Function Skeleton Design

Yu/Yachuan

December 4, 2017

The point marked by ? is additional, which we haven't thought thoroughly about how to design.

Input

1. Dataset
2. Type of regression
   - lm()
   - glm()
3. Criterion
   - AIC(default)
   - BIC
   - . . . (Other common criteria)
   - User can provide objective function?
4. Selection method and generation gap, G (a proportion of the generation to be replaced by generated offspring)
   - https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_parent_selection.htm (This website provides multiple methods on parent selection)
5. Genetic operators choice
   - Crossover, mutation(default) - Mutation rate and crossover rate ?
   - Additional operators provided by user?
6. Maximum number of iterations
7. Other arguments, similar to those of lm() and glm(), including some choices on the output format.

Funtion Design

1. Encoding the solution(chromosome)
   - It is natural in this problem. If the dataset includes C independent variables, then the length of the chromosome should be C. 1 represents that variable is included and 0 otherwise.
2. Select starting values
   - It is recommended to use a heuristic approach to select starting chromosome instead of random selection.
3. Parent selection in n-th generation
   - Transform objective function $f(\theta)$ to fitness function $\Phi(\theta)$
   - Selection mechanisom. Mutiple methods are provided in the website mentioned in the input part. Generation gap G is also considered.
4. Use genetic operator, crossover and mutation to generate the n+1-th generation.
   - The number of individuals, P, generated should be large at first and then decrease because the convergency is very fast at the beginning. It is recommanded that $C \leq P \leq 2C$.
5. Stopping critetia:
   - Whether the number of iterations exceeds the max number.
   - Whether the population fitness is good enough, or whether diversity of the population converges.(Design a good measure about population fitness?)
6. Choose the best chromosome in the final generation?

Other thoughts about the function design

1. Good data structure design to improve computing efficiency.
    - For example, if using partial updating(only update a proportion of the n-th generation in the n+1-th generation), the fitness value of a given chromosome may be used more than once. And if calculating the fitness value of one chromosome is complicated, it may be better to store the value. I think hashtable can solve the probelm.
2. Can we use smart select methods to avoid colinear problem?

Output

1. Result (?prediction of the parameters)
2. Objective funtion value
3. The number of iterations
4. ?The corelation of the variables selected.
5. Other outputs which can make our function perfect.
6. Convergency analysis

Implementation on real example

Test

1. Simulation tests.

The skeleton we designed is based on the material provided by Chris. There are a lot details in the design that we haven't considered yet. Further work is needed. Feel free t correct and improve it.