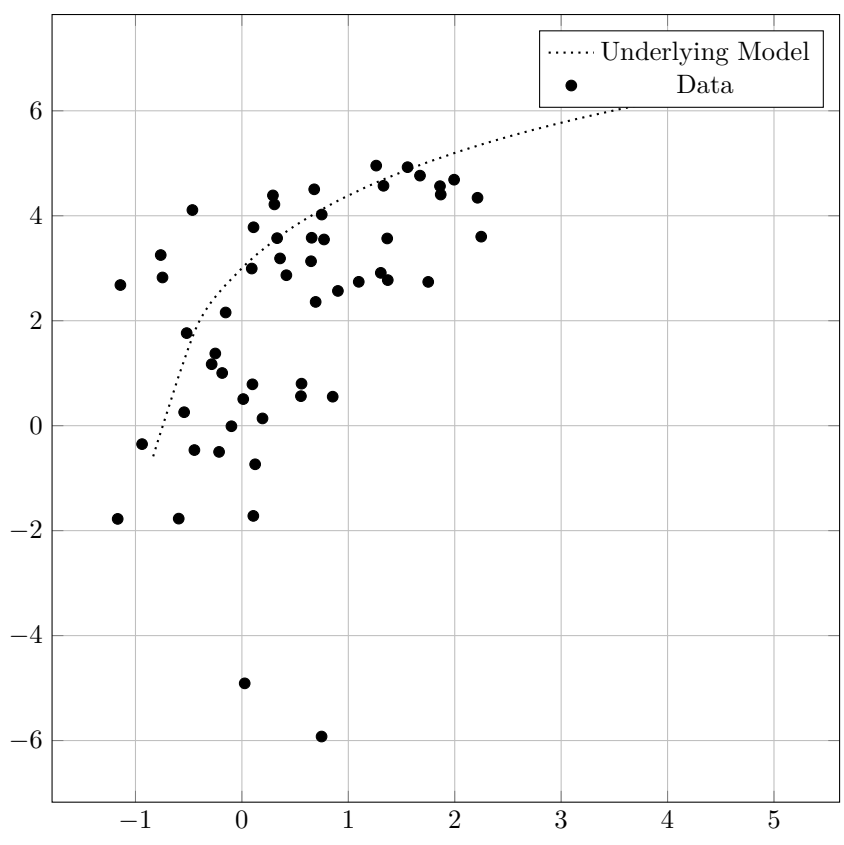
DE Optimization Explained

Optimizing a Logarithmic Function

Example Data created from a Logarithmic Function



Given the example data, fit a logarithmic function:

$$y = a * ln(x + b) + c$$

For a logarithmic equation in this form, optimize for 3 parameters:

$$(a , b , c)$$

1 Begin with an Initial Population of Models

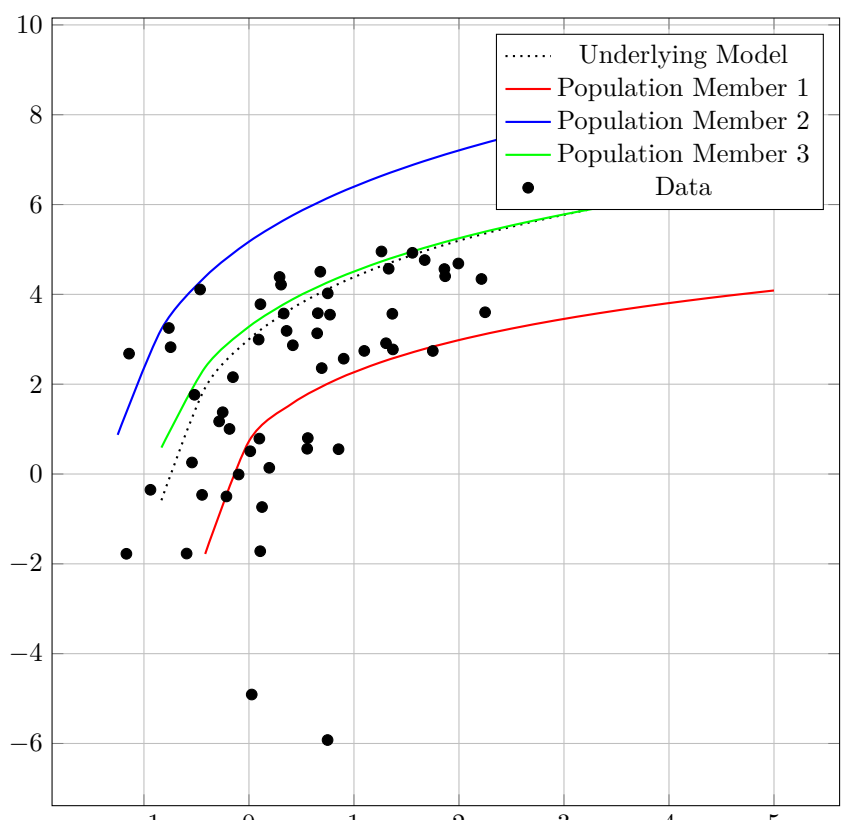
The initial population is a matrix-like list of of randomly (or pseudo-randomly) generated vectors:

$$\mathbf{P} = (v_1 , \vdots , v_n)$$

the element of each being parameters from the model being optimized:

$$v_1 = (a_1 , b_1 , c_1)$$
$$\vdots \quad \vdots$$
$$v_n = (a_n , b_n , c_n)$$

Visualize



2 Define a fitness function

As each member of the population defines a potential model (i.e., curve) to fit over the data, define a fitness function to compare each population member. For example, the root mean square error of each curve can be used:

$$RMSE = \sqrt{\frac{\Sigma(y-p)^2}{n}}$$
$$Fitness(v) = RMSE(v)$$

where

- y is the actual value of the data
- p is the predicted value from a model
- n is the population size

3 Iterate the Population

The population must evolve across discrete steps, called iterations, with each iteration involving 2 procedures: Mutation and Crossover

3.1 Mutation

Mutation involves combining population members to create a new potential mutant population member:

$$v_m = F(v_x , \dots , v_z)$$

where v_m is the mutant vector/model, and v_x through v_z are any number of randomly drawn models from the population. Mutation functions are often simple arithmetic combinations of candidate models. For example, the default mutation function in this implementation is:

$$v_m = v_x + M * (v_y - v_z)$$

where v_m is the mutant vector, and v_x through v_z are randomly selected models from the population. Additionally, M is a simple numerical constant set by the user. Constants such as this one are often included in mutation functions as they allow users to influence variation within the mutation process.

3.2 Crossover

Crossover involves selecting or creating a function that determines whether a mutant model replaces any single population member, as this function is applied to every individual population member. A simple and commonly used crossover function is a simple binomial function wherein a randomly determined constant number (between 0 and 1) is compared to a crossover constant:

$$v_n = v_m \; if \; \mathbf{X} > C \; else \; v_n = v_i$$

where v_n is the model being recorded during the iteration, v_m is the potential mutant model, v_i is the current model being considered, $\boldsymbol{X}$ is a randomly generated constant between 0 and 1, and C is a simple numerical constant (set by the user) between 0 and 1. In this case, if a user sets a value for C that is closer to 0, more mutants substitutions will occur. If the user sets a value closer to 1, fewer substitutions will occur.

4 Finalize via the Fitness Function

Once the initial population of models has evolved, the final step is to compute the fitness value for each evolved model:

$$v_{final} = Fitness_{Best}(\mathbf{P})$$

The model with the best RMSE is the optimal solution outputted by the algorithm at that particular iteration. Of note, the algorithm can be iterated any arbitrary number of times. It is best to stop when the algorithm converges to a model with a fitness function value that does not improve over further iterations.

Optimal Solution