2005 Summer School in Statistics for Astronomers and Physicists

Practicum on Genetic Algorithms

17 June 2005

1 Genetic Algorithms

1.1 Review

Genetic algorithms (GAs) are stochastic search techniques motivated by the evolutionary model proposed by Darwin. These algorithms incorporate the basic reproductive mechanisms of evolution (selection, crossover, and mutation) within a computational setting. The "survival of the fittest" idea behind GAs is exploited to perform an iterative, multidimensional search for an optimal value of a given objective function. GAs are appealing since they are general and robust, naturally parallel, require no smoothness assumptions on the fitness function or its domain, and can easily locate global maxima in the presence of many local maxima.

1.2 Downloading and Running Pikaia

Pikaia is a free Fortran implementation of a genetic algorithm written by Paul Charbonneau. It can be downloaded from: `www.hao.ucar.edu/Public/models/pikaia/pikaia.html`

Slightly modified versions are available at `www.stat.cmu.edu/∼mbuot/astro-index.html` for today's lab exercises. Please download `pikaia-twod-1.f`, `pikaia-twod-2.f`, `pikaia-twod-3.f`, `pikaia-twod-4.f` and `pikaia-orbit.f`.

To run Piakia, on a PSU Public Lab computer, do the following:

(1) Click on **Start → Programming → The F Compiler → F Compiler Prompt**

(2) In the command line window, change to the directory containing the downloaded files. For example, `cd ''C:\Documents and Settings\ <login id> \Desktop''`.

(3) Create an executable: type `g77 -o twod-1 pikaia-twod-1.f`

(4) Run the exectuable: type `twod-1.exe`. The screen output will summarize the GA, and an output file, `ga-twod-out-1.txt`

The following subsections show the GA in action. NOTE: In all the example programs, the random seed was set to 123456. By changing this value, you can randomize your GA results.

1.2.1 Toy Example

Download and run `pikaia-twod-1.f`. From the screen output, you'll notice a GA was run for 250 generations (or populations), where each generation consists of 4 individuals. The crossover rate is 0.85, and the (constant) mutation rate is 0.0001. The objective function, (function `twod` in the source code), is a 2-dimensional surface, with a unique global maximum, and many local minimum:

Theoretically, the maximum occurs at $(0.5, 0.5)$ and has a value of 1. From the output, you'll see that after 250 generations, the best value found by the GA was $f(0.50295, 0.61015) = 0.9216$

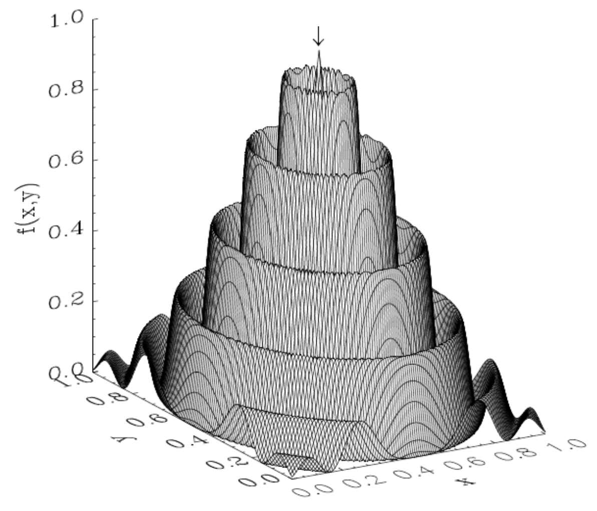

Figure 1: Function `twod`

(this is shown below the `status` line).

To gauge the performance of the GA, an output file named `ga-twod-out-1.txt` is generated. The first column is the generation number, the second column is the fitness value of the optimal genotype (i.e., the value of the function for the best individual of the population), and the third column is the fitness value of the median individual. To plot the generation vs. maximum fitness, we could do the following in R:

```
> gadat1 <- matrix(scan(‘‘ga-twod-out-1.txt’’),ncol=3,byrow=T)
> plot(gadat1[,1],gadat1[,2],ylim=c(0,1),xlab=‘‘Generation’’,
+ ylab=‘‘fitness’’,main=‘‘ga-twd-out-1.txt’’,type=‘‘l’’)
```

Notice that the best individual in the initial generation (a simple random sample four points in $[0,1]\times[0,1]$) has a function value of 0.4194. (You can type `gadat1` at the R prompt to verify this).

What happens when the parameters of a GA are changed? Repeat the steps above for the other files `ga-twod-out-2.txt`, `ga-twod-out-3.txt`, and `ga-twod-out-4.txt`, which are the output files for `pikaia-twod-2.f`, `pikaia-twod-3.f`, and `pikaia-twod-4.f`, respectively. Verify that your plots look like Figure 2.

These exercises illustrate two main observations about GAs: (1) performance is dependent on the choice of parameters, such as population size and mutation rate, and (2) GAs may exhibit “epoch and innovation” behavior, in which the GA may not be able to improve for many generations (an epoch), but then jump to a set of better generations (innovations). For further details about this GA phenomena, please see is “Statistical Dynamics of the Royal Road Genetic Algorithm”, by Nimwegen, Crutchfield, and Mitchell.

Customizing PIKAIA to solve a different optimization problem can be done with relative ease. Essentially, the user writes the Fortran code for the function to be optimized, and makes minor changes in the program driver (`xpkaia`) and subroutines `pikaia` and `setctl`. The following

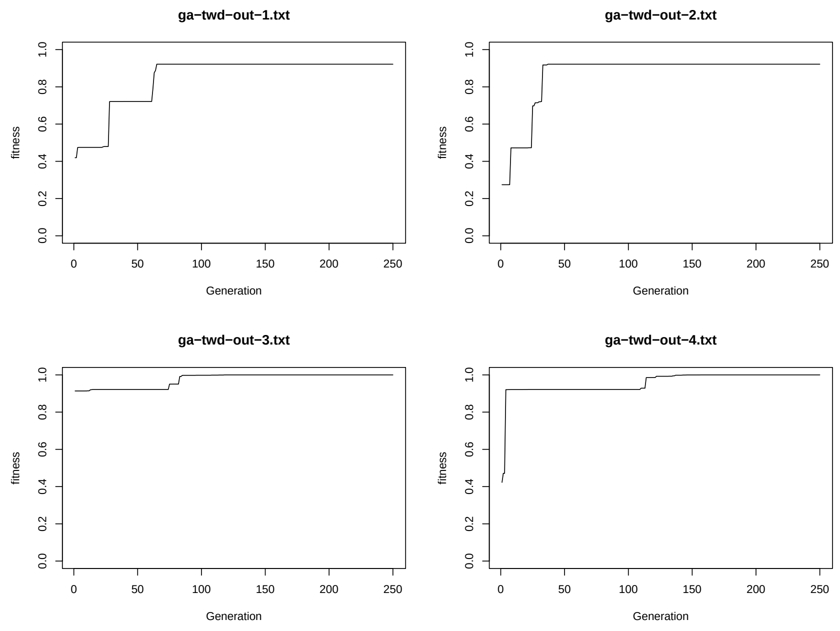

Figure 2: Maximum fitness values over generations.

astronomy example demonstrates this.

1.2.2 Astronomy Example

The following example is taken from a PIKAIA tutorial: section 4, "An Introduction to Genetic Algorithms for Numerical Optimization", by P. Charbonneau.

`www.hao.ucar.edu/Public/models/pikaia/tutorial/tutorial.ps`

Further details regarding the problem described below can be found there.

Suppose we have radial velocity data for the star ρCrB (the 17th brightest star in *Corona Borealis*). In particular, for a given time point (t), we have a measure of radial velocity $(V(t))$, and an estimate of measurement error (σ). It is known that "the radial velocity variation associated with the motion of a binary component in an arbitrarily positioned elliptical orbit about the common center of mass of the system" is

$$V(t) = V_0 + K(cos(\omega + v(t)) + ecos(\omega)) \tag{1}$$

where $V(t)$ is the radial velocity, V_0 is the radial velocity of the center of mass, K is the orbital velocity amplitude, ω is the longitude of the perihelion, $v(t)$ is the true anomaly (angular velocity about the center of mass) and e is the orbital eccentricity. We also know that $v(t)$ and the eccentric anomaly E are related:

$$\tan\frac{v}{2} = \sqrt{\frac{1+e}{1-e}}\tan\frac{E}{2}. \tag{2}$$

Finally, by Kepler's equation

$$E - e\sin(E) = \frac{2\pi}{P}(t - \tau) \tag{3}$$

where P is the orbital period, t is time, τ is the time of perihelion passage.

Using a GA we wish to determine the value of the parameters $(P, \tau, \omega, e, K, V_0)$ which minimizes

$$\chi^2(P, \tau, \omega, e, K, V_0) = \frac{1}{N-6}\sum_{j=1}^{N}\Big(\frac{V_j^{obs} - V(t_j; P, \tau, \omega, e, K, V_0)}{\sigma_j}\Big)^2 \tag{4}$$

where N is the number of observations, V_j^{obs} is the radial velocity observed in the data set, and $V(t_j; P, \tau, \omega, e, K, V_0)$ is the radial velocity at time t_j given the parameters. (To calculate this, we first solve Equation (3) for E, then solve Equation (2) for v, and finally solve Equation (1) for $V(t_j; P, \tau, \omega, e, K, V_0)$, which can be thought of as the "predicted" radial velocity. Hence, Equation (4) is a measure of how far the predicted velocities are from the observed.

The Pikaia implementation is `pikaia-orbit.f`. Some highlights of the code:

- Note that by design, Pikaia operates on $[0, 1]^p$, where p is the number of variables (in this case 6). Hence, we have to scale and translate the chromosomes (see function `orbit` in the source code for further details).
- To solve equations (1)-(3) for the predicted radial velocity, we included the functions `kepler` (Equation (3)) and `rtbis` (a bisection algorithm; see T. Krishnan's "Principles of Statistical Algorithms" lecture on 15 June 2005).
- The data is hardcoded in the source file; the actual data is in `rhocorb.dat`
- The number of generations, `ngen`, is set to 2500. The variable `ctrl` contains the GA parameters.

In 1997 Noyes, et. al. in an article entitled, "A Planet Orbiting the Star Rho Coronae Borealis" estimated the parameters of interest. Run `pikaia-orbit.f` and compare your results. They will appear once Pikaia completes execution.

Parameter	Noyes et. al. estimate
P	39.645 ± 0.088
τ	413.7 ± 8.2
ω	210 ± 74
e	0.028 ± 0.040
K	67.4 ± 2.2
V_0	not listed