Project 3
Due: 9 Mar 2009

The last project for this course is to write a program that evolves genetic programs. Specifically, genetic programs with numeric inputs and outputs. It’s a group project that you all will contribute to, and I’ll leave the component design largely up to you, though we’ll do a bit of it in class.

1 The basic problems

I’m giving two problem classes here. The first one I know to be solvable this way. The second one I’m less sure about, but I think it can and it would be cool.

1.1 A boring problem

From a list of \((x, y)\) pairs, find an equation that fits them (preferably without overfitting them). To do preliminary testing, you can generate data by picking a suitable \(f(x)\), generating points on that curve, and seeing if the program can figure out what \(f\) is. Later, you can introduce noise into the data (by jittering the points in both dimensions) and see if it’s still learning \(f\) correctly.

1.2 A nifty problem

Elvis the dog likes to play fetch. His owner, standing on a lake shore, throws a ball out into the lake at an angle, and Elvis goes after it. But Elvis knows that he runs faster than he swims, so rather than go straight to it, he runs along the shore for a bit before swimming out. What is the optimal run/swim balance? Assume that your input is a 4-tuple \((x, y, r, s)\), with \(x\) and \(y\) orthogonal distances in meters (\(y\) the distance the ball lands from shore, and \(x\) the distance from Elvis to the shore point closest to the ball) and \(r\) and \(s\) speeds in meters per second (running and swimming respectively). The output is then a number, in meters, of how far to run before swimming.
Even better, if that works: what if Elvis’s owner is already standing in the water? When should Elvis swim the whole way vs swimming to shore, running, and then swimming back out? If you guys get this far, we can talk about an appropriate input/output spec (it’d need to be a bit more complex).

2 Working together

Generally, although I don’t have to get all the minutiae, I’d like to be cc’d on group communications of significance, as when you complete a major section and are explaining its quirks to the others.

I’ve created a subdirectory of the course directory called shared, which you can all put stuff in. Before you create anything in there, you should run “newgrp ai” so that your creation will have the appropriate gid; depending on your umask you may also need to chmod it. I recommend using the shared directory for a version control system; if one of you has a preference, go for it, otherwise I’ll help you set up a Subversion repository there.

3 Receivables

By Monday, 9 March, I’ll expect some final product in the shared directory; it should include all the code I need to run the program, any data files you find appropriate, and a README telling me how to run it all. It may also contain interesting output that you generated at some point along the way. The README file or another file it points to should contain a lab report describing what you tried and what did or didn’t work.

Also, from each of you I want an email by then describing what each person (including yourself) contributed to the project. It doesn’t have to be long, just a few sentences each, but this is how I keep everyone honest and keep the project load balanced.