How to optimize a genetic algorithm?

Choosing the right parameter values for machine learning tasks is a challenge. Some results can be bad not because the data is noisy or the learning algorithm is weak, but because of incorrect selection of border values. This article provides a brief introduction to evolutionary algorithms (EAs) and describes a genetic algorithm (GA) that is one of the simplest random EAs.

Effective strategies are divided into four main categories:

- Constrained Optimization
- Multimodal Optimization
- Multiobjective Optimization
- Combinatorial Optimization

Example:

A GA program basically consists of the following parts:

1. Initialize the engine
2. Calculate fitness function
3. Parent selection
4. Crossover
5. Mutation
6. Optimization and printing best solution

Let us take an example based on guestimation, Guess word from given alphabets "TutorialsPoint".

Initialize the engine: Initialize the initial population , target and import required libraries as follows,

```
import random
import datetime
init_population="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrst
uvwxyz!."
target = "TutorialsPoint"
```

Calculate fitness function: Calculate the fitness value for selected parents means how close the selected parent is to the target.

```
def best_Fitness(selected_candidate, target):
   fit_value= 0
   for i in range(0, len(selected_candidate)):
       if target[i] == selected_candidate[i]:
           fit_value += 1
   return(fit_value)
```

Parent selection and CrossOver: Randomly select the parent as follows,

```
def best_Parent(length):
    genes = list("")
    for i in range(0,length):
        geneIndex = random.randint(0, len(init_population) -1);
        genes.append(init_population[geneIndex])
    return(''.join(genes))
```

Mutation: Perform mutation on offsprings in our case bitflip mutation performed as follows,

```
def mutation(parent):
   geneIndex = random.randint(0, len(init_population) -1);
   index = random.randint(0, len(parent) - 1)
   genes = list(parent)
   genes[index] = init_population[geneIndex]
   return(''.join(genes))
```

Optimization and printing best solution: In this section we call all these functions together until target value is not achieved as shown in output.

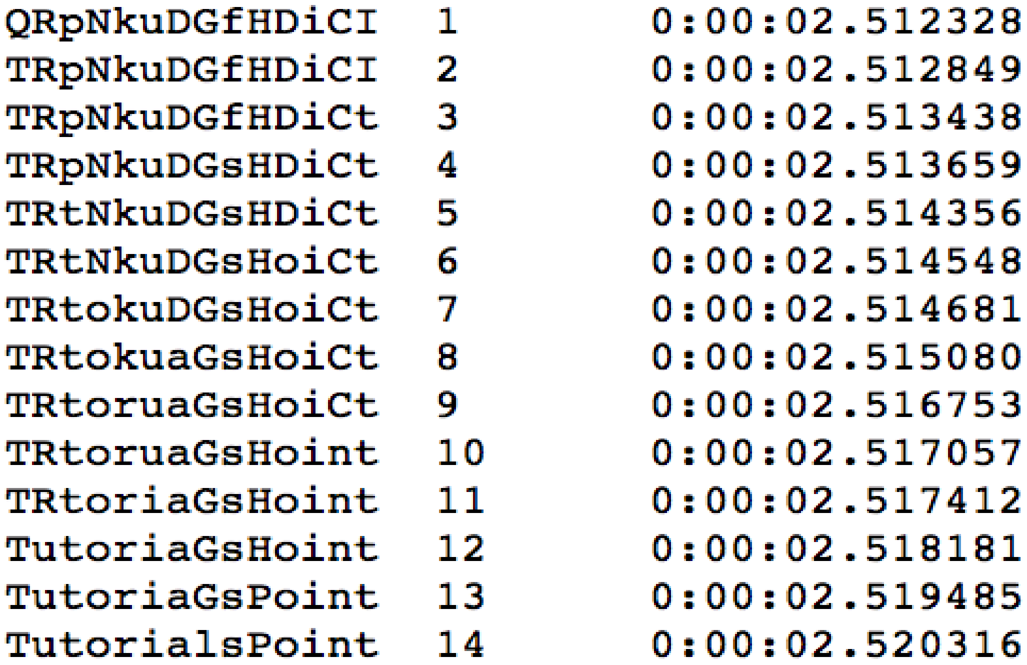
```
startTime=datetime.datetime.now()
def print_Best_Solution(candidate, startTime):
    timeDiff = datetime.datetime.now() - startTime
    fitness = best_Fitness(candidate, target)
    print("{}\t{}\t{}".format(candidate,fitness,str(timeDiff)))

bestParent = best_Parent(len(target))
bestFitness = best_Fitness(bestParent, target)
print_Best_Solution(bestParent, startTime)

while bestFitness < len(bestParent):
    offspring = mutation(bestParent)
    offspringFitness = best_Fitness(offspring, target)
    if offspringFitness > bestFitness:
        bestFitness = offspringFitness
        bestParent = offspring

        print_Best_Solution(bestParent, startTime)
```

Output:



```
QRpNkuDGfHDiCI   1        0:00:02.512328
TRpNkuDGfHDiCI   2        0:00:02.512849
TRpNkuDGfHDiCt   3        0:00:02.513438
TRpNkuDGsHDiCt   4        0:00:02.513659
TRtNkuDGsHDiCt   5        0:00:02.514356
TRtNkuDGsHoiCt   6        0:00:02.514548
TRtokuDGsHoiCt   7        0:00:02.514681
TRtokuaGsHoiCt   8        0:00:02.515080
TRtoruaGsHoiCt   9        0:00:02.516753
TRtoruaGsHoint   10       0:00:02.517057
TRtoriaGsHoint   11       0:00:02.517412
TutoriaGsHoint   12       0:00:02.518181
TutoriaGsPoint   13       0:00:02.519485
TutorialsPoint   14       0:00:02.520316
```