```

6. If you have plenty of time, try calculating the Jacobian of the Rastrigin objective function on paper or using Sympy.

```
x = sympy.symarray("x", n)
rastrigin = A*n + sum(xi**2 -
     A*sympy.cos(2*sympy.pi * xi) for xi in x)
rastrigin_df_dx0 = sympy.diff(rastrigin, x[0])
```

Run `scipy.optimize.minimize` again, this time passing in the Jacobian. Inspect the output carefully. Has it made a difference?