Coding and Minimizing a Fitness Function Using the Genetic Algorithm

[Open This Example](matlab:edit%20gafitness)

This example shows how to create and minimize a fitness function using the Genetic Algorithm in the Global Optimization Toolbox.

**A Simple Fitness Function**

Here we want to minimize a simple function of two variables

min f(x) = 100 \* (x(1)^2 - x(2)) ^2 + (1 - x(1))^2;

x

**Coding the Fitness Function**

We create a MATLAB file named simple\_fitness.m with the following code in it:

function y = simple\_fitness(x)

y = 100 \* (x(1)^2 - x(2)) ^2 + (1 - x(1))^2;

The Genetic Algorithm solver assumes the fitness function will take one input x where x is a row vector with as many elements as number of variables in the problem. The fitness function computes the value of the function and returns that scalar value in its one return argument y.

**Minimizing Using ga**

To minimize our fitness function using the ga function, we need to pass in a function handle to the fitness function as well as specifying the number of variables in the problem.

FitnessFunction = @simple\_fitness;

numberOfVariables = 2;

[x,fval] = ga(FitnessFunction,numberOfVariables)

Optimization terminated: average change in the fitness value less than options.FunctionTolerance.

x =

0.3454 0.1444

fval =

0.4913

The x returned by the solver is the best point in the final population computed by ga. The fval is the value of the function simple\_fitness evaluated at the point x.

**A Fitness Function with Additional Arguments**

Sometimes we want our fitness function to be parameterized by extra arguments that act as constants during the optimization. For example, in the previous fitness function, say we want to replace the constants 100 and 1 with parameters that we can change to create a family of objective functions. We can re-write the above function to take two additional parameters to give the new minimization problem

min f(x) = a \* (x(1)^2 - x(2)) ^2 + (b - x(1))^2;

x

a and b are parameters to the fitness function that act as constants during the optimization (they are not varied as part of the minimization). One can create a MATLAB file called parameterized\_fitness.m containing the following code:

function y = parameterized\_fitness(x,a,b)

y = a \* (x(1)^2 - x(2)) ^2 + (b - x(1))^2;

**Minimizing Using Additional Arguments**

Again, we need to pass in a function handle to the fitness function as well as the number of variables as the second argument.

ga will call our fitness function with just one argument x, but our fitness function has three arguments: x, a, b. We can use an anonymous function to capture the values of the additional arguments, the constants a and b. We create a function handle FitnessFunction to an anonymous function that takes one input x, but callsparameterized\_fitness with x, a, and b. The variables a and b have values when the function handle FitnessFunction is created, so these values are captured by the anonymous function.

a = 100; b = 1; % define constant values

FitnessFunction = @(x) parameterized\_fitness(x,a,b);

numberOfVariables = 2;

[x,fval] = ga(FitnessFunction,numberOfVariables)

Optimization terminated: average change in the fitness value less than options.FunctionTolerance.

x =

1.6878 2.8924

fval =

0.6642

**Vectorizing Your Fitness Function**

Consider the previous fitness function again:

f(x) = a \* (x(1)^2 - x(2)) ^2 + (b - x(1))^2;

By default, the ga solver only passes in one point at a time to the fitness function. However, sometimes speed up can be achieved if the fitness function is vectorized to take a set of points and return a set of function values.

For example if the solver wants to evaluate a set of five points in one call to this fitness function, then it will call the function with a matrix of size 5-by-2, i.e. , 5 rows and 2 columns (recall 2 is the number of variables).

Create a MATLAB file called vectorized\_fitness.m with the following code:

function y = vectorized\_fitness(x,a,b)

y = a \* (x(:,1).^2 - x(:,2)).^2 + (b - x(:,1)).^2;

This vectorized version of the fitness function takes a matrix x with an arbitrary number of points, the rows of x, and returns a column vector y with the same number of rows as x.

We need to tell the solver that the fitness function is vectorized using the options. The options are passed in as the ninth argument.

FitnessFunction = @(x) vectorized\_fitness(x,100,1);

numberOfVariables = 2;

options = optimoptions(@ga,'UseVectorized',true);

[x,fval] = ga(FitnessFunction,numberOfVariables,[],[],[],[],[],[],[],options)

Optimization terminated: average change in the fitness value less than options.FunctionTolerance.

x =

-1.0815 1.1467

fval =

4.3850