MDL PROJECT: GENETIC ALGORITHMS

Team 95: Arushi Mittal (2019101120) Meghna Mishra (2019111030)

Project Description

Using the given coefficients of an overfit model, apply genetic algorithms in order to generalize the model for a variety of datasets. Vectors consisting of 11 floating numbers in the range [-10, 10] are used to represent individual members of a population. The submission contains 10 top vectors which contain the coefficients of the individual features and can reduce the overfitting on the given population.

Genetic Algorithms

Genetic Algorithms are based on Charles Darwin's theory of natural evolution which states that the process of natural selection ensures the survival of the fittest members of a population which over time, leads to an overall improvement in the characteristics and abilities of all subsequent populations. Genetic algorithms are metaheuristics inspired by this theory that attempt to simulate the processes of selection and genetics. These algorithms are randomized search algorithms and are used to solve optimization problems so that the problems and solutions can evolve, especially when an objective function value needs to be maximized or minimized under some given constraints.

The following are the phases of a genetic algorithm:

- 1. Initial Population
- 2. Fitness Function
- 3. Selection
- 4. Cross Over
- 5. Mutation

Initial Population

In genetic algorithms, the initial population is used to create subsequent generations so that the model can be improved over time. This initial population contains members who can be used to create better children who can in turn be used to create better children. For this model, the initial population we used multiple approaches. The first approach involved using the initial vector, which led to high validation errors. The second approach we used involved using a vector of random numbers which led to some very mixed results, so because of the unpredictability we had to let go of it. The third approach involved mutating (described in the section on mutation below) the original vector by a small factor and small probability which led to issues similar to those of the original vector. Using a zero vector led to very low errors but it was not a good fit generally. That led to our final approach which involved mutating the initial vector with a high probability and low factor so that most of the original vector's components would change by a small amount. This ensured we were following the right approach but made minor changes so the vector was not overfit.

```
for i in range(popsize):
    population[i] = mutate(initial_vector, 0.9, mut_range)
```

Fitness Function

The fitness function is used to determine how compatible a data point is with the model. For this project the fitness function we used varied at different points. Initially we just used the reciprocal of the total error (train error + validation error). After this, we realized that while the total error was decreasing, the absolute difference between the train and validation error values was significant. Therefore we decided to use the reciprocal of the sum of the total error and absolute difference. While this did help, the effect was not significant enough, so we increased the weightage of the difference by multiplying it by 1, various floats between 1 and 2, 2, 5, 10, 50, 75, 100, 150, 200, 500, 1000 and 2000. We finally settled on 2000. We did try using random numbers but the results were not significant enough to warrant continued use of this method.

```
factor = 2000

for i in range(popsize):
    err = client.get_errors(key, population[i].tolist())
    errors[i] = np.copy(err[0]+err[1]+factor*(abs(err[0]-err[1])))
    errors1[i] = np.copy(err[0])
    errors2[i] = np.copy(err[1])

indices = np.copy(np.argsort(errors))
```

Selection Function

The selection function determines which individuals from the population are chosen to be the parents for the next generation. The selection function we have used is relatively simple. It is a modified version of the roulette wheel selection algorithm. Since we are selecting k parents, where k is 6, we pick the top k fittest individuals from the population. These individuals are then given different probabilities depending on their fitness. The probability array determines which individuals have a greater probability of getting selected, so that the children produced are better and can thus make better children for the next generation. As a result, members are chosen solely from the top k vectors.

Crossover Function

The crossover function describes how the genes of the individual parents are combined in order to produce children. In this case there are 2 children produced by two parents. For the crossover function we used a simulated binary crossover. This involves setting a random number to calculate the distribution factor depending on the value of the random number. The distribution factor determines how close the children are to the parents - larger factors indicate larger differences. The children are both calculated in different ways, where for the first child, the first parent is more significant and for the second child, the second parent is more significant. A total of 10 children are created by crossing over the 6 parents.

```
def crossover(parent1, parent2):
    child1 = np.zeros(11)
    child2 = np.zeros(11)
    parent1 = np.array(parent1)
    parent2 = np.array(parent2)

n = 3

u = random.random()
    if(u <= 0.5):
        b = (2*u)**(1/(n + 1))
    else:
        b = (1/(2*(1 - u)))**(1/(n + 1))</pre>
```

```
child1 = 0.5*((1 + b) * parent1 + (1 - b) * parent2)
child2 = 0.5*((1 - b) * parent1 + (1 + b) * parent2)
return child1, child2
```

Mutation

The mutation function involves simulating mutations in the real world, where genes can sometimes be changed due to external circumstances and biological reasons. Here, the mutation involves probability and range. The probability determines how likely a gene is to get mutated and the range determines the extent to which it can get mutated. For the initial vector, we wanted the mutations to occur more frequently in order to generalize the algorithm and we used probability of 0.9 and range of 10%. For the next generations, we used mutation probability of 0.4 and range of 10% so small changes would occur but less frequently. If the number is zero, then a random number on a very small scale is added to the zero data point to induce some mutation but not disrupt the shape of the model.

```
def mutate(arr, prob, mrange):
    for i in range(len(arr)):
        val = np.random.uniform(-mrange, mrange)
        if(arr[i] == 0.00000000e+00):
            val = random.uniform(-1e-11, 1e-11)
                 arr[i] = np.random.choice([arr[i] + val, arr[i]], p=[prob, 1-prob])
        else:
            arr[i] = np.random.choice([(arr[i]*val) + arr[i], arr[i]], p=[prob, 1-prob])

        if(arr[i] > 10):
            arr[i] = 10
        elif(arr[i] < -10):
            arr[i] = -10</pre>
```

Algorithm and Code Explanation

The algorithm is simple. We create the initial population from the given vector as explained above and then used the get_errors function in order to determine the errors for each member of the population. The fitness function was applied using these error values and then it was used for selection of the parents. The parents were selected using the selection algorithm described above after which the children were created.

```
child1, child2 = crossover(parent1, parent2)
```

After crossover, the children were mutated.

```
child1 = mutate(child1, mut_prob, mut_range)
child2 = mutate(child2, mut_prob, mut_range)
```

If any of the children were duplicates of the parents, they were immediately discarded. This was achieved by creating a while loop that ran until the number of children was 10. If the children were duplicates, the iterator did not update, and the clones were discarded from the populations of crossovers and mutated children.

```
comparison1 = (child1 == parent1)
comparison2 = (child2 == parent1)
comparison3 = (child1 == parent2)
comparison4 = (child2 == parent2)
if(comparison1.all() == True or comparison2.all() == True or comparison3.all() ==
True or comparison4.all() == True):
    gen crossover children[-1].pop()
    gen_mutated_children[-1].pop()
    gen_crossover_children[-1].pop()
    gen_mutated_children[-1].pop()
    gen_crossover_children[-1].pop()
    gen_mutated_children[-1].pop()
    gen_crossover_children[-1].pop()
    gen_mutated_children[-1].pop()
    continue
child_population[x2] = child1
x2 += 1
child_population[x2] = child2
x2 += 1
```

After the child_population array has finally been set, we find the errors for the children using the API. The benefit we get from this is that we are able to save our API calls. Since we use 16 calls in the beginning in order to find

errors in the population for sorting, we have the errors for the parents. When the children are created, they are all chosen for the next generation along with the set of parents from the previous generations. This way, the number of calls is 16 + 10 * n where n is the number of iterations. This is how the next generation population is determined.

```
for i in range(popsize-k):
    new_population[i] = np.copy(child_population[i])
    new_errors[i] = np.copy(child_errors[i])
    new_errors2[i] = np.copy(child_errors2[i])
    new_errors2[i] = np.copy(child_errors2[i])

for i in range(k):
    new_population[popsize-k+i] = np.copy(population[i])
    new_errors[popsize-k+i] = np.copy(errors[i])
    new_errors1[popsize-k+i] = np.copy(errors1[i])
    new_errors2[popsize-k+i] = np.copy(errors2[i])
```

Hyperparameters

POPULATION SIZE: 16

After trying 10 and 20 as population sizes, we realized that 10 was too restricted and did not yield sufficiently diverse results. On the other hand 20 was too expansive to manage and used a lot of API calls. The results in both were not ideal, and 16 gave us the advantage of both sizes.

SELECTED PARENTS PER GENERATION: 6 Using 6 parents from each generation ensured that the best parents are used to mate so the subsequent generations are better. Additionally, carrying them forward to mate with superior genes ensures the next generation will be even better.

CHILDREN PER GENERATION: 10 Creating almost double the number of parents ensures sufficient diversity and opportunity for mutation along with different combinations of the parent vectors. Additionally, since these children have a high probability of being superior to the previous generations, they will undoubtedly create better offspring and a more diverse pool. This ensures that if the children are mutated too much or not ideal, the parent genes can hopefully offer correction and if not, the subsequent generations are better than the previous ones.

Heuristics

The project was completed by dividing it into two different phases - the minimization of error phase, and the balancing of train and validation error phase. Dividing it this way ensured that the error was sufficiently reduced before we began to generalize for a larger dataset. Since the model works for data with respect to all datasets not just the training data or the leaderboard data, it is generalized for various datasets.

We used various heuristics as explained in the functions above. We tried using various types of initial vectors before we settled on our current strategy of mutating the initial vector by a large probability and a small factor.

We varied the number of parents and children a few times before settling on the current version. This way we were able to settle on our final numbers. The decision to include all the children and the best parents ensured that the subsequent generations showed the greatest improvements. This decision ensured that top vectors from previous generations remained in the next population, while also introducing sufficient diversity in the population. Other approaches with fewer children were too repetitive and yielded no new results, while too many children made the results very erratic.

We varied the factor for absolute difference while calculating fitness so we could bring the train and validation errors as close as possible and settled on 2000 after trying smaller values that barely made an impact.

Initially, we were calculating the errors inside the iteration loop for each population which wasted a lot of API calls. We then switched to the current approach which calculates the initial errors outside the loop and then calculates the errors for children in each iteration. The API calls per run went down from (populationsize + number of children) * number of iterations to population size + (number of children * number of iterations). The extra API calls helped us make better judgements and improve the model with better results everyday.

During error minimization, we encountered a total error value that did not change at all after 20+ iterations, leading to the realization that we have hit a local minima. In order to prevent this, we mutated the top vector the

way we mutated the initial vector and progressed from there (by increasing probability of mutation).											

Diagrams

Diagram 1

Original Population	[2.15791778e 12* 2.03546787e 12* 2.34047731e 13* 4.59193271e 11* 3.15093485e 11* 1.17909621e 15* 3.57499717e 11* 13* 10* 10* 10* 10* 10* 10* 10* 10* 10* 10	2.16338470e- 12- 2.25477315e- 12- 2.42314550e- 13- 4.2067555e- 11- 1.17906411e- 15- 1.00278475e- 15- 1.00278475e- 15- 1.57376725e- 16576775e- 16576775e- 167767611803e- 10	[2.15744751e- 117. 2.11849161e- 12- 2.36062055e- 13- 4.81537443e- 11- 11-11790611e- 15- 15- 16- 16- 16- 16- 16- 16- 16- 16- 16- 16	[2,391219776-12-2-2775216-12-2-2-2775216-12-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2	[2.19815812e 12 2.00051246e 11 12 2.34716927e 13 4.38524552e 11 11 5.62457246e 11 1.17907611e 1.0501796e 1.0501796e 1.0501796e 1.05057596 0.0507596 1.05057596 1.05057596	[2.19940078-12-24531346-12-2.2507,0556-13-13-2.2507,0556-13-13-2.2507,0556-13-13-2.2507,0556-13-13-2.2507,056-13-13-2.2507,056-1	[2.30240776-12.12.12.12.12.12.12.12.12.12.12.12.12.1	[7.271666077e- 12*.27465008e- 12*.27465300e- 12*.27465370e- 13*.27465370e- 11*.2746640758e- 11*.17909621e- 15*.366640758e- 14*.7846476- 06*.746928e- 06*.746928e- 06*.746928e- 06*.746928e- 06*.746928e- 06*.746928e- 07*.30951204e- 10]	T. 200038799e- 12 T. 200038799e- 12 T. 214577768e- 12 T. 155761486- 13 T. 155761486- 13 T. 155761486- 14 T. 155761486- 15 T. 155761486- 16 T. 155761486- 17 T. 155761486- 17 T. 155761486- 18 T. 155761486- 19 T. 155761486- 10 T. 15576	[[] 2.307364346 12. 2.067366346 12. 2.281192086 13. 4.673873626 11. 1.179363146 11. 1.17936314 1.17936314 1.17936314 1.17936316 1.17936314 1.17936314 1.17936314 1.17936314 1.17936314 1.1793631534 1.1793631534 1.1793631534	1.19350471e- 11- 2.54703455e- 13 4.54076456e- 11- 11- 117909611e- 15 9.58451703e- 16 10.5950971e- 06 1.45505779e- 08 7.255454430e-	2.28707010e- 12- 2.11853477e- 12- 2.17356017e- 13- 4.41575722e- 11- 15- 11- 15- 11- 15- 15- 16- 177769281- 177769281- 	[] 2.88166676 12 2.114787520 12 2.250975820 13 4.788104206 11 1 2.250975820 11 1 1.217472286 15 1.050129300 15 2.25612556 05 1131541a 06 1.455857596 07.766118036 10]		1: 137002120- 1131096779- 12131096779- 1314096779- 131409090- 1314090900- 1324099110- 1314090900- 1324099110- 1346648570- 134664850- 1346648	
Parents	2.13791776-12.230547876-12.23047216- 14.433927276-14.5.2507485-11. 15.00047615-13.200476-1						43e-11 -5.3685193 e-15 9.52913745e-1	6-11 2.396969816-05	[-2.391215776-12 13 4.798800896-1 1.179096316-15 1 -1.673769286-05 10]	11 -5.55035300e 1.05146531e 15	11 39696881e 05	[219815312e12 2026 13 458554852e11 56 1.17909631e15 1.0750 1.67378528e06 1.4590 10]	796e-15 2.3969698	13 4.801 1e-05 1.179096	3404e-11 -5.628946 1e-15 1.02466383e-	6-12-224551134-12-2250570586- 046-11-5425946556-11- -15-102462838-15-239699816-05- -06-145905758-08-7766118036-	
Crossover Children	[7.16743782e12 2.75167871e12, 2.4134367e13, 4.76157111e11, 5.54681487e11, 11.17905631e15, 5.97027211e16, 2.35696981e05, 1.4569318e08, 1.456905759e08, 7.76611893e10]	2.20 2.35 4.65 1.17: 9.62: 2.39: 1.67: 1.45:	3185915e-12 3185915e-12 317805e-13 507161e-11 320030e-11 309631e-15 57476e-16 596831e-06 505753e-08 505753e-08 521497e-10	[2,1854603e 2,108707746-1; 2,37077850-1 4,31475356-1 5,57476737-1; 1,17905431e-1; 2,82904756-0 1,57376256-0 1,57376256-0 7,76611803e-1;	2, 2, 3, 4, 1, 4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,	1.95898846e 12. 17233331e 12. 193318859e 13. 77233005e 11. 12223543e 11. 195017200e 15. 13172490e 16. 19595831e 05. 67376928e 06. 659057550 08.	[-2,28690 2,105037 2,340503 4,636218 5,274547 1,179096 9,565784 2,396946 1,457057 7,766118	86e 12 99e 13 94e 11 54e 11 51e 15 58e 16 51e 05 58e 06	[-7.197540956-12, 2.151548146-12, 2.240550776-13, 4.56081752-11, 5.61362566-11, 1.0677217506-15, 9.733445456-16, 2.275629458-05, 1.8394550-68, 1.8394550-68, 7.76611803-6-10]	2 2 4 6 1 1 2 1	2.34187087e-12, 02651659e-12, 34790278e-13, 55495415e-11, 129362617e-11, 17909631e-15, 39696381e-05, 55413976e-06, 45405755e-08,	[2.19789040 2.076463726 2.19363939 4.59254556 5.21613946 117995316 9.51213156 2.39098316 1.775639856 1.459057556 7.62801066	12, 13, 11, 15, 16, 05, 06,	[2.15755738e-12, 2.14641418e-12, 2.340585556-13, 4.76261386e-11, 1.17906316-15, 2.33966216-16, 2.33966216-16, 2.39663816-05, 1.427851506-08, 7.766118036-10	2,06354 2,34055 4,90785 5,27865 1,17905 8,72655 2,39696 1,87376 1,59422	170e-11, 200e-11, 631e-15,	
Mutated Children	[2.16743782e-12, 2.25367871e-12, 2.41343977e-13, 4.7815711e-11, 5.54081497e-11, 1.17909631e-15, 9.7907771e-16, 2.3955981e-05, 1.68595313e-06, 1.45505755e-08, 7.76611803e-10]	2.20 2.39 4.61 5.26 1.17 9.62 2.39 1.67 1.45	3185915e-12. 107-693e-12. 117-905e-13. 502161e-11. 120030e-11. 120030e-11. 120030e-16. 127-7476e-16. 120-757476e-16. 120-757476e-16. 120-757476e-16. 120-757476e-16. 120-757476e-16. 120-757476e-16. 120-757476e-16. 120-757476e-16. 120-757476e-16. 120-757476e-16.	[2.18546403e 2.10070774e-1; 2.37077805e1; 4.31475105e1; 5.57476737e1; 1.17909431e1; 5.8399435e1; 2.600478000; 1.6737622e0; 1.57347708e0; 7.76611803e1;	2. 2. 1. 4. 1. 5. 5. 5. 5. 5. 5. 5. 5. 5. 5. 5. 5. 5.	1.958988486-12 172333316-12 193338296-13 7729330096-13 1930272006-15 10727006-15 10727006-15 10727006-15 10727006-15 10727006-15 10727006-15 10727006-15 10727006-15 10727006-15 10727006-15	[-2,28690 2,105037 2,340502 4,576513 5,274547 1,179096 9,565764 2,396965 1,473769 1,475057 7,766118	99e-13, 94e-11, 94e-11, 91e-15, 98e-16, 81e-05, 98e-08,	[2.197540996-12 2.151548146-12 2.340590776-13 4.54685776-11 1.067217606-15 9.73445486-15 2.276619486-05 1.831845506-05 1.337246336-03 7.766118036-10]	2 2 4 6 1 1 2 2	2.34187087e-12, 	[2.19785040 2.10545372- 2.1988978- 4.59255556- 5.21613948- 1.1790531e- 9.57213186- 2.3966921e- 1.477563985- 1.4590756- 7.62869106-	12, 13, 11, 15, 16, 55, 56,	[2.19755738e-12 - 2.14641418e-12 - 2.14641418e-12 - 2.14641418e-12 - 2.1468555e-13 - 4.7626118e-11 - 5.40973986-11 - 1.17509631e-15 - 2.3998981e-05 - 1.57376928e-05 - 1.45785196-08 - 7.76611803e-10 - 10 - 10 - 10 - 10 - 10 - 10 - 10	2.06354 2.34056 4.9075 5.27865 1.17905 8.72651 2.39696 1.67375 1.59427		

Diagram 2

Original Population	[2.15791278e 12: 2.00546787e 12: 2.00546787e 13: 2.15057271e 11: 2.15057271e 11: 2.1505721e 11: 2.1505721e 12: 2.1505721e 13: 2.177697127e 14: 2.177697127e 16: 2.177769726e 06: 1.457597799 1.457597990 07: 7.76611803e 10]	2.75650398e 12.72550398e 12.10503786e 12.134050209e 13.445561094e 11.17509611e 15.56578489e 16.57376928e 06.57559e 07.76511903e 10]	[12 - 2.185491616-12 - 2.24062055-13 - 2.24062055-13 - 2.24062055-13 - 2.24062055-11 - 1.175096116-15 - 2.2556908116-55 - 2.2556908100000000000000000000000000000000000	2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	2	[[[: 2.16747720- 12-; 2.5567871e- 2.43143567e- 13-478147111e- 11-; 2.546814620- 11-; 2.546814620- 11-; 2.546814620- 11-; 2.5468681e- 05- 06- 06- 06- 06- 06- 06- 06- 06- 06- 06	[[12.193859156 12.203854936 12.203854936 12.305178056 13.4618071616 11.5264793306 11.52659816 08.77574766 08.77574766 08.77574766 08.77574766 08.7757556 08.7757556 19.775756 19.7757556 19.775756 19	[1.2341870876-12. 2.05518596-11. 2.05518596-11. 2.05518596-11. 347902786-11. 3487902786-11. 3487902786-11. 3487902786-11. 3487902786-11. 355418786-0. 3487902786-0. 34879	[: 2.157250406-12: 2.157250406-12: 2.157250505-13: 2.35521595-3: 2.35521595-3: 2.35521595-3: 2.3552516-13: 2.355255-3: 2.355255-3: 2.355259106-10]	1.9578248e 12. 11731311e 2.37313259e 13. 4.7233078e 11. 5.37212548e 11. 1.09017700e 13. 1173490e 14. 2.39569516 06. 1.67776725e 06. 1.67776725e 06. 1.00099034e 10]	12 137540790- 12 137540740- 12 1351548140- 12 1340590770- 13 4546807920- 11 1507217600- 15 773454680- 16 1377454580- 00 137746538- 00 7766118093- 10 1	1,1421120- 2,1421120- 2,242120- 12- 2,1421120- 2,14221120- 4,0715119- 4,0715119- 1,1799511- 1,1799511-
Parents	13 4.59193221e-1 1.17909631e-15-9	2.02546787e 12 - 2.2 1 - 5.25092485e 11 - 57495737e 15 - 2.39 L45905755e 08 - 7.7	. 13 96969816-05 1.1	1.285903986-12-7.105037 4.636818046-11-7.2745 79096316-15-9.56576488 573769286-06-1.4590575	764e-11 - r-15 2.39696981e-05	13 4.80679585 1.17909631e-1	12 2 254573196 6e 11 5 36400459 5 1.007284956 15 36 4.459057596 0	e-11 - 2.39696981e-05	[2.19744753e-12 13 4.81537443e-1 1.17509531e-15 5 1.67376928e-05 1 10]	1 -5.36851935e-1 52913749e-16 -2	11 - 39696981e-05	[-2.391219776-12-2.292 13-4.798880896-11-5-5 1.179096316-15-1.05146 1.673769786-06-1.4590 10]	5035300e 11 531e 15 2.396969	13 4.5855 81e-05 1.1790963	:126-12-2.026512446 18526-11-5.6245234 16-15-1.025017966-1 86-06-1.459057596-1	60-11 - 5 Z.39898981e-0 5
Crossover Children	[2.134540706-12 2.114825056-12, 2.33953576-13, 4.235440746-11, 5.230927656-11, 1.179096316-15, 9.75823456-16, 2.39696916-05, 1.455057596-08, 7.766118036-10	2.17: 2.34: 4.57: 5.59: 1.17: 9.42: 2.39: 1.67: 1.32:	2709146e-12 271336e-12 271336e-12 811337e-13 915844e-11 660455e-11 909631e-15 9057031e-16 696631e-05 119095e-06 119095e-08 696666e-10	[2.784970546-12 2.10155955-12, 2.41475496-13, 4.41331456-11, 5.265436006-11, 1.26712388-15, 9.568139046-16, 2.55489454-06, 1.54384454-06, 1.45907596-08, 7.766118006-10]	2,460 2,287 4,665 5,555 1,179 1,853 2,386 1,688	Hisito 17, 18372e 12, 19372e 12, 19373e 13, 19382e 11, 46451e 11, 19561le 15, 19382e 15, 19382e 15, 19382e 15, 19382e 15, 19382e 16, 19382e 16	[-2.198907 2.1770933 2.3380854 4.58577285 5.2476605 1.2366854 9.5600673 2.4610951 1.6737692 1.4590575 7.2225156	5e 12 4e 13, 5e 11, 7e 11, 5e 15, 8e 16, 5e 05, 8e 08,	[7.16238732e-12, 2.76115884e-12, 2.4253227e-13, 4.8129925-11, 1.17909631e-15, 9.24563846-16, 2.3969681e-05, 1.5604487e-05, 1.45905759-08, 7.76611803e-10]	2 2 4 5 1 9 2 1 1	2 23534473e 12, 25319014e 12, 33541447e 13, 52841076e 11, 53111657e 11, 22904745e 15, 5317800e 16, 11540814e 05, 50787233e 08, 22350430e 10	[2.1572/478 2.26196356 2.34959126 4.81593176 5.331811156 1.17996316 1.00456546 2.3965916 1.63882016 1.45995756 7.429917936	12, 13, 11, 15, 15, 06, 06,	[2.19796730e 12, 2.07647798e 12, 2.34501440e 12, 4.59045401e 11, 5.37640416e 11, 1.14526360e 15, 9.59685500e 16, 2.3968631e 05, 1.51005994e 06, 1.45905758e 08, 7.76611803e 10]	[2.1367] 2.02650 2.23955 4.58699 5.53976 1.25096 1.09700 2.62490 1.6736 1.43505 7.574366	756-13, 7726-11, 726-11, 746-15, 846-15, 356-05, 728-06,
Mutated Children	[-2.12494070e-12, 2.11462595e-12, 2.33963857e-13, 4.2844974e-11, 5.23092765e-11, 1.17909631e-15, 9.75682345e-16, 2.39696981e-05, 1.67376522e-06, 1.455057596-08, 7.76611803e-10]	2.17: 2.34 4.57: 5.59 1.17: 9.42: 2.39: 1.67: 1.32:	187091466-12 3713366-12 3713366-12 3113396-13 9153446-11 6804556-11 5906316-15 6906316-05 3759728-06 1950956-08 6906566-10	3.72467764e-12 2.1015995e-12, 2.4447549e-13, 4.63281445e-11, 5.264546001, 5.54819904e-16, 2.5548904e-05, 1.55384484e-06, 1.4559759e-08, 7.76611893e-10]	2,460 2,287 4,605 5,555 1,179 1,053 2,396 1,655 1,650	H15110e-12, 1877e-12, 9073f-13, 19082e-11, 46461e-11, 99631e-15, 96891e-95, 96929-95, 54492e-05, 11803e-10]	[.2.19850] 2.1720931 2.3390254 4.58577.88 5.2476601 1.2366364 5.5620673 2.4010951 1.6737693 1.4550575 7.2225156	56 12, 46 13, 56 11, 56 15, 56 15, 56 15, 56 05, 56 08,	[2.162387326-12, 2.261158546-12, 2.42532276-13, 4.81299056-11, 1.179094316-15, 9.24536468-16, 2.395908816-05, 1.56604876-06, 1.455057596-08, 7.766118036-10]	2. 4. 5. 1. 9. 2. 1.	2 23534473e-12, 23315014e-12, 33641447e-13, 67241076e-11, 53111657e-11, 23904745e-15, 5317800e-16, 31940824e-05, 5999200e-06, 50287233e-08, 22350430e-10]	[2.15777478 2.26199352 2.49197126 4.815301375 5.33181135 1.179093126 1.00454564 2.39698126 1.638820126 1.638820155 7.42917536	12, 13, 11, 11, 15, 15, 06, 06,	[2.19796730e-12, 2.02647798e-12, 2.34701440e-13, 4.5904840e-11, 1.14725340e-15, 9.5968580e-16, 2.3969816-05, 1.149305756-06, 1.45905756-06, 7.76511803e-10]	[-2.1367. 2.076500 2.278393. 4.586999 5.539761 1.256996 1.009700 2.674900 1.455057 7.574360	756-13, 776-11, 776-11, 1446-15, 1846-15, 1396-05, 1286-06,

Diagram 3

Original Population	(2.5.5%)/06 1.11481505-0 1.11481505-0 1.2390587-0 1.2390587-0 1.2390587-0 1.2390587-0 1.17909611-0 1.579031405-0 1.17909611-0 1.0590971-0 06 1.45905799-0 07.766115036-1 10	(2.19791798 (2.19791798 (2.107917919 (2.1079179 (2.1079179 (2.1079179 (2.1079179 (2.107919 (2.1079179 (2.1	(2.2 2.155377656 12: 2.345502056- 13: 3.445502056- 13: 4.455019046- 11: 5.274547646- 11: 1.179096116- 15: 5.55776488- 16: 2.379696816- 05: 1.47957758- 08: 7.766118036- 10]	(2.16.167.09 (2.16.167.19 1.76.17715-1 1.76.17715-1 1.76.17715-1 1.76.17715-1 1.76.17715-1 1.76.17715-1 1.76.17715-1 1.76.17715-1 1.76.17715-1 1.76.17715-1 1.76.17715-1 1.76.17715-1 1.76.17715-1 1.76.17715-1 1.76.17715-1	1.18744753e 1.18349151e 2.18349151e 2.234962055e 1.113 1.113 1.115 1.115 1.11743e 1.11749e 1.	2, 2391219776 13- 12- 2252715216 13- 13- 13- 13- 13- 13- 13- 13- 13- 13-	2. 13615312e 12. 12. 12. 12. 12. 12. 12. 12. 12. 12.	[13713316 1313502336 12 12 1315502356 131 435096776 131 137760246 11 11 137760246 13 1474760256 06 1474760256 06 1474760256 06 1474760256 06 1474760256 06 1474760256 06 1474760256 06 1474760256 06 1474760256 06 1474760256	1,13703146e 13,1371336e 13,1371336e 13,23713544e 11,23713544e 11,1370311e 13,43057011e 14,43057011e 16,43057011e 16,13716936e 16,13716936e 16,13716936e 16,13716936e 16,13716936e 16,13716936e 16,13716936e 16,13716936e 16,13716936e 16,13716936e 16,13716936e 16,13716936e	12.39315210e 12.240018872e 12.240018872e 12.22700737e 13.460519082e 11.555545463e 11.17909611e 15.23556541e 05.1605492e 06.1605492e 06.776611803e 10]	2.157274786- 12- 2.261369356- 12- 2.340502126- 13- 4.315203176- 11- 1.379056216- 15- 1.379056216- 06- 1.459057556- 08- 7.42917936- 10	1333473e- 12-3919014e- 12-3919014e- 13- 13-581107e- 11- 11- 11- 11- 11- 11- 11- 11- 11- 1	2.19800965e 12. 2.17709136e 12. 2.13808544e 13. 4.58572285e 11. 1.23808646e 13. 5.58706738e 16. 2.40109516e 05. 1.459057596 06. 1.459057596 06. 7.22251565e 10]	1 1 24518732-11 2 11 2 11 2 11 2 11 2 11 2 11 2 11	219796730e- 12 219796730e- 12 2192947798e- 12 21929440e- 13 439945401a- 11 11 2192940e- 13 519696981a- 06 13 13095981a- 06 145905759e- 08 179611803e- 10 1	1 22. 12. 12. 12. 12. 12. 12. 12. 12. 12
Parents	13 4.28644074e 1 1.17909631e 15 9	·2 11482505e-12 ·2 1 · 5.23092765e-11 1.75683495e-15 ·2.39 1.45905759e-08 ·7.7	6969816-05 1 6611803e - 1	2 19791278e-12 - 2.02546 8 4.59193221e-11 - 5.256 17909631e-15 9.574997 57275928e-06 - 1.45905 0)	92485e-11 - 17e-15 2.39696981e-0	13 4.636811 05 1.179096314	5e 12 -2.10503786e 934e 11 -5.2745476 e 15 -9.56578488e 18 e 06 -1.45905759e 0	le-11 - 2.39696981e-05	13 4.80679589e-1 1.17909631e-15 1	2 2.25457319e-12 / 11 536400459e-11 1.00228495e-15 2.3 1.45905759e-08 7.	19696881e 05 1 76611803e	2 15744753e-12 - 2183 3 4 81537443e-11 - 636 17909631e-15 - 952913 1 67376578e-06 - 1 4590 0)	5851935e-11 + 1749e-16 - 2.3969698	13 4.7988 1e-05 1.1790963	977e-12-2-292715216 8089e-11 -5.5503530 1e-15 1.05146531e-1 88e-06 -1.45905759e	06-11 - 5 2.39696981e-0 5
Crossover Children	[2,38987112e12, 2,02710692e12, 2,47988138e11, 5,08755853e11, 5,43792155e11, 1,25419015e15, 2,560225e16, 2,3908091e05, 1,5896922e0, 1,42971105e00, 7,76611893e10]	2.21/ 2.33 4.27/ 5.22/ 1.17/ 9.76/ 2.32/ 1.67/ 1.45/	9454867e 12, 177017e 13, 1961841e 13, 135752e 11, 195057e 11, 195051e 15, 195051e 15, 11452e 05, 176728e 06, 197755e 08, 11803e 10	[2,281638266] 2,982779576-12 2,379885770-15 5,075757436-11 1,17909316-11 2,38689816-0 1,67376928-0 1,459057936-11 7,648558946-11	19 23 46 48 11 95 23 16	145670856-12 115606336-12 40480766-13 00476596-11 11498956-11 175415276-15 96968316-05 77769286-06 59057596-08 81005746-10	[-2,14950 2,26574 2,40315; 4,857093 5,37543; 1,215827; 1,001569 2,395968 1,673768; 1,459057; 7,418521;	676 17 606 13 846 11 666 11 686 15 676 15 816 05 826 06 836 08	[2,22123678612, 1,89685366612, 2,32248877613, 1,94416745611, 5,21949503611, 1,17999631645, 8,52600622616, 2,56176922605, 1,72246715606, 1,47244637603, 7,63047872610]	2.1 2.3 4.0 5.2 1.2 9.7 2.3 1.6	49282556e 12, 1400374e 12, 8552159e 13, 4950357e 11, 3458810e 11, 0815654e 15, 0808262e 16, 9598821e 05, 7376928e 08, 5905755e 08,	[2.2909590 2.03618776 2.34043130 4.657415476 5.270887186 1.179996336 9.58181720 2.396968316 1.673768726 1.512243026 7.746118036	12, 13, 11, 15, 16, 05, 08,	[2,19752877612 2,18344785612 2,34051994613 5,24077987611 1,23256832615 2,5523235616 1,3969831605 1,57376328606 1,45905753608 1,45905753608 1,7666118036-10	[-2.1248 2.13139 2.13617 4.68562 5.23301 1.27244 1.63311 2.39683 1.67376 1.60104 7.34473	163e-13, 173e-11, 153e-11, 153e-15, 152e-15, 152e-06, 114e-08,
Mutated Children	[238987112e-12, 2.02210697e-12, 2.42968358e-13, 5.06768393e-11, 5.47792155e-11, 1.25412015e-15, 9.5602250e-16, 2.35969321e-05, 1.83696222e-06, 1.42971106e-08, 7.76611803e-10]	2.21/ 2.33 4.27/ 5.22/ 1.17/ 9.76/ 2.32/ 1.67/	8454867e-12, 177017e-12, 961841e-13, 136797e-11, 199087e-11, 109631e-15, 509977e-16, 11457e-05, 376928e-05, 507759e-08, \$11893e-10	[2.78163680- 2.0827755746-11 2.97385776-13 5.07575746-11 1.1790831-1 2.3856891-03 1.57375758-0 1.47395759-0 7.4455894-11	15 23 46 48 11 95 23 16	1.6567089e-12. 11560633e-12. 4040076e-13. 0047659e-11. 1146895e-11. 7441877e-15. 7738167e-16. 9696531e-05. 77376522e-06. 5506759e-08.	[.2.14950 2.266574 2.430315; 4.867053 5.375433 1.215227 1.004569 2.3969655 1.6737695 1.459057 7.4186211	576-12 706-13, 966-11, 166-11, 166-15, 176-15, 116-05, 128-06, 196-08,	[-7.271236786-12. 1.896851656-12. 2.332488776-13. 1544167456-11. 5.219499086-11. 1.173967316-15. 8.826006236-16. 2.561769226-05. 1.722467156-05. 1.478446876-08. 7.630478772-16]	2.1 2.3 4.0 5.2 1.7 9.7 2.3 1.6 1.4	49727556e-12, 1400174e-12, 0552199e-13, 0552199e-13, 1458810e-11, 1458810e-11, 9696981e-05, 73776928e-06, 5500755e-06, 2964696e-10]	[2.2909550 2.03618727e 2.340951100e 4.60741547e 5.27088712e 9.55181720e 2.356569512e 1.67376522e 1.5124392e 7.76611833e	12, 13, 11, 11, 15, 16, 05, 06,	[2.197528776-12. 2.183447856-12. 2.340519946-13. 5.240739876-11. 5.140442476-11. 1.27268820-15. 9.57528258-16. 2.39698216-05. 1.459657586-08. 7.766118036-10]	[-2.1245] 2.13130 2.11632 4.69567 5.23103 1.77244 1.03311 2.39695 1.57376 1.50104 7.34473	163e 13, 173e 11, 153e 11, 153e 15, 152e 15, 151e 15, 152e 06,

Statistics

The top vectors were chosen on the basis of how much they were able to reduce the train and validation error and subsequently, the total error, along with their ability to reduce the absolute difference between the train error and validation error. The absolute difference between train and validation error is roughly 0.27 percent of the train error. The train error is roughly 49.9 percent of the total error which we believe is a fairly even split.

TOP VECTORS:

[-2.1343305086353493e-12, -2.2896363962565298e-12, -2.255677140167976e-13, 5.097975671687973e-11, -5.174042346093453e-11, -1.1790963051945613e-15, 1.0359193610109647e-15, 2.3969698057747152e-05, -1.6737692764075564e-06, -1.4590575921693652e-08, 7.766118028201611e-10]

TRAIN ERROR: 6.484964039445597E+10

VALIDATION ERROR: 6.467263187502826E+10

TOTAL ERROR: 1.29522E+11

DIFFERENCE IN ERRORS: 1.770085194E+8

We used a total of 8000 - 10 000 API calls in order to generate roughly 400 generations of the initial population. This is exclusive of the datapoints which led to flawed approaches or unacceptable errors.