SciPy – the embarrassing way to code

vetta.org/2008/05/scipy-the-embarrassing-way-to-code/

By Shane 6 May, 2008

I've programmed in many languages before, indeed I've spent at least a year working in Basic, C, C++, C#, java, assembler, modula-2, powerhouse and prolog. One thing I've never done before is Matlab, well except a few basic exercises for some course I did way back. A couple of years ago I started using python and more recently I've started to use the scipy libraries which essentially provide something similar to Matlab. The experience has been unlike anything I've coded in before. The development cycle has gone like this:

1) Write the code in python like I would write it in, say, java. I have data stored in some places, then I have algorithms that iterate over these data structures computing stuff, calling methods, changing values and doing various complex things in order to implement the desired algorithm. 10 pages of code, somewhat general.

2) Then I realise that in a few places I don't need to iterate over something, I can just use some vectors and work with those directly. 7 pages of code, a little more general.

3) Then I realise that part of my code is really just running an optimisation algorithm, so I can replace it with a call to an optimiser in one of the scipy libraries. 5 pages of code, and a bit faster now.

4) Then I try to further generalise my system and in the process I realise that really what I'm doing is taking a Cartesian space, building a multi-dimensional matrix and then applying some kind of optimiser to the space. 3 pages of code, very general.

5) Finally I'm like, hey, how far can I push this? With some more thought and spending a few days trying to get my head around all the powerful scipy libraries, I finally figure out that the core of my entire algorithm can be implemented in an extremely general and yet fast way in just a few lines. It's really just a matrix with some flexible number of dimensions to which I am applying some kind of n-dimensional filter, followed by an n-dimensional non-linear optimiser on top of an n-dimensional interpolation and finally coordinate mapping back out of the space to produce the end results. 2 pages of code, of which half is comments, over a quarter is trivial supporting stuff like creating the necessary matrices, and just a few lines make the necessary calls to implement the algorithm. And it's all super general.

Now this is great in a sense. You end up throwing away most of your code now that all the real computation work is being done by sophisticated mathematical functions which are using optimised matrix computation libraries. The bottleneck in writing code isn't in the writing of the code, it's in understanding and conceptualising what needs to be done. Once you've done that, i.e. come up with mathematical objects and equations that describe your algorithm, you simply express these in a few lines of scipy and hit go.

It's not just with my financial software either. I recently implemented a certain kind of neural network using nothing but scipy and found that the core of the algorithm was just one line of code — a few matrix transformations and calls to scipy functions. I hear that one of the IDSIA guys working on playing Go recently collapsed the code he's been working on for six months down to two pages.

The downside to all this is that you spend months developing your complex algorithms and when you're done you show somebody the result of all your efforts — a page or two of code. It looks like something that somebody could have written in an afternoon. Even worse, *you* start to suspect that if you had really known scipy and spent a few days carefully thinking about the problem to start with, then you probably *could* have coded it in an afternoon. It's a little embarrassing.