MDL Project Report

1. Genetic Algorithm:

- The initial population consists of mutating the overfit vector significantly so that we get a diverse initial population. We initially chose a size 10 population but later reduced it to 8 to reduce the server calls.

- We read the initial population and sorted the vectors according to their fitness.

- Our fitness function considers the difference between training and validation error and a term involving the validation error, and we generate the fitness with a value (Difference + Validation_Error / 2). The vector with a minimum of this value gets the maximum fitness

- Then we select the best pairs according to their fitness for crossover. Selection is fixed with selection happening with parents (1,2);(1,3);(1,4);(2,3). Here numbers are the vectors' ranks according to their fitness, with 1 being the best rank.

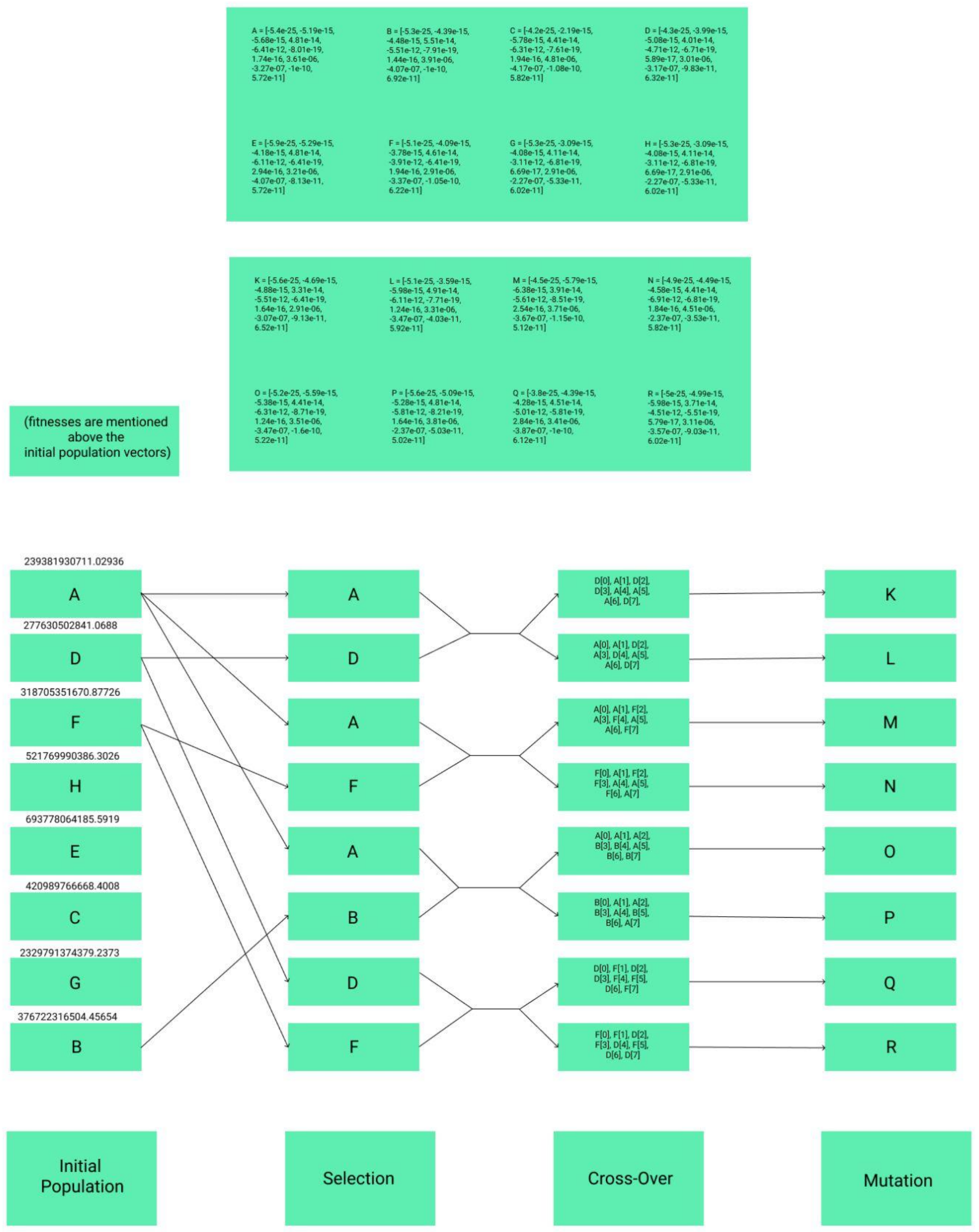
- Then each of these pairs produces two offsprings with every offspring being mutated upon.

- Each gene of the offspring has a 70% chance of getting the gene from the fitter parent.

- We noticed the overfit vector had a training error of 1e10 and the validation error of 3e10, and considering the vector was termed 'overfit', we assumed the value 3e11 is high, and 1e10 is low, so the ideal errors should be somewhere in between

2. Iterations of the Algorithm

Iteration 1


A = [-5.4e-25, -5.19e-15, -5.68e-15, 4.81e-14, -6.41e-12, -8.01e-19, 1.74e-16, 3.61e-06, -3.27e-07, -1e-10, 5.72e-11]

B = [-5.3e-25, -4.39e-15, -4.48e-15, 5.51e-14, -5.51e-12, -7.91e-19, 1.44e-16, 3.91e-06, -4.07e-07, -1e-10, 6.92e-11]

C = [-4.2e-25, -2.19e-15, -5.78e-15, 4.41e-14, -6.31e-12, -7.61e-19, 1.94e-16, 4.81e-06, -4.17e-07, -1.08e-10, 5.82e-11]

D = [-4.3e-25, -3.99e-15, -5.08e-15, 4.01e-14, -4.71e-12, -6.71e-19, 5.89e-17, 3.01e-06, -3.17e-07, -9.83e-11, 6.32e-11]

E = [-5.9e-25, -5.29e-15, -4.18e-15, 4.81e-14, -6.11e-12, -6.41e-19, 2.94e-16, 3.21e-06, -4.07e-07, -8.13e-11, 5.72e-11]

F = [-5.1e-25, -4.09e-15, -3.78e-15, 4.61e-14, -3.91e-12, -6.41e-19, 1.94e-16, 2.91e-06, -3.37e-07, -1.05e-10, 6.22e-11]

G = [-5.3e-25, -3.09e-15, -4.08e-15, 4.11e-14, -3.11e-12, -6.81e-19, 6.69e-17, 2.91e-06, -2.27e-07, -5.33e-11, 6.02e-11]

H = [-5.3e-25, -3.09e-15, -4.08e-15, 4.11e-14, -3.11e-12, -6.81e-19, 6.69e-17, 2.91e-06, -2.27e-07, -5.33e-11, 6.02e-11]

K = [-5.6e-25, -4.69e-15, -4.88e-15, 3.31e-14, -5.51e-12, -6.41e-19, 1.64e-16, 2.91e-06, -3.07e-07, -9.13e-11, 6.52e-11]

L = [-5.1e-25, -3.59e-15, -5.98e-15, 4.91e-14, -6.11e-12, -7.71e-19, 1.24e-16, 3.31e-06, -3.47e-07, -4.03e-11, 5.92e-11]

M = [-4.5e-25, -5.79e-15, -6.38e-15, 3.91e-14, -5.61e-12, -8.51e-19, 2.54e-16, 3.71e-06, -3.67e-07, -1.15e-10, 5.12e-11]

N = [-4.9e-25, -4.49e-15, -4.58e-15, 4.41e-14, -6.91e-12, -6.81e-19, 1.84e-16, 4.51e-06, -2.37e-07, -3.53e-11, 5.82e-11]

O = [-5.2e-25, -5.59e-15, -5.38e-15, 4.41e-14, -6.31e-12, -8.71e-19, 1.24e-16, 3.51e-06, -3.47e-07, -1.6e-10, 5.22e-11]

P = [-5.6e-25, -5.09e-15, -5.28e-15, 4.81e-14, -5.81e-12, -8.21e-19, 1.64e-16, 3.81e-06, -2.37e-07, -5.03e-11, 5.02e-11]

Q = [-3.8e-25, -4.39e-15, -4.28e-15, 4.51e-14, -5.01e-12, -5.81e-19, 2.84e-16, 3.41e-06, -3.87e-07, -1e-10, 6.12e-11]

R = [-5e-25, -4.99e-15, -5.98e-15, 3.71e-14, -4.51e-12, -5.51e-19, 5.79e-17, 3.11e-06, -3.57e-07, -9.03e-11, 6.02e-11]

(fitnesses are mentioned above the initial population vectors)

239381930711.02936
A
277630502841.0688
D
318705351670.87726
F
521769990386.3026
H
693778064185.5919
E
420989766668.4008
C
2329791374379.2373
G
376722316504.45654
B
A
D
A
F
A
B
D
F
D[0], A[1], D[2], D[3], A[4], A[5], A[6], D[7],
A[0], A[1], D[2], A[3], D[4], A[5], A[6], D[7]
A[0], A[1], F[2], A[3], F[4], A[5], A[6], F[7]
F[0], A[1], F[2], F[3], A[4], A[5], F[6], A[7]
A[0], A[1], A[2], B[3], B[4], A[5], B[6], B[7]
B[0], A[1], A[2], B[3], A[4], B[5], B[6], A[7]
D[0], F[1], D[2], D[3], F[4], F[5], D[6], F[7]
F[0], F[1], D[2], F[3], D[4], F[5], D[6], D[7]
K
L
M
N
O
P
Q
R
Initial Population
Selection
Cross-Over
Mutation

Iteration 2

A = [-5.6e-25, -4.69e-15, -4.88e-15, 3.31e-14, -5.51e-12, -6.41e-19, 1.64e-16, 2.91e-06, -3.07e-07, -9.13e-11, 6.52e-11]

B = [-5.1e-25, -3.59e-15, -5.98e-15, 4.91e-14, -6.11e-12, -7.71e-19, 1.24e-16, 3.31e-06, -3.47e-07, -4.03e-11, 5.92e-11]

C = [-4.5e-25, -5.79e-15, -6.38e-15, 3.91e-14, -5.61e-12, -8.51e-19, 2.54e-16, 3.71e-06, -3.67e-07, -1.15e-10, 5.12e-11]

D = [-4.9e-25, -4.49e-15, -4.58e-15, 4.41e-14, -6.91e-12, -6.81e-19, 1.84e-16, 4.51e-06, -2.37e-07, -3.53e-11, 5.82e-11]

E = [-5.2e-25, -5.59e-15, -5.38e-15, 4.41e-14, -6.31e-12, -8.71e-19, 1.24e-16, 3.51e-06, -3.47e-07, -1.6e-10, 5.22e-11]

F = [-5.6e-25, -5.09e-15, -5.28e-15, 4.81e-14, -5.81e-12, -8.21e-19, 1.64e-16, 3.81e-06, -2.37e-07, -5.03e-11, 5.02e-11]

G = [-3.8e-25, -4.39e-15, -4.28e-15, 4.51e-14, -5.01e-12, -5.81e-19, 2.84e-16, 3.41e-06, -3.87e-07, -1e-10, 6.12e-11]

H = [-5e-25, -4.99e-15, -5.98e-15, 3.71e-14, -4.51e-12, -5.51e-19, 5.79e-17, 3.11e-06, -3.57e-07, -9.03e-11, 6.02e-11]

K = [-4.6e-25, -2.89e-15, -6.78e-15, 5.51e-14, -3.81e-12, -7.41e-19, 1.64e-16, 3.81e-06, -3.17e-07, -4.43e-11, 6.52e-11]

L = [-4.4e-25, -4.29e-15, -5.38e-15, 4.41e-14, -6.01e-12, -7.91e-19, 8.39e-17, 3.21e-06, -4.37e-07, -9.13e-11, 5.32e-11]

M = [-4.8e-25, -4.09e-15, -5.88e-15, 4.51e-14, -6.61e-12, -8.21e-19, 2.04e-16, 4.01e-06, -3.27e-07, -3.33e-11, 4.72e-11]

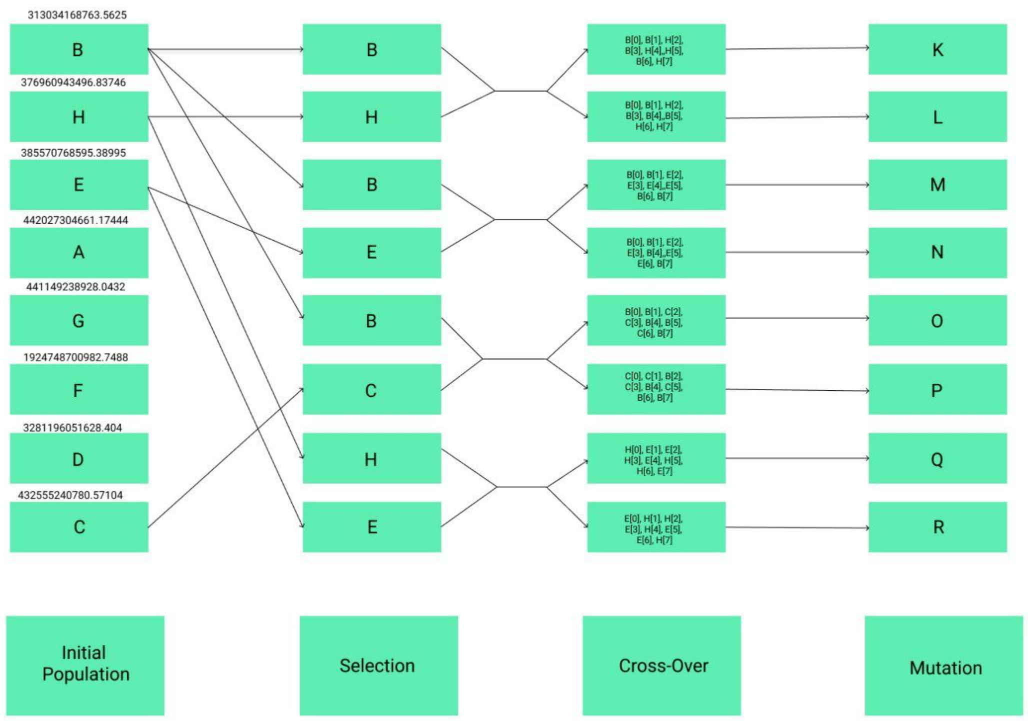
N = [-5.9e-25, -2.79e-15, -5.68e-15, 4.71e-14, -5.21e-12, -8.61e-19, 1.64e-16, 4.11e-06, -3.87e-07, -3.33e-11, 6.72e-11]

O = [-6e-25, -2.79e-15, -5.88e-15, 4.61e-14, -6.51e-12, -9.41e-19, 2.94e-16, 3.21e-06, -2.87e-07, -4.13e-11, 4.72e-11]

P = [-5.7e-25, -6.19e-15, -5.68e-15, 5.61e-14, -4.91e-12, -9.41e-19, 1.34e-16, 4.51e-06, -3.87e-07, -4.23e-11, 4.42e-11]

Q = [-5.4e-25, -5.19e-15, -6.48e-15, 3.11e-14, -4.81e-12, -6.41e-19, 5.69e-17, 2.81e-06, -2.97e-07, -1.9e-10, 5.52e-11]

R = [-4.8e-25, -4.79e-15, -6.48e-15, 4.21e-14, -5.31e-12, -5.11e-19, 6.19e-17, 3.61e-06, -3.07e-07, -9.43e-11, 6.72e-11]


313034168763.5625
B
376960943496.83746
H
385570768595.38995
E
442027304661.17444
A
441149238928.0432
G
1924748700982.7488
F
3281196051628.404
D
432555240780.57104
C
B
H
B
E
B
C
H
E
B[0], B[1], H[2], B[3], H[4],,H[5], B[6], H[7]
B[0], B[1], H[2], B[3], B[4],,B[5], H[6], H[7]
B[0], B[1], E[2], E[3], E[4],,E[5], B[6], B[7]
B[0], B[1], E[2], E[3], B[4],,E[5], E[6], B[7]
B[0], B[1], C[2], C[3], B[4], B[5], C[6], B[7]
C[0], C[1], B[2], C[3], B[4], C[5], B[6], B[7]
H[0], E[1], E[2], H[3], E[4], H[5], H[6], E[7]
E[0], H[1], H[2], E[3], H[4], E[5], E[6], H[7]
K
L
M
N
O
P
Q
R
Initial Population
Selection
Cross-Over
Mutation

Iteration 3

A = [-4.6e-25, -2.89e-15, -6.78e-15, 5.51e-14, -3.81e-12, -7.41e-19, 1.64e-16, 3.81e-06, -3.17e-07, -4.43e-11, 6.52e-11]

B = [-4.4e-25, -4.29e-15, -5.38e-15, 4.41e-14, -6.01e-12, -7.91e-19, 8.39e-17, 3.21e-06, -4.37e-07, -9.13e-11, 5.32e-11]

C = [-4.8e-25, -4.09e-15, -5.88e-15, 4.51e-14, -6.61e-12, -8.21e-19, 2.04e-16, 4.01e-06, -3.27e-07, -3.33e-11, 4.72e-11]

D = [-5.9e-25, -2.79e-15, -5.68e-15, 4.71e-14, -5.21e-12, -8.61e-19, 1.64e-16, 4.11e-06, -3.87e-07, -3.33e-11, 6.72e-11]

E = [-6e-25, -2.79e-15, -5.88e-15, 4.61e-14, -6.51e-12, -9.41e-19, 2.94e-16, 3.21e-06, -2.87e-07, -4.13e-11, 4.72e-11]

F = [-5.7e-25, -6.19e-15, -5.68e-15, 5.61e-14, -4.91e-12, -9.41e-19, 1.34e-16, 4.51e-06, -3.87e-07, -4.23e-11, 4.42e-11]

G = [-5.4e-25, -5.19e-15, -6.48e-15, 3.11e-14, -4.81e-12, -6.41e-19, 5.69e-17, 2.81e-06, -2.97e-07, -1.9e-10, 5.52e-11]

H = [-4.8e-25, -4.79e-15, -6.48e-15, 4.21e-14, -5.31e-12, -5.11e-19, 6.19e-17, 3.61e-06, -3.07e-07, -9.43e-11, 6.72e-11]

K = [-4.5e-25, -4.69e-15, -5.68e-15, 5.11e-14, -5.71e-12, -7.91e-19, 2.54e-16, 3.41e-06, -3.47e-07, -4.23e-11, 7.32e-11]

L = [-6.6e-25, -3.29e-15, -5.18e-15, 4.21e-14, -6.11e-12, -7.41e-19, 2.14e-16, 3.81e-06, -2.77e-07, -3.03e-11, 3.82e-11]

M = [-4.1e-25, -4.99e-15, -5.88e-15, 5.11e-14, -5.81e-12, -7.61e-19, 2.64e-16, 4.41e-06, -3.17e-07, -2.93e-11, 5.82e-11]

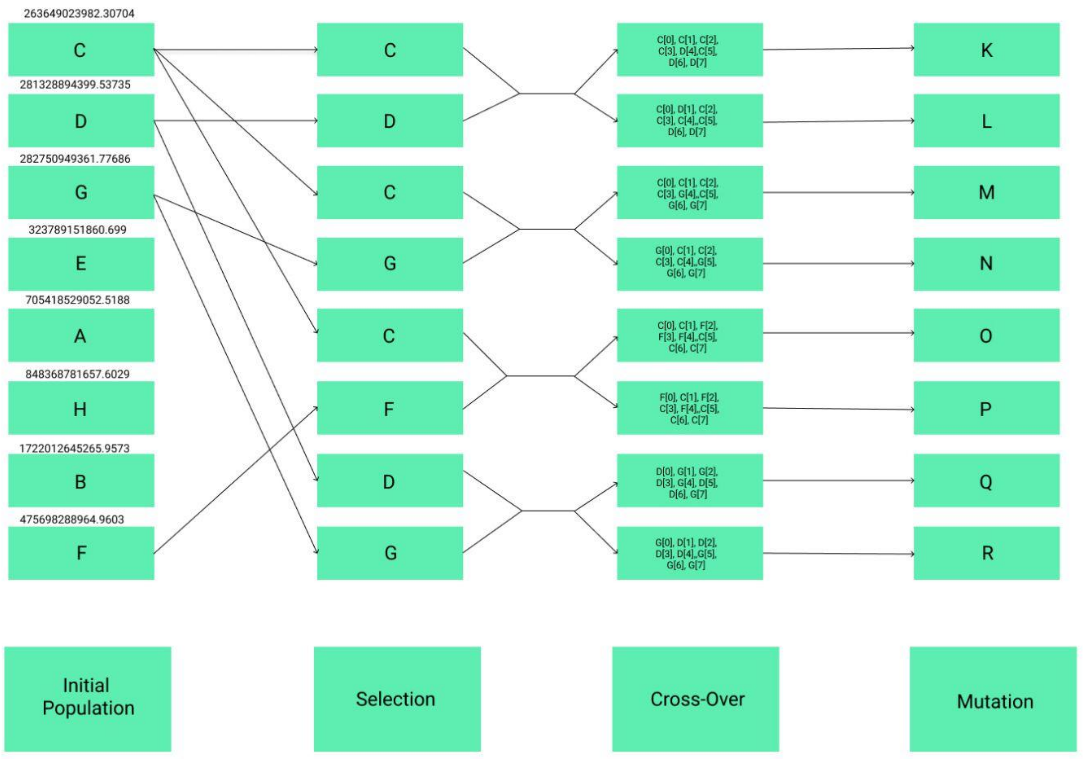
N = [-6e-25, -3.89e-15, -5.48e-15, 3.61e-14, -6.41e-12, -8.41e-19, 1.44e-16, 2.91e-06, -2.57e-07, -2.43e-11, 4.12e-11]

O = [-5.7e-25, -4.59e-15, -6.38e-15, 5.41e-14, -5.01e-12, -6.61e-19, 1.44e-16, 3.31e-06, -4.17e-07, -3.43e-11, 7.52e-11]

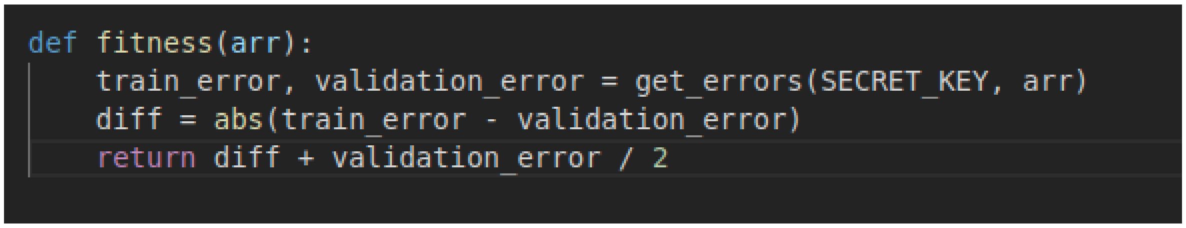
P = [-6.3e-25, -2.19e-15, -6.28e-15, 4.01e-14, -5.01e-12, -9.21e-19, 1.74e-16, 4.21e-06, -2.17e-07, -2.73e-11, 7.52e-11]

Q = [-5.5e-25, -3.59e-15, -5.58e-15, 3.61e-14, -5.91e-12, -8.61e-19, 1.74e-16, 4.31e-06, -2.87e-07, -3.93e-11, 5.12e-11]

R = [-4e-25, -2.49e-15, -6.28e-15, 3.71e-14, -6.51e-12, -7.41e-19, 3.14e-16, 3.21e-06, -3.97e-07, -3.73e-11, 4.42e-11]


263649023982.30704
C
281328894399.53735
D
282750949361.77686
G
323789151860.699
E
705418529052.5188
A
848368781657.6029
H
1722012645265.9573
B
475698288964.9603
F
C
D
C
G
C
F
D
G
C[0], C[1], C[2], C[3], D[4],C[5], D[6], D[7]
C[0], D[1], C[2], C[3], C[4],,C[5], D[6], D[7]
C[0], C[1], C[2], C[3], G[4],,C[5], G[6], G[7]
G[0], C[1], C[2], C[3], C[4],,G[5], G[6], G[7]
C[0], C[1], F[2], F[3], F[4],,C[5], C[6], C[7]
F[0], C[1], F[2], C[3], F[4],,C[5], C[6], C[7]
D[0], G[1], G[2], D[3], G[4], D[5], D[6], G[7]
G[0], D[1], D[2], D[3], D[4],,G[5], G[6], G[7]
K
L
M
N
O
P
Q
R
Initial Population
Selection
Cross-Over
Mutation

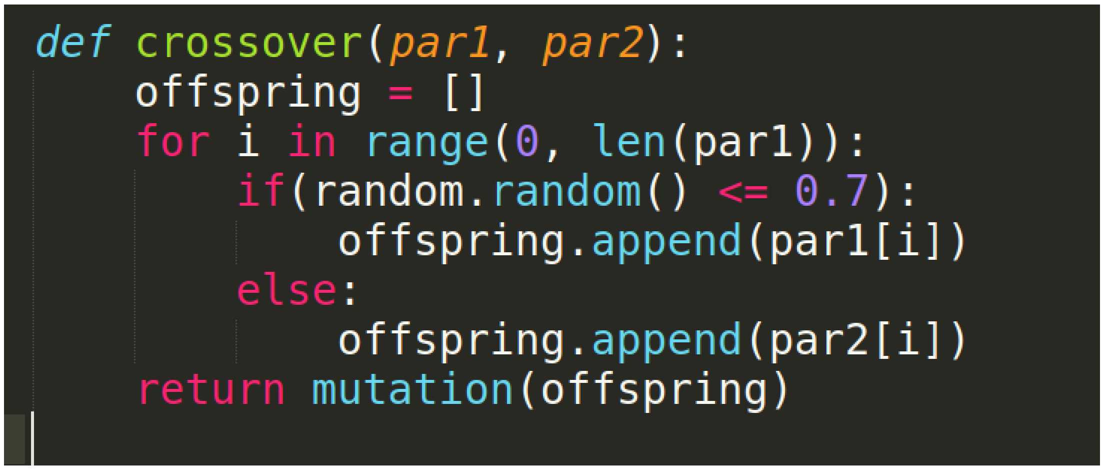
3. Fitness Function



```python
def fitness(arr):
    train_error, validation_error = get_errors(SECRET_KEY, arr)
    diff = abs(train_error - validation_error)
    return diff + validation_error / 2
```

For our fitness function, we have considered the difference between the training and validation error of the vector and also we put a term which accounts for the validation error so as to not overfit the difference parameter. The vector with the least value is the fittest vector. Minimizing difference ensures that the training and validation error are not far apart and the performance to both datasets is similar

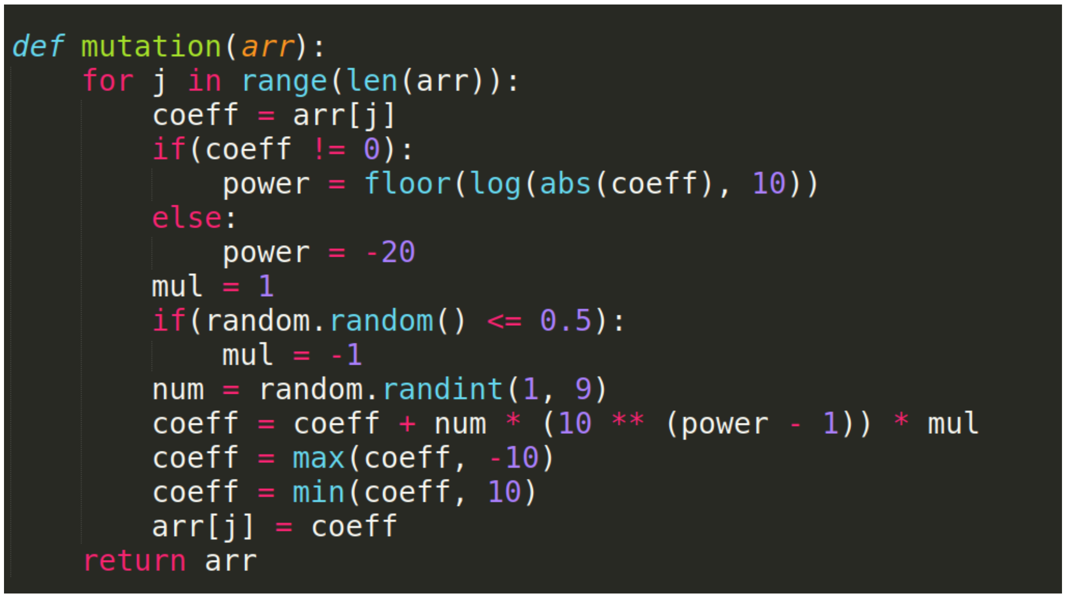
4. CrossOver Function



```python
def crossover(par1, par2):
    offspring = []
    for i in range(0, len(par1)):
        if(random.random() <= 0.7):
            offspring.append(par1[i])
        else:
            offspring.append(par2[i])
    return mutation(offspring)
```

For our crossover function, we have two parents with the first parent having better fitness than the second parent. Then, we generate a random number between 0 and 1, and if the number is less than 0.7 i.e. about 70% of the time, the offspring is given the gene of the first parent and the rest of the time offspring is given the gene of the second parent. This was done so that the offspring gets more number of genes from the fitter parent and might result in better fitness.

5. Mutations



```python
def mutation(arr):
    for j in range(len(arr)):
        coeff = arr[j]
        if(coeff != 0):
            power = floor(log(abs(coeff), 10))
        else:
            power = -20
        mul = 1
        if(random.random() <= 0.5):
            mul = -1
        num = random.randint(1, 9)
        coeff = coeff + num * (10 ** (power - 1)) * mul
        coeff = max(coeff, -10)
        coeff = min(coeff, 10)
        arr[j] = coeff
    return arr
```

We mutate every gene of the vector. So for the mutation of each gene, we added or subtracted a multiple(between 1 and 9) of the 1/10th exponent of 10 of the absolute value, For example: For the value 2e-13, we calculate 1e-14 and add or subtract it's multiple to the gene value

6. Hyperparameters

- The mating pool size is 4 for our population size of 8
- The size is 4 as we have chosen the top 4 vectors sorted according to their fitness(minimum value is 1st) and we pair them not using probabilities but using the best pair sequence according to their fitness
- Our population size is 8
- We sort vectors and pair them like (1,2),(1,3),(2,3),(1,4) and this is fixed
- We chose to avoid probabilities for selecting parents as with such a small pool size, there were high chances we got the same pair of parents again
- There is no splitting point in our crossover function as it is completely randomised as to which parent's gene will be given to the offspring, but we gave a 70%

preference to the parent with the higher fitness as we thought offspring should have more genes of the fitter parent.

7. Statistical Information

Over the last month or so, we observed different outputs of our learning algorithm, a few of which are listed below:

- When we were considering large mutations of the gene, the error skyrocketed to 1e50. No improvements were seen even till the 200th generations, so we decided to constrain the mutation to 1/10th of the gene's absolute value.

- The base algorithm diverged instead of converging. Then we made amends to the mutation and selection criteria. After those, our algorithm started converging around the 10th generation with generations before having a similar error to the initial population vectors.

- Also, after the 250th generation mark, our algorithm stopped to reduce errors. Instead, the errors started increasing a little but were more or less at the same level, and we tried till the 400th generation, but no progress was made, and hence our algorithm peaked out near the 250th generation.

8. Heuristics

We tried many different mutation, crossover and fitness functions and continuously improved upon them to get to the final algorithm.

- The initial mutation function made changes to a large extent. We added or subtracted random numbers to the gene, which proved to be a disaster as that kind of significant mutation resulted in terrible errors.

- For the crossover function, this was the crossover function we came up with from the starting. The first parent's probability was 50% initially, which is increased to 70% for better performance expectations as the first parent is the parent with better fitness.

- We went through many options for selecting the mating pool, like generating random pairs, mating every member with the best fitness vector. Still, we finally arrived at our current pool selection strategy where the best vectors are mutated amongst them so that the more fitter vectors participate in mating. For our population size of 8, every time 1st and 2nd, 1st and 3rd,1st and 4th, 2nd and 3rd are chosen where these ranks are their ranks according to their fitness. Every pair of parents crossover twice to produce two new offspring, and hence we get a new population after the mutation of the crossover population.

9. Comments of Error

- The overfit vector had a training error around 1e10 and a validation error around 3e11, so we aimed to bring down validation error close to the training error. We were able to bring the validation error down and keep the training error near the level. The best vectors have training and validation errors between 1e10 and 1e11, so their difference is also minimised, and also the error values are less.

- These errors indicate that these vectors may work well on unseen points. The difference between the training error and validation error is less, so the error on an unseen dataset with theoretically sound points may also be near this range. As the values are even less, the test error around might come near these values only and have a low value.

- Also, for the intermediate evaluations leaderboard, we tested our best vectors. They seemed to give a decent Test error, which made us believe that these vectors may provide good results for any unseen dataset with theoretically sound points.

10. Tricks and Brute Force

A little brute force was applied on the overfit vector to select the initial population. We mutated upon the overfit vector several times to get a decent initial population to start with, which gave us better vectors in the future. We didn't use any vectors from the initial 10 generations, which provided good errors as we thought that might be unfair because we didn't use any algorithms. Instead we submitted the vectors from the later generations where the errors were good and the vectors came from an algorithm.