Section 14: optimization

PHYSICS 91SI

Learning objectives.

(a) Have an idea of why Python is inefficient, and how to avoid some of the worst inefficiencies.

(b) How to profile code to get a non-eyeballed idea of why it's slow.

(c) Know that advanced options exist (Cython, C extensions, Theano) exist if you *really* want them. For a more rewarding example, do the same thing with the Collatz conjecture instead.

The second problem might be a bit more rewarding than the first one.

Part 1. *Factorials.*

The factorial is the product of all positive integers less than n:

$$n! = \prod_{k=1}^{n} k$$

It can be computed in several ways:

(a) Recursive function, using the fact that $n! = n \cdot (n-1)!$.

(b) Direct product for $k = 1, \ldots, n$.

(c) Built-in Python functions: `math.factorial`.

(d) Cythonized version of (b).

(e) C extension for (b).

Play with some of those a bit, especially if you have access to Python or have prior knowledge of C. How does their performance compare? Use IPython's `%timeit` or Unix's `time` to compare them.

Part 2. *Similar bands.*

You're writing a dating website, and you need to come up with some algorithm to find the ideal matches for a given users. Or maybe you're writing an algorithm to find similar bands that sound like a given band. Let's go with the latter.

Your input is a text file with many artists. For every artist a_i, you have a vector of genres $\mathbf{v}_i$, where each component is the intensity of that genre in a given band. For instance, the first component might be rock, the second component might be pop, and the third component might be folk. For Pink Floyd, we would expect this vector to be something like $\mathbf{v}_1 = (0.93, 0.14, 0.08)$. For Creedence Clearwater Revival, it might be $\mathbf{v}_2 = (0.72, 0.05, 0.81)$. We define the similarity between two artists to be

$$s = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{|\mathbf{v}_1||\mathbf{v}_2|}$$

The data is in a text file. It looks like this:

```
Pink Floyd,0.93,0.14,0.08,0.15,0.42,0.31,0.27,0.01 (etc)
The Clash,0.83,0.54,0.18,0.05,0.12,0.13,0.72,0.10 (etc)
```

You will need to parse the data and, for a given artist, compute the similarity between that artist and the rest of them (or, say, get the most similar artist, or the 10 most similar).[1] You are encouraged to create an efficient implementation from scratch, but for simplicity some starter code is provided.

However, that starter code is very inefficient. Think about ways in which you can improve it, especially by using NumPy:

(a) Is there a better, faster way to load the data?

(b) Is there a better way to calculate the similarities?

(c) Perhaps profile the code to see what the slowest parts are?

(d) Are you brave enough to write a C/Cython module for it?

[1] Output for Pink Floyd, for instance, is

```
98% - AC/DC
95% - Aerosmith
95% - Led Zeppelin
95% - U2
95% - Dire Straits
```

which is not ideal but not completely unreasonable, and to be fair we're using a limited sample and model.