A Study of Evolutionary Algorithms and their applications in statistics

Sina Sanei

Dec. 2018

Evolutionary Algorithms

- Evolutionary algorithms are used to optimize functions by simulating natural evolution in a "Population" of candidate solutions.

- Suppose we want to find $\mathbf{x} \in R^n$ to maximize $f : R^n \rightarrow R$.
- We initialize a "Population" $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_N}$ where each $\mathbf{x_i} \in R^n$ called an "individual" having "fitness" $f(\mathbf{x_i})$
- These individuals are allowed to reproduce and mutate with a Selection Scheme that mimics the evolutionary concept of "survival of the fittest"
- After many generations, the optimal fitness (the function maximum) may be reached.
- Claim to fame is an ability to find a global maximum in presence of local maximum; Computations do not require derivatives or convexity but still can be intensive.

Simulation Study 1: Robust Regression

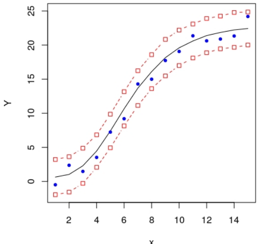
- We want to fit a *Gompertz* model for sigmoidal growth, as discussed by Lawrence and Arthur (1990):

$$Y_i = \alpha exp[-exp(\beta - \gamma x_i)] + \epsilon_i$$

  for $i = 1, ..., 15$; Where α, β and γ are parameters to fit, and errors are independent double-exponential.
- The maximum likelihood estimator for parameters minimizes:

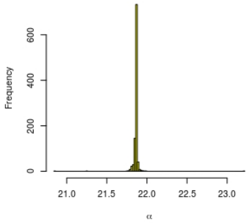
$$-\sum_{i=1}^{15} |Y_i - \alpha exp[-exp(\beta - \gamma x_i)]|$$

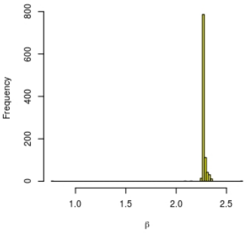
- Best fit parameters are know to be : $\hat{\alpha} = 22.37, \hat{\beta} = 2.14$ and $\hat{\gamma} = 0.395$

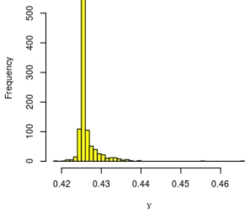


Implementation

- Initialization and Evaluation: Initial population with N=200 and randomly chosen with bounds of (15,30), (0,5) and (0,1)
- Selection Scheme: A version of tournament selection is implemented. The fitness of *i*th individual is compared to that of *i*+N/2, for *i*=1,2,..,N/2 and the *i*th individual is replaced by the more fit of the two.
- Recombination: Two parents randomly selected after selection : $x_{bi} = \frac{x_{mi} - x_{di}}{2} + C\frac{x_{mi} - x_{di}}{2}$, C is a truncated Cauchy random variate.
- With 1000 implementation of algorithm for 100 generations at each implementation







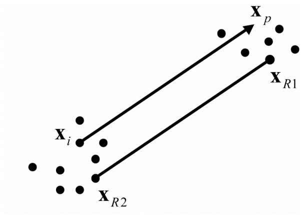
A Markov Chain Monte Carlo Version of Evolutionary Algorithm: Differential Evolution Markov Chain

- Differential evolution (DE) is a simple genetic algorithm. Assuming $N > 4$, the default proposal for ith member x_i in DE is:

$$x_p = x_{R0} + \gamma(x_{R1} - x_{R2})$$

- The idea is to instead of running the chains independently as a way to check convergence N chains are run in parallel and the jumps for a current chain are derived from the remaining N-1 chains.

- In order to turn DE into a Markov chain for drawing samples from a target distribution, the proposal and acceptance scheme must be such that there is detailed balance with respect to $\pi(.)$

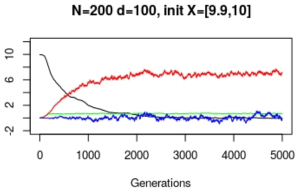


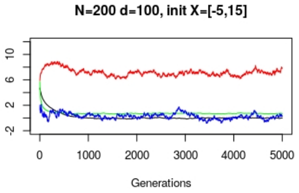
pseudocode

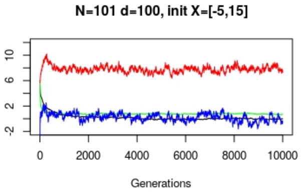
```
- for ( $s \leqslant N_{generations}$)cycle through generations
- for ( $i \leqslant N$)cycle through members of population
  Randomly select R1 and R2 unequal to i
  proposal : for ($j \leqslant N_{dimentions}$) {
        $x_p[j] = X[i][j] + \gamma(X[R_1][j] - X[R_2][j]) + e$
  r= fitness($x_p$)/fitness($X[i]$)
  selection: accept if Metropolis ratio $>$ unif[0,1]
- *end cycle through members of population*
- Record(X)
- *end cycle through generations*
```

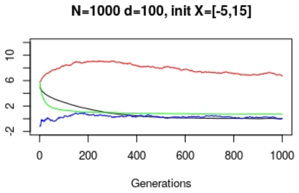
Simulation Study 2: Sampling from multivariate normal

- We want to obtain samples from a d=100 dimentional normal distribution , centered around zero.
- The covariance matrix was set such that the variance of the jth variable was equal to j and all pairwise covariances was 0.5 and $\gamma = 2.38/\sqrt{2d}$ (Mean, sd1,sd2, cov)









Comparison

Estimates	Mean	Sd1	Sd2	Cov
N=200,d=100	1.2581	0.6826077	5.985286	0.04945786
N=200,d=100	0.1837391	0.7643289	7.16684	0.2938345
N=101,d=100	0.2123698	0.8023778	7.791788	0.2385019
N=1000,d=100	0.7951775	0.9771417	7.866672	0.3077599

MCMCse	Mean	Sd1	Sd2	Cov
N=200,d=100	0.2672871	0.01099185	0.2086255	0.03255739
N=200,d=100	0.07257392	0.0291917	0.06084573	0.04823261
N=101,d=100	0.05331377	0.01651616	0.04199969	0.05117855
N=1000,d=100	0.195024	0.08997991	0.1336287	0.04075231