Josh Engelsma

Adam Terwilliger

April 26, 2016

CIS 678 – Machine Learning

Project 5

Abstract

With our final project in CIS 678 – Machine Learning, we implement a genetic algorithm (GA) to find the global minimum of two complex functions: Rosenbrock's Banana function and Goldstein-Price function.

Implementation

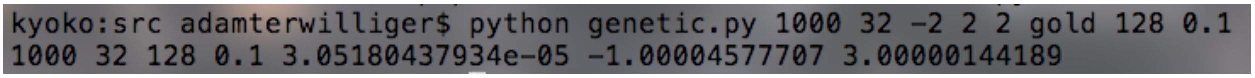
Our program is written in Python 2.7 and bash scripting in Unix. These programs were executed locally on each member's respective Macbook Pro (2012), testing on eos23 and okami.

Background

Two in-class datasets involved fishing, but testing was too variable with too few observations

Results

An example of our output can be seen in Figure 1. The parameters to the program include in order: population size, number of chromosome bits, function range min, function range max, number of function inputs, function of choice, number of generations, and percent of parents to keep.



```
kyoko:src adamterwilliger$ python genetic.py 1000 32 -2 2 2 gold 128 0.1
1000 32 128 0.1 3.05180437934e-05 -1.00004577707 3.00000144189
```

Figure 1. Example output of GA.

In Figure 1, we find with 1000 sized population, 32 chromosome bits, 128 number of generations, and 10% of the parents kept around after each generation, our GA finds the global minimum with accuracy to the fifth decimal place. In Figures 2 and 3, we observe the effect population size has on GA convergence, as not until log base 2 of 8 (256) sized population shows convergence. In Figures 4 and 5, we see that GAs do not converge well given under 32 chromosome bits. We find with Figures 6 and 7, that at least 32 generations are needed for convergence for Goldstein and 64-128 generations were needed for Rosenbrock. Inconsequentially, Figure 8 shows us that the percentage of parents kept after each generation from 1% to 99% is negligible with 50 generations and 1000 sized population.

Figure 2. Genetic Algorithm -- Rosenbrock’s Banana Function
F(x,y) vs. log2(popSize)

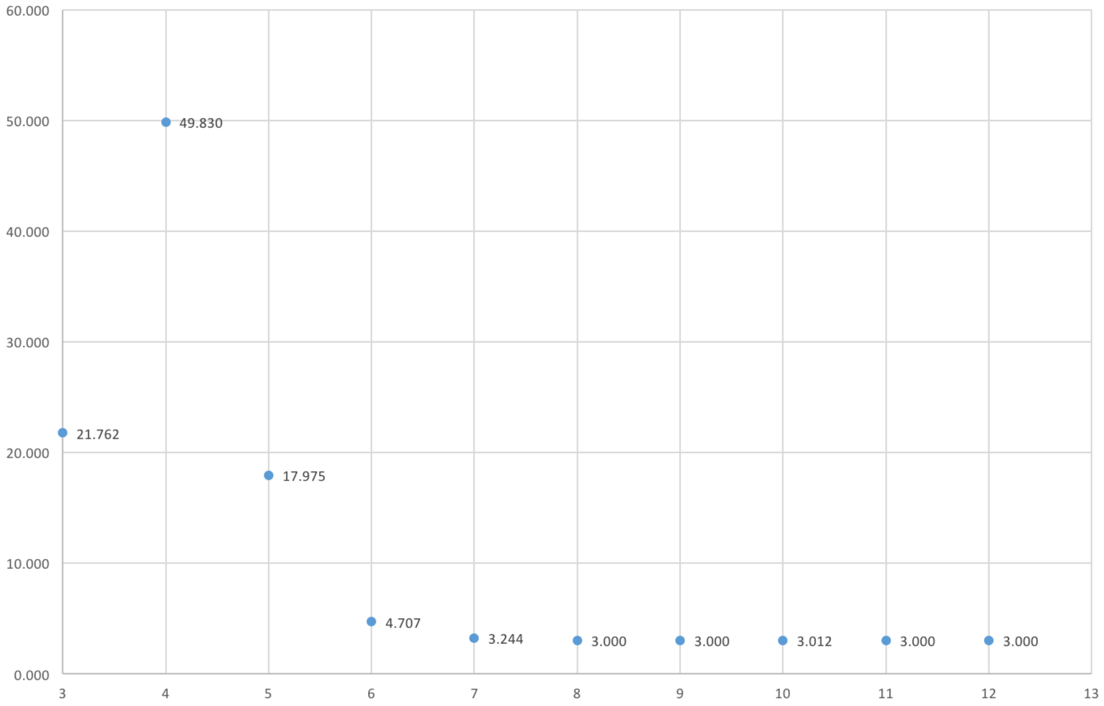


Figure 3. Genetic Algorithm -- Goldstein-Price Function
F(x,y) vs. log2(popSize)

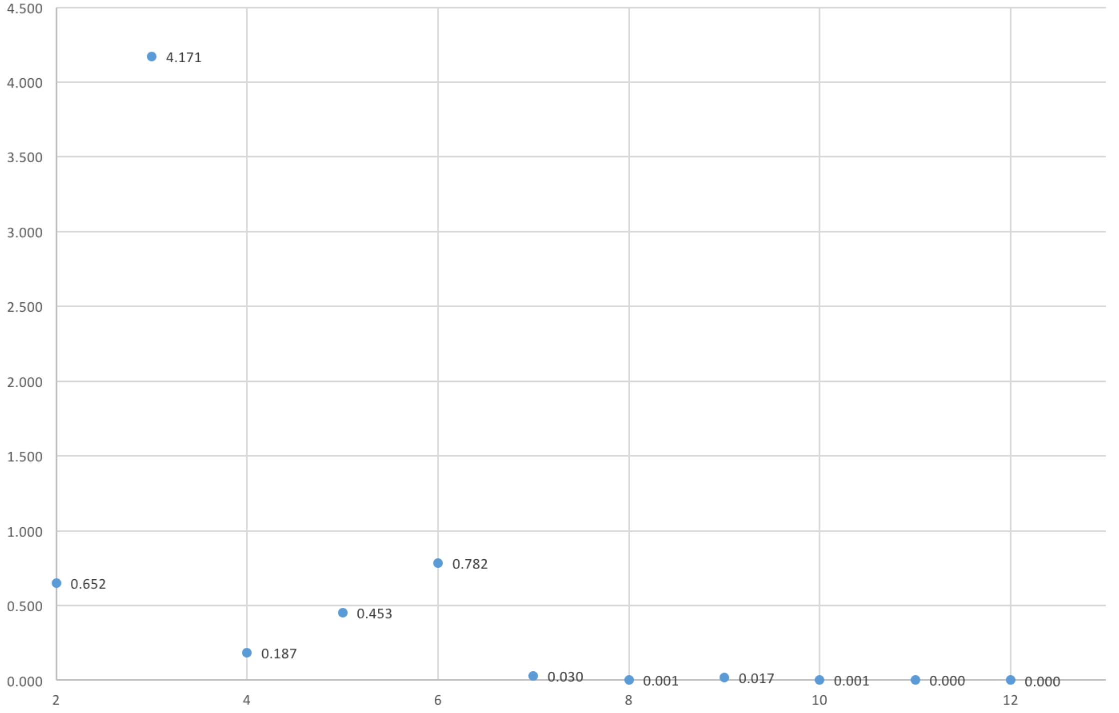


Figure 4. Genetic Algorithm -- Rosenbrock’s Banana Function
F(x,y) vs. log2(num bits)

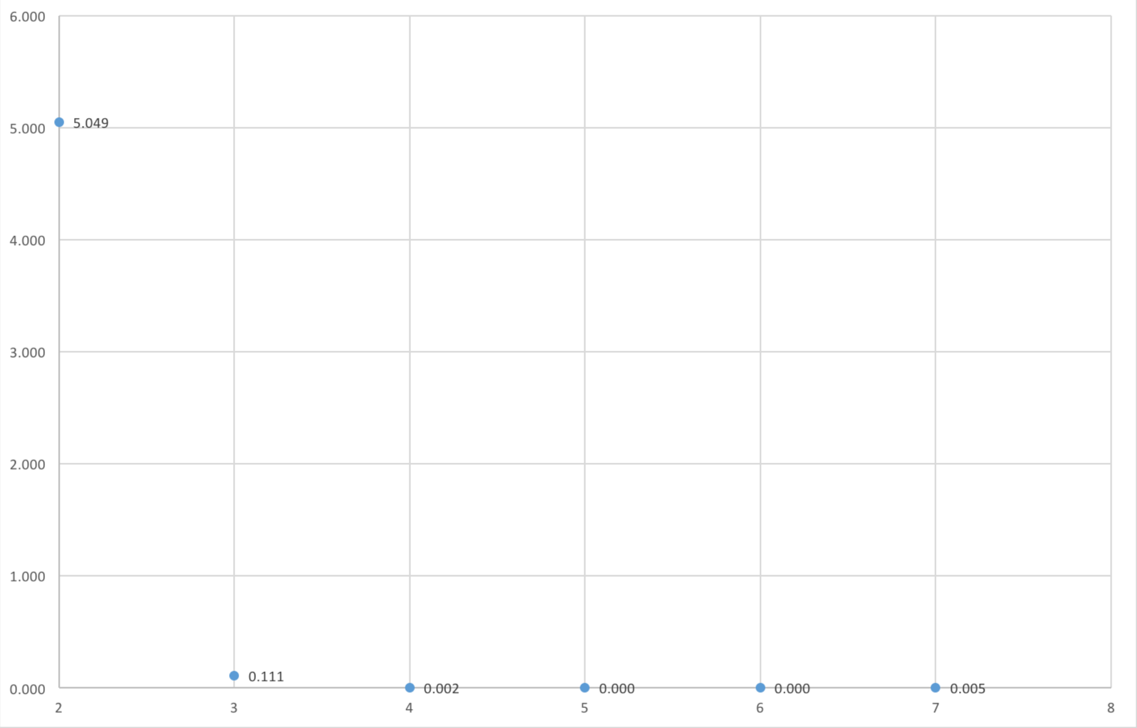


Figure 5. Genetic Algorithm -- Goldstein-Price Function
F(x,y) vs. log2(num bits)

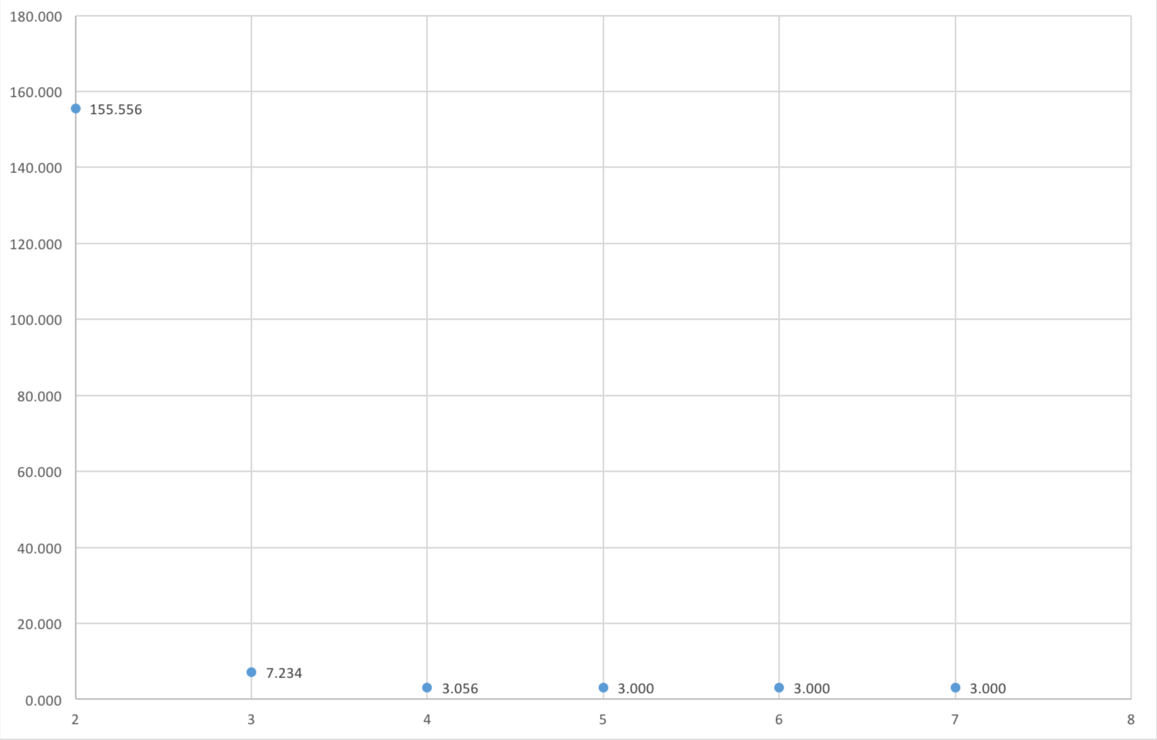


Figure 6. Genetic Algorithm -- Rosenbrock's Banana Function
F(x,y) vs. log2(num generations)

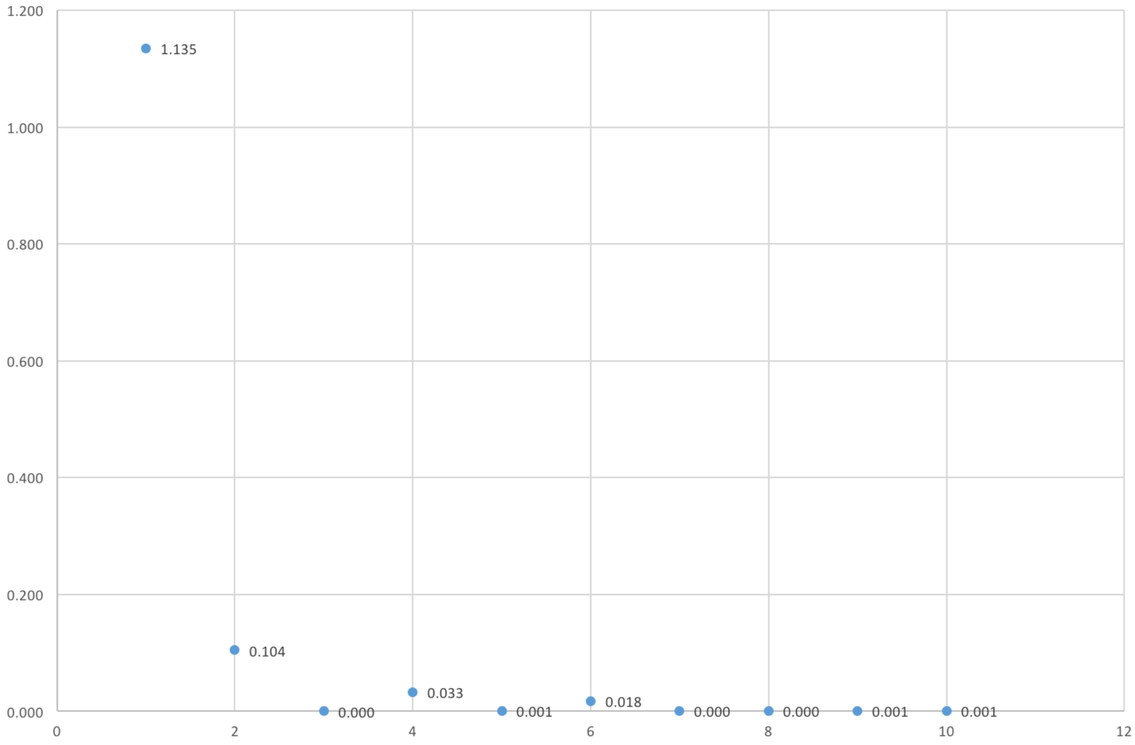
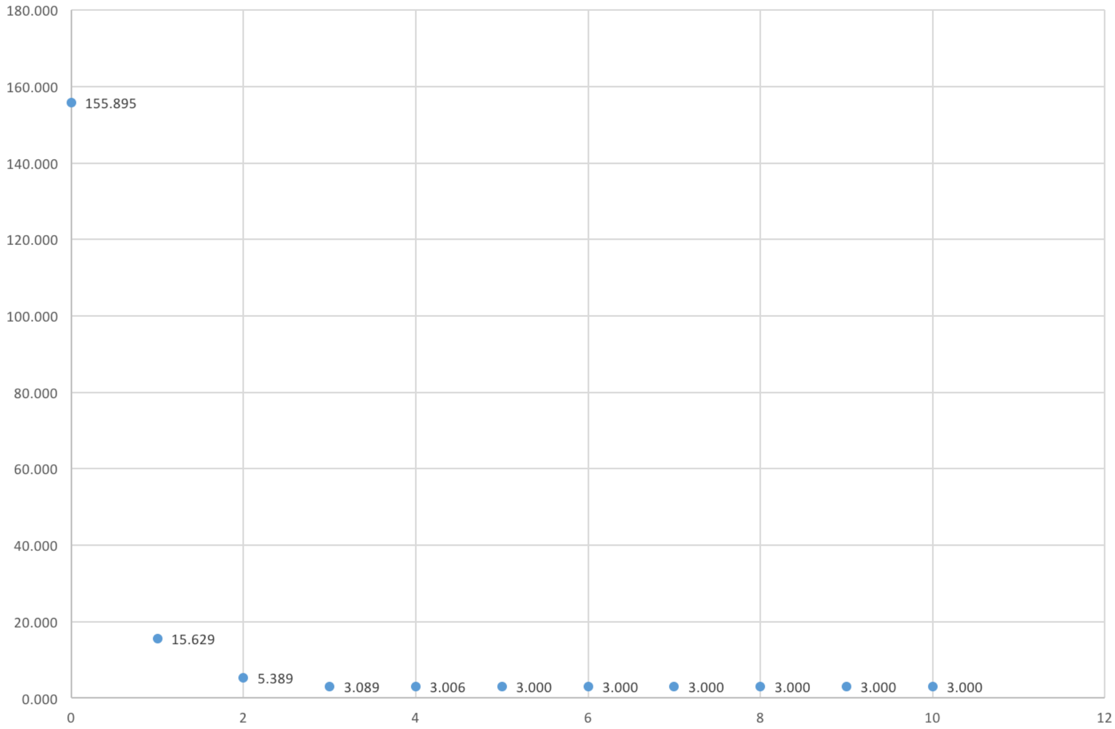


Figure 7. Genetic Algorithm -- Goldstein-Price Function
F(x,y) vs. log2(num generations)



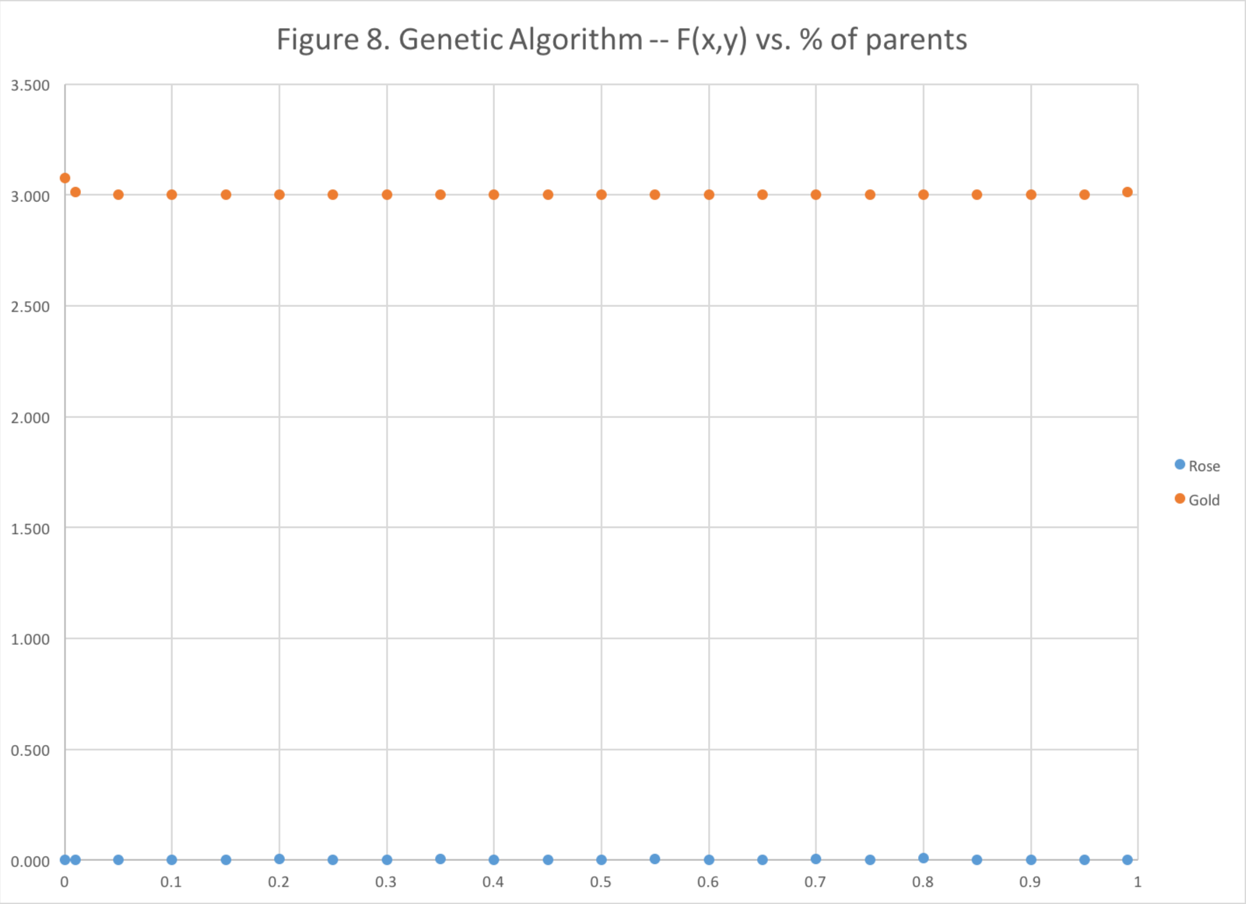


Figure 8. Genetic Algorithm -- F(x,y) vs. % of parents
3.500
3.000
2.500
2.000
1.500
1.000
0.500
0.000
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
● Rose
● Gold