# Genetic Algorithms

April 16, 2020

[1]:
```
import sys
sys.path.append(r'C:\Users\Maggie\Desktop\George')
```

[2]:
```
from George import *
```

[3]:
```
help(Evolution)
```

```
Help on class Evolution in module George:

class Evolution(builtins.object)
 |  Evolution(target, populationsize=100)
 |
 |  Class for Evoloving a population to reach the target string
 |
 |  Attributes:
 |      target  target string you wish to evolve to
 |      populationsize  size of population of each generation, default is 100
 |      population      population of each generation, will change with
evolution
 |      fitness         fitness scores of the population
 |      mating_pool     mating pool to generate new members of population
 |      generation      number to represent the generation you are on
 |      target_acquired boolean field, if true then the target was required
 |      max_fitness     current maximum fitness score of population
 |      target_population   the member of the population that is equal to the
target string once evolution is completed
 |      closest_target  the current member of hte population that is closest to
the target
 |
 |  Methods defined here:
 |
 |  __init__(self, target, populationsize=100)
 |      :param target: target string
 |      :type target: str
 |      :param populationsize: size of initial population and subsequent new
generations population, default is 100
 |      :type populationsize: int
 |
```

```
 |  check_progress(self)
 |      Function to check progress of evolution
 |      can check max_fitness to check the highest fitness score
 |      closet_target will be the member of the population that has the highest
score
 |
 |  create_mating_pool(self)
 |      Function to generate the mating pool
 |      the matingpool will be generated from members of the population
 |      a member will be added to the matingpool as determined by their fitness
score %
 |      for example: if a member has .5 fitness score, it will be added 50 times
to the mating pool
 |
 |  evolution_rounds(self, rounds, mutation_rate=0.01)
 |      Function to generate rounds of evolution
 |      main driver of the genetic algorithm class
 |      evolution_rounds will stop when the target is acquired
 |
 |      :param rounds: rounds of evolution
 |      :type rounds: int
 |      :param mutation_rate: rate for mutation
 |      :type mutation_rate: float
 |
 |  generate_fitness(self)
 |      Function to generate the fitness score of each population member
 |      fitness score is determined by the number of correctly placed characters
in the string
 |
 |  generate_initial_population(self)
 |      Function to generate a population sized denoted as size
 |      population is of N generated random DNA elements
 |      N is the length of the target
 |
 |  mutation(self, mutation_rate=0.01)
 |      Function to mutate childs DNA
 |      This will allow more diversity in the new generation
 |      mutation rate is set at 0.01, but can be changed when calling
evolution.evolution_rounds
 |      if mutating then the dna character will be randomly exchanged for one
that is within the ASCII codes of the
 |      target str
 |
 |      :param mutation_rate: rate to mutate
 |      :type mutation_rate: float
 |
 |  new_generation(self, mutation_rate=0.01)
 |      Function to create a new generation
```

```
 |
 |  reproduction(self)
 |      Function to pick two parents from the mating pool and combine them for
the new generation
 |      2 parents are picked randomly
 |      The two parents will make 2 babies.
 |      The first will be the "best" parts from parentA and the rest from
parentB
 |      the second will be the "best" parts from parentB and the rest from
parentA
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
```

### 0.1 A simple example

The longer the target string is the more evolution rounds will need to occur

[7]:
```
# specify the target string
target = "to be or not to be"
# create the class instance, this will use the default population size 100
evolution = Evolution(target)
```

[8]:
```
# generate one evolution round
evolution.evolution_rounds(1)
```

```
Generations:
100%|                                          |
1/1 [00:00<00:00, 13.36it/s]

Need more evolution rounds!
Closest to target:  t t   or n o to ee
Max Population Fitness: 61.11%
Generation:  2
```

[9]:
```
# you can continue evolution rounds by recalling evolution_rounds
# you can also change the mutation rate in evolution_rounds
# mutation rate = 0.1 = 10% mutation
# default mutation_rate = 0.01 (1%)
# careful to not increase mutation rate too high, it can throw off evolution
```

```
evolution.evolution_rounds(1, mutation_rate=0.1)
```

```
Generations:
100%|                                          |
1/1 [00:00<00:00,  9.28it/s]
```

```
Need more evolution rounds!
Closest to target:  to b  ob not to ee
Max Population Fitness: 83.33%
Generation:  3
```

[10]:
```
# It is close!

evolution.evolution_rounds(10)
```

```
Generations:  20%|
| 2/10 [00:00<00:01,  7.40it/s]
```

```
Target Acquired:  to be or not to be
Generation acquired: 5
```

evolution_rounds will automatically stop when the target string is acquired

### 0.2 A longer Example

All of Shakespeare's work is freely available online. Sonnet 1 was found from: https://www.holybooks.com/wp-content/uploads/Shakespeare-Complete-Works.pdf

[11]:
```
with open('sonnet_1.txt') as f:
    mylist = f.read().splitlines()
```

[12]:
```
text = ''.join(str(v) for v in mylist)
```

[13]:
```
print(text)
```

```
 From fairest creatures we desire increase, That thereby beauty's rose might
never die, But as the riper should by time decease, His tender heir might bear
his memory: But thou contracted to thine own bright eyes, Feed'st thy light's
flame with self-substantial fuel, Making a famine where abundance lies, Thy self
thy foe, to thy sweet self too cruel: Thou that art now the world's fresh
ornament, And only herald to the gaudy spring, Within thine own bud buriest thy
content, And tender churl mak'st waste in niggarding: Pity the world, or else
this glutton be, To eat the world's due, by the grave and thee.
```

[14]:
```
# strip the text and make everything lowercase, this will help speed up␣
↪evolution!
```

```
text = text.strip()
text = text.lower()
print(text)
```

```
from fairest creatures we desire increase, that thereby beauty's rose might
never die, but as the riper should by time decease, his tender heir might bear
his memory: but thou contracted to thine own bright eyes, feed'st thy light's
flame with self-substantial fuel, making a famine where abundance lies, thy self
thy foe, to thy sweet self too cruel: thou that art now the world's fresh
ornament, and only herald to the gaudy spring, within thine own bud buriest thy
content, and tender churl mak'st waste in niggarding: pity the world, or else
this glutton be, to eat the world's due, by the grave and thee.
```

[15]:
```
# you can also increase the population size,
# increasing population sizes increases the complexity for each evolution round
# decreasing population sizes will decrease the variability in the population,
# but evolution rounds will complete faster
evolution1 = Evolution(text, populationsize=150)
```

[16]:
```
evolution1.evolution_rounds(1)
```

```
Generations:
100%|                                          |
1/1 [00:02<00:00,  2.43s/it]

Need more evolution rounds!
Closest to target:    the hm t v teergwge rsentiearn ies lnrbi nrmt guryn
rtamhlot pot nshhsiua io p kagtlsataeut r eioe nern gu  n'elhee ye attiai ymiw
sutrsrth ac hrtnee tei,olenrt osiihstereaoisynuhrst omem tkeoelllr iedthps s  te
ge'ldci ,  irdt ,olteerbie  aelfr odfuiihdtreareb utehirguh yrsgpahlhi
,adei,soae il fnrro: anbnlbi,e rece,sen,snm t  t scotnh b  h  dseblat  irwealmdt
is bebysee yg sga hbotn ghbr e is o sa setek   a ebdm wboa u  aenii t t syblhwa
btd it-teir i.retnibwlnredtusttldre,me  ahrmh raeuea vtaatbi  e' gy r agt wiby
e    bee a nfe  thrhiw n,uctmn'phitam,to d teon dilde uhto rtlhl o rnwfs   g s
ao
Max Population Fitness: 16.58%
Generation:  2
```

[17]:
```
evolution1.evolution_rounds(10)
```

```
Generations:
100%|                                          |
10/10 [00:45<00:00,  4.60s/it]

Need more evolution rounds!
Closest to target:  from fairest ireatures we desire inurease, that thereb
beautyls rose might neler dien but as the rieer should by tiee de ease, his
```

```
tenoer heir might bear his memory: but thou wontracted to thine own bright eyesi
heedsst toy lightos flame nith selfwsu stantial fuele making a famine where
abundance lies, thy self thy foe, to thy sweet self too eruelf thou that art now
the world's fresh ornament: and only herald to the gaudy spring, w thin thine
own  ud buriest thy content, and tender churl mas'st waste in niggardange hity
the torld, or else this geutton be, to eat the world's due, dy the rrade and
thee.
Max Population Fitness: 94.42%
Generation:  12
```

[18]:
```
# this will need a lot more rounds than the simple example!
evolution1.evolution_rounds(100)
```

```
Generations:
100%|                                        |
100/100 [10:31<00:00,  6.31s/it]

Need more evolution rounds!
Closest to target:  from fairest creatures we desire increase, that thereby
beauty's rose might ne er die, but as the rider should by time decease, his
tender heir might bear his memory: but thou iontracted to thine own bright eyes,
feedtut thy lightws flame with self-substantial fuel, making a famine where
abundance lies, thy self thy foe, to thy sweet self too crueli thou that art now
the world's fresh ornamente and only herald to the gaudy spring, within thine
own bud buriest thy content, and tender churl mat'st waste in niggarding   ity
the world, or else this glutton be, to eat the world's due, by the graue and
thee.
Max Population Fitness: 98.03%
Generation:  112
```

[19]:
```
# It's pretty close !
# it's possible that the variation in the population is decreased by now and we␣
 ↪might never converge
# we can try to increase mutation rate to increase variation in the population
evolution1.evolution_rounds(50, mutation_rate=0.1)
```

```
Generations:
100%|                                         |
50/50 [06:06<00:00,  7.32s/it]

Need more evolution rounds!
Closest to target:  from fairest creatures we desire increasec thatrthereby
beeut 's roee might neuer duei but as yie rietr should by time decease, his
tender heir mightfbear hi  memory:edut thor con racted to toine ownebright
eyeswefeed'st thy light's flamg with self-eubstantias fuel, making a famine
where abundance lies, thy self thy foe, do thr s een self too cruel: teou that
```

```
art now the world's fresh orna ent, and onlyeherold to the gaudy sp'ing, w,thin
thire own bud buoiest thy content, tnd tender churl mak'st waste i, niggarding:
pity the wored, or else t,is bluiton be, to eft the world's due, by she grave a
v thee.
Max Population Fitness: 92.28%
Generation:  162
```

[20]:
```
# looks like the high mutation rate decreased the max population fitness!
evolution1.evolution_rounds(100)
```

```
Generations:  26%|
| 26/100 [02:32<07:15,  5.88s/it]
```

```
Target Acquired:  from fairest creatures we desire increase, that thereby
beauty's rose might never die, but as the riper should by time decease, his
tender heir might bear his memory: but thou contracted to thine own bright eyes,
feed'st thy light's flame with self-substantial fuel, making a famine where
abundance lies, thy self thy foe, to thy sweet self too cruel: thou that art now
the world's fresh ornament, and only herald to the gaudy spring, within thine
own bud buriest thy content, and tender churl mak'st waste in niggarding: pity
the world, or else this glutton be, to eat the world's due, by the grave and
thee.
Generation acquired: 188
```

After only 188 evolution rounds, we were able to derive Shakespeare's Sonnet 1. The text is 609 characters long

There is 28 different characters in Sonnet 1. According to the infinite monkey theorem the probability to randomly type sonnet 1 is (1/28)^609

This is an infinitely small probability. But with a genetic algorithm we were able to do it in 188 evolution rounds, which roughly took 25 minutes (the tqdm status bar will tell the time)