Project Proposal

Andrew Zito

November 24, 2015

1 Genesis

Simulation - capturing an aspect of a real situation and representing it, however simplified, inside a magical electronic box. By observing models of natural phenomena you can, in a sense, create your own data, free from the struggles of trying to observe actual nature. Real nature is complicated, and messy, and it is notoriously difficult to isolate whatever it is you wish to study – not so in simulation. The downside to this is, of course, that your data is inherently less reliable than data obtained from real-world observation. If you make a mistake in your simulation, or leave out variables that you didn't realize were important, you can get results that, while interesting, are unrelated to the actual phenomenon.

Thus, simulation is particularly useful in cases where there is no real-world analog of your phenomenon. Testing hypothetical scenarios becomes much easier when you can actually have them play out in front of you, and see the results. For example, take the so-called "Infinite Monkey Theorem." This is the idea that if you have a bunch of monkeys banging away on typewriters, they will eventually produce every possible combination of letters, including the full works of William Shakespeare. Obviously, this is practically impossible to test. It does serve, however, as an illustration of a property of infinity, and specifically infinite letter combinations. A computer cannot produce an infinitely long string of letters (at least not in a finite amount of time) but it can produce more than enough material to study.

My quest began with these questions: what are the properties of infinite random strings? Will English words appear? How often? What will the relationship be between number of characters and number of words? This quickly sparked a flurry of slapdash coding which brought me a few answers. Yes, words will absolutely appear. Their frequency will increase in a linear relationship to the number of random characters generated. I was somewhat disappointed with, although not enormously surprised by, this result. After further experimentation I discovered that, again unsurprisingly, weighting the chances of each letter by their actual frequency in English boosted the number of words that appeared (although their increase was still linear). This lead me to wonder if it would be possible to evolve the correct frequencies – in essence, to train my monkeys.

2 The Now

My program right now is a basic example of evolutionary computation. It creates a number of Monkey objects, each of which contains letter frequencies, and then alters them over the course of a specified number of generations. All Monkeys start with unweighted letters (where each letter has an equal chance of being selected). The first step in a generation is to mutate these frequencies. Each entry in the list of letter frequencies has a small chance to be increased or decreased by a random number between 0 and 0.1. The Monkey then uses it's current letter frequencies to generate a random string and checks it for English words. It does this several times and stores the average number of words found.

To create the next generation, the program iterates over each Monkey and chooses a parent for it. Monkeys have a higher chance of being parents if they find more words. The parent Monkey's letter frequencies are passed down to it's child Monkey. Finally, all Monkeys are reset by erasing their stored number of average words found. Because of the intensive nature of the computation, the program can take a long time to run. For example, 100 Monkeys take almost 10 minutes to complete 100 generations. These constraints made it helpful to have the program write a sample letter frequency list to a file each generation (specifically, the list belonging to the most successful Monkey of that generation).

Preliminary experiments with this program were not promising, as the letter frequencies produced consistently performed the same as or even worse than unweighted letters. However, after surrendering my laptop for almost 5 hours and letting the program run, it produced frequencies that significantly outperformed unweighted letters.

3 Beyond

First, I want to try giving my evolving Monkeys program to the Hampshire high performance cluster. Based on my small success I think that allowing the program to run for many more generations with many more Monkeys will produce more interesting results, and may eventually even produce frequencies close to the actual English letter frequencies. Even more interestingly, it might produce letter frequencies that perform as well as the real thing, but are significantly different.

I also want to continue exploring this idea in a less evolutionary direction. Right now Monkeys just look for words. Extending this ability to phrases opens up a whole new world of possibilities and problems to explore. How can we determine if something is a valid English phrase? This problem has been explored before, but I think despite this I will learn a great deal by exploring it.