MDL Project

The Team (87)

March 17, 2021

1 Genetic Algorithm

1.1 Summary

Genetic algorithms are inspired by the process of natural selection, and our genetic algorithm uses the same principle. The first generation is initialized by randomly slightly increasing or decreasing the values in the given overfit vector. The future generations are generated by combining parents picked from this generation, with their probability of being picked depending on their evaluated fitness. These children are also randomly mutated by a mutation function to increase diversity. This process is repeated multiple times till satisfactory vectors are obtained.

1.2 Pseudocode

1. Initialize first generation of population size n by perturbing overfit vector,
2. Repeat till convergence
   (a) Calculate fitness based on errors using fitness function
   (b) Pick parents from pool size s
   (c) Generate n children with a crossover function
   (d) Mutate children with mutation function
   (e) Set generated children as next population

1.3 Algorithm and code snippets

1.3.1 Initializing first generation

The first generation is initialized by slightly perturbing the values in the given overfit vector. The values are randomly increased or decreased by a percentage. We also manually experimented with different values of changes at different indices.

```
population = np.zeros(shape = (n, 11))
    for i in range(n):
        rng = np.random.uniform(low = low_limit, high = high_limit, size=(1, 11))
        population[i, :] = overfit_vector + rng*overfit_vector
```

1.3.2 Generating next generations

The next generations are generated by first pruning the parent generation to a pool size based on their fitness, and then picking parents from this pool. The children are generated with a crossover function which takes both the parents as input and then randomly mutating them. We also retained some of the best parents between generations, which is explained in the heuristics section. The following code snippet shows the procedure for generating one generation of children.

```
def get_next_pop(initial_pop, fitness, errors, n_parents = n_parents):
    log = {"initial_pop": initial_pop, "mutations": False, "parents": [], "next_pop
    ": []}

    triple = sorted(list(zip(initial_pop, fitness, errors)), key = lambda x:  x[1])
    fitness = np.array(sorted(fitness))
    p = [(1/i)/np.sum(1/fitness[:pool_size]) for i in fitness[:pool_size]]

    next_fitness = []
    next_errors = []
    next_pop = []
    parents = []
    mutations = []

    for _ in range(population-n_parents):
        parent1 = np.array(triple[np.random.choice(range(pool_size), p = p)][0])
        parent2 = np.array(triple[np.random.choice(range(pool_size), p = p)][0])
        parents.append((parent1, parent2))

        child = crossover(parent1, parent2)
        if(np.random.uniform() < .5 or (list(parent1) == list(parent2))):
            mutations.append(child)
            child = mutation(child, prob = 1)
        else:
            mutations.append(False)

        c_fitness, c_error = fitness_function(child)
        next_fitness.append(c_fitness)
        next_errors.append(c_error)
        next_pop.append(child)

    for i in range(n_parents):
        # print(i, initial_pop[i], errors[i])
        next_pop.append(np.array(triple[i][0]))
        next_fitness.append(triple[i][1])
        next_errors.append(triple[i][2])

    log["errors"] = next_errors
    log["fitness"] = next_fitness
    log["parents"] = parents
    log["next_pop"] = next_pop
    log["mutations"] = mutations

    f = open(f'{path}/{time.time()}.txt', 'wb')
    pickle.dump(log, f)
    return log
```

2 Functions

2.1 Fitness function

The fitness function used is a weighted sum of the train and validation error. The weight of train error was modified between generations according to our need, which is explained in the heuristics section. Another type of fitness function we used is given in the comment code. We hypothesized that this would allow our algorithm to pick errors closer to a target error, and reduced the target error as the generations progressed further. A combination of the two error functions was used.

```
def fitness_function(vector, weight):
    errors = get_errors(key, list(vector))
    fitness = abs(weight*errors[0]+errors[1])
    #fitness = (abs(errors[1]-5e10) +abs(errors[0]-5e10))
    return fitness, errors
```

2.2 Crossover function

The crossover function used is **blend crossover**. Given parents p_1 and p_2 $(p_2 > p_1)$, it randomly picks children in the range $[p_1 - \alpha(p_2 - p_1), p2 + \alpha(p_2 - p_1)]$ with a uniform distribution.

```
def crossover(parent1, parent2):
    alpha = 0.5
    child = np.zeros(11)
    for i in range(11):
        max_p = max(parent2[i], parent1[i])
        min_p = min(parent2[i], parent1[i])
        low = min_p - alpha*(max_p - min_p)
        high = max_p + alpha*(max_p - min_p)

        child[i] = np.random.uniform(low = low, high = high)
    return child
```

2.3 Mutation function

The mutation function randomly increases or decrease the value at an index by a certain small percentage. This introduces more diversity in the generations. Depending on the requirement, this range of modifications was from as little as .005 to as large as .5

```
def mutation(vector, prob = index_prob):
    rng = (np.random.uniform(size = 11)) < prob
    delta = np.random.uniform(low = low_limit, high = high_limit, size = 11)
    vector = vector + delta*vector
    return vector
```

3 Hyperparameters

- Population size - This was initially chosen to be 20. However, it was gradually reduced to 10 as we obtained better generations and to preserve requests.
- Pool size - This ranged from 5-10 based on the population size. Half the population was pruned. This was done because it results in faster convergence.

- Alpha - Based on recommended choices of alpha for blend crossover in our research, we intially chose alpha as 0.5. This was modified to reduce or increase the range of values generated depending on performance.
- Parents retained - Between 1-4 parents were retained between generations. We did not want to make this value too high so as to ensure that diversity is maintained.
- Mutation probability - The mutation probability ranged from .3 to .7 depending on how fast the error was decreasing between generations. We increased this if the error was dropping too slowly or if we got stuck in a local minima.

4 Heuristics

- Weighted sum as fitness function - As explained above, the fitness function we used was a weighted sum of the errors. This was done to prioritize train or validation error at different errors. Assigning a higher weight to validation error allowed us to reduce it quicker, while using equal weights allowed us to bring the values together. By assigning the weight as -1, we also used the absolute difference between the two errors as a fitness metric. This allowed our genetic algorithm to generate vectors that minimize the difference between the train and validation error, minimizing the chance of them overfitting to any particular dataset.
- Parents retained - The parents retained between generations allowed us to ensure that at the very least the performance does not degrade between generations. However, we took care to keep this value small, to ensure that diversity isn't compromised. On occasions, the parents retained continued achieving the best fitness for multiple generations. This was undesirable, as it indicated that the newer generations were not performing well. These iterations were scrapped.
- Mutation probability - The mutation probability was varied between various generations. When the performance wasn't improving, we increased the probability to introduce more diversity in the generations. We also changed the range of variation for certain indices manually according to our estimate of performance. A blanket range of change for all indices was tried earlier, but this resulted in erratic performance as certain indices required much finer modifications that other.
- Through observation and manual experimentation, we observed that even drastic change in values at certain indices did not affect the error in any appreciable way. We focused our efforts on fine-tuning the value of the other indices. Initializing the first generation on this basis was tried, but it did not result in any significant changes to performance.

5 Results

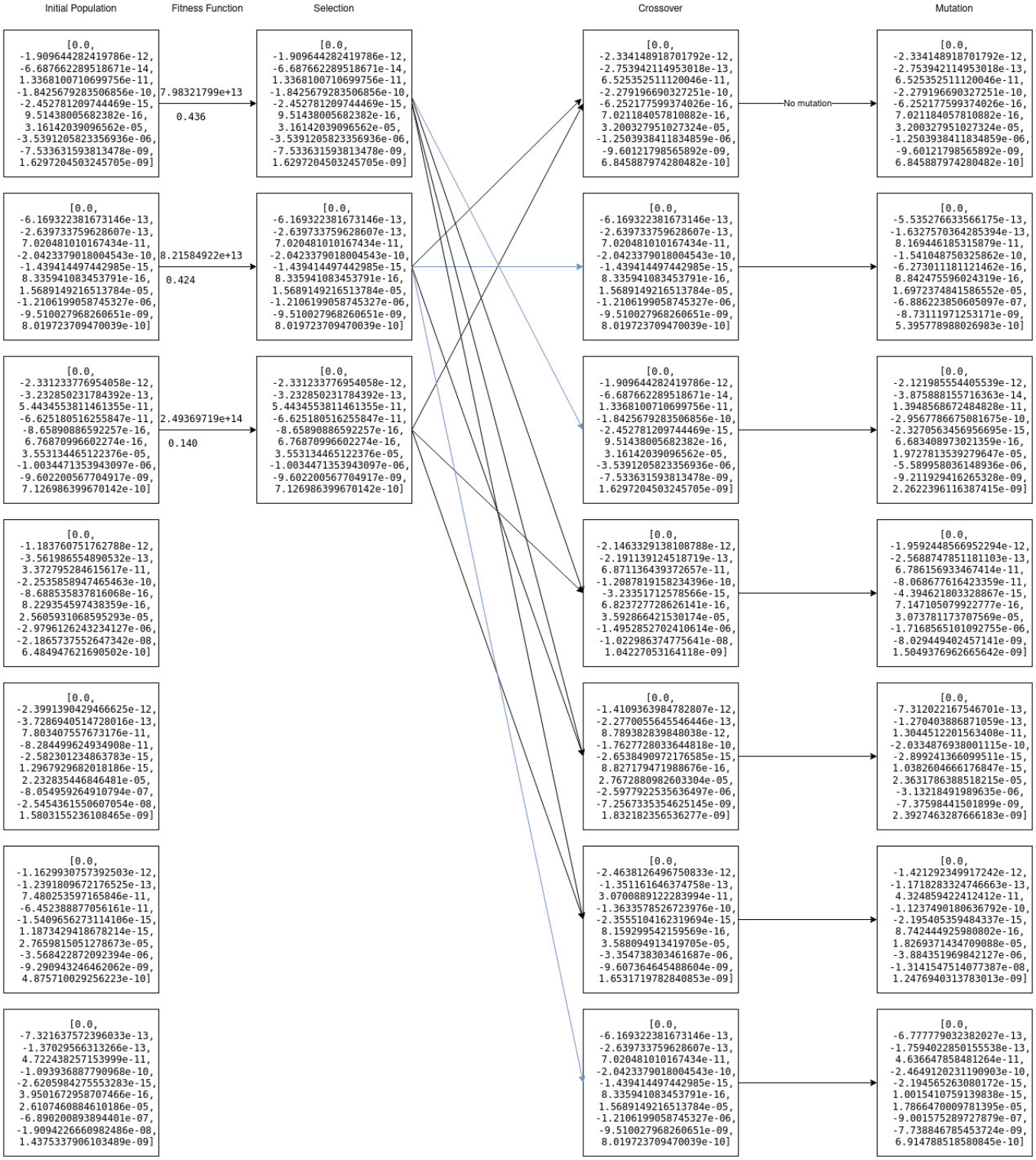
1. The best errors we obtained for a single vector are -
   - Train Error - $0.74e11$
   - Validation - $0.83e11$

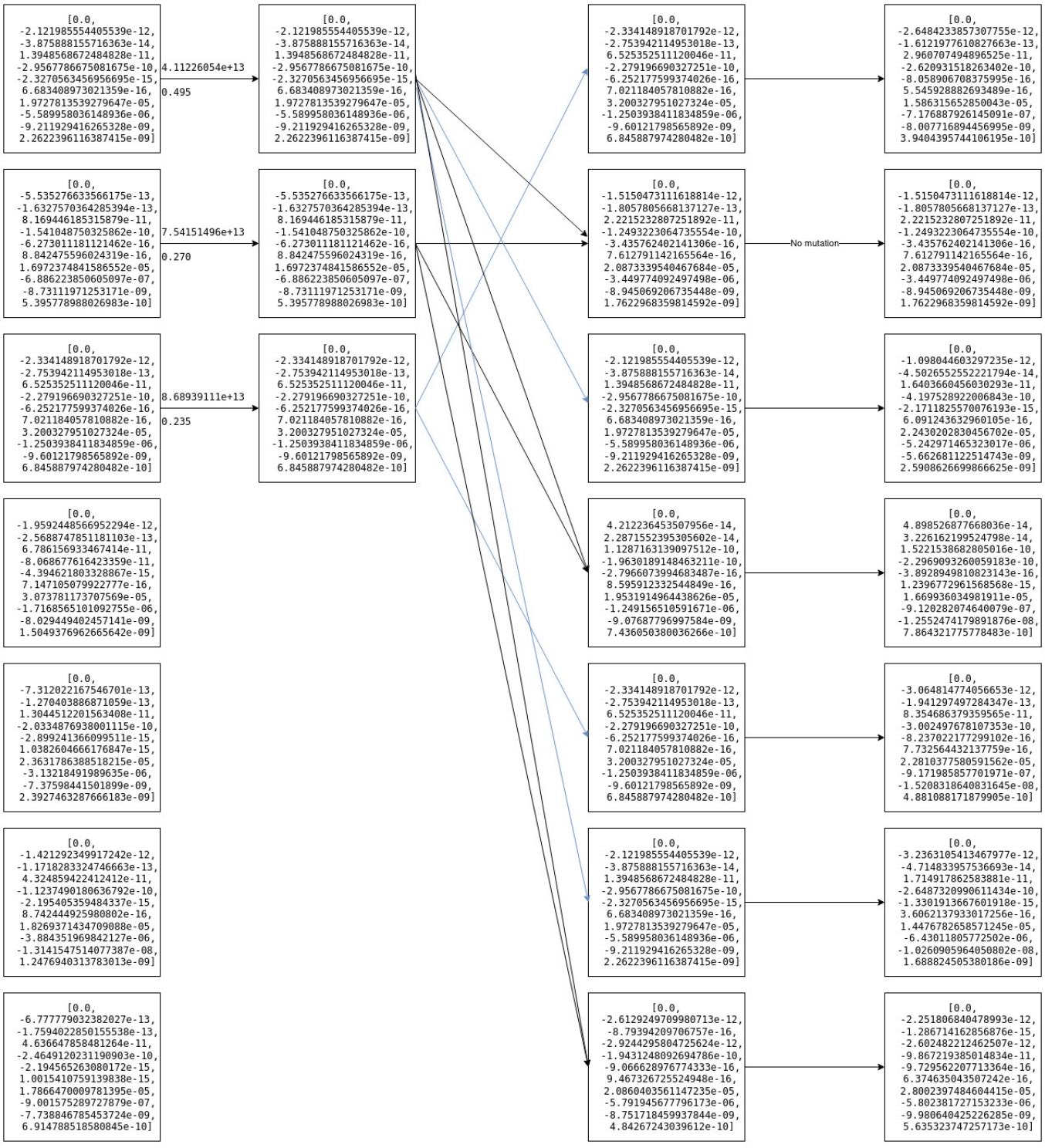
   The overfit vector we started with had much lower train error and higher validation error. That is a pattern observed when the algorithm overfits to the training set and does not generalize well to unseen data. By contrast, the vectors our genetic algorithm was able to generate had lower validation error and higher training error, indicating that these vectors no longer overfit the training set. These values are also close to each other, meaning that similar performance is

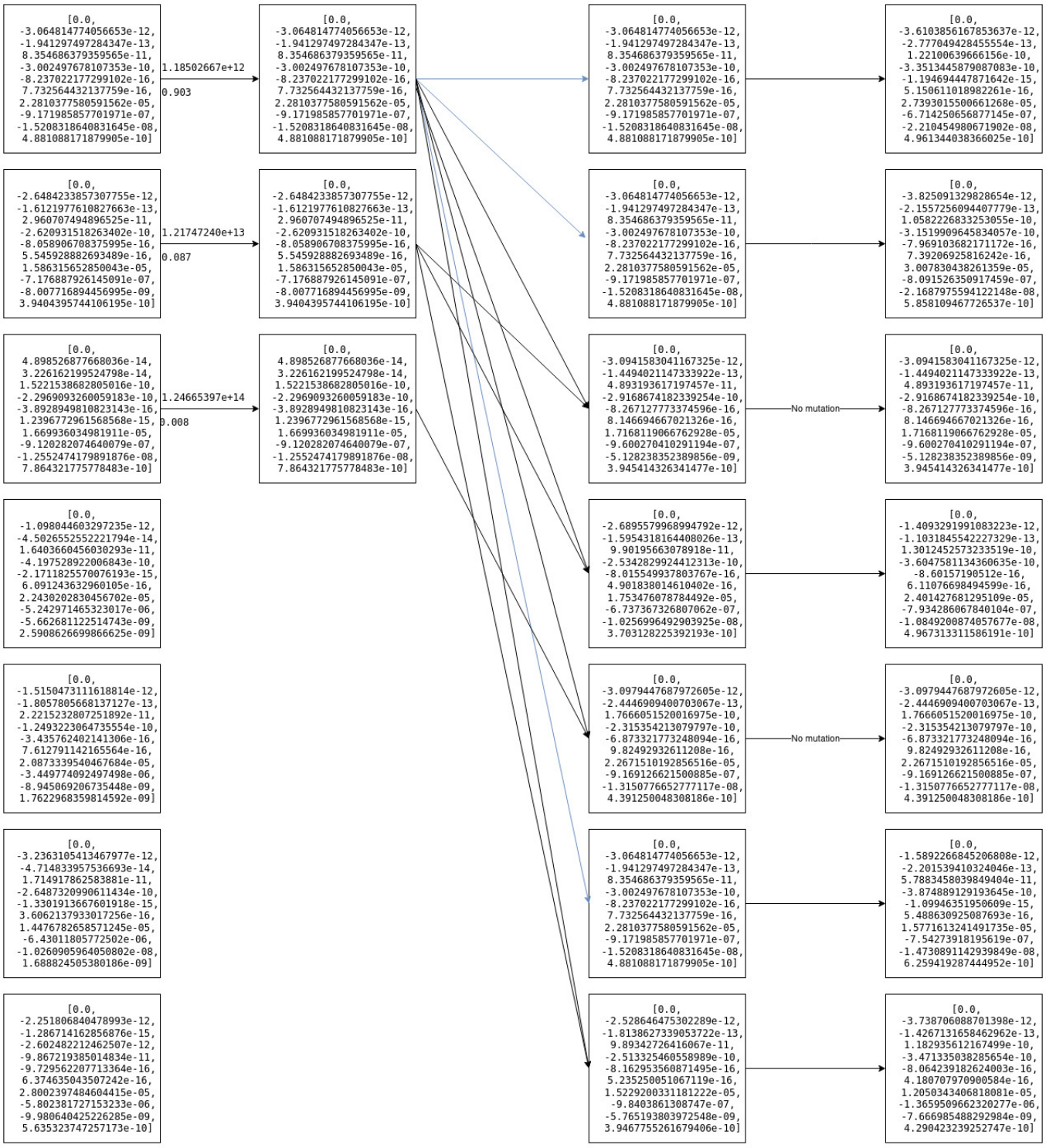
being attained in different datasets. This should generalize well to unseen data. We also obtained vectors which had much lower validation errors, but these had likely overfit to the validation data.

2. The errors converged in around 70 generations. Further generations allowed us to finetune the parameters.

6 Illustration




[0.0,
-2.1219855544055539e-12,
-3.8758881557163636e-14,
1.3948568672484828e-11,
-2.9567786675081675e-10,
-2.3270563456956695e-15,
6.683408973021359e-16,
1.9727813539279647e-05,
-5.589958036148936e-06,
-9.211929416265328e-09,
2.2622396116387415e-09]
4.11226054e+13
0.495
[0.0,
-2.1219855544055539e-12,
-3.875888155716363e-14,
1.3948568672484828e-11,
-2.9567786675081675e-10,
-2.3270563456956695e-15,
6.683408973021359e-16,
1.9727813539279647e-05,
-5.589958036148936e-06,
-9.211929416265328e-09,
2.2622396116387415e-09]
[0.0,
-2.334148918701792e-12,
-2.753942114953018e-13,
6.525352511120046e-11,
-2.279196690327251e-10,
-6.252177599374026e-16,
7.021184057810882e-16,
3.200327951027324e-05,
-1.2503938411834859e-06,
-9.60121798565892e-09,
6.845887974280482e-10]
[0.0,
-2.6484233857307755e-12,
-1.6121977610827663e-13,
2.960707494896525e-11,
-2.620931518263402e-10,
-8.058906708375995e-16,
5.545928882693489e-16,
1.586315652850043e-05,
-7.176887926145091e-07,
-8.007716894456995e-09,
3.9404395744106195e-10]
[0.0,
-5.535276633566175e-13,
-1.6327570364285394e-13,
8.169446185315879e-11,
-1.541048750325862e-10,
-6.273011181121462e-16,
8.842475596024319e-16,
1.6972374841586552e-05,
-6.886223850605097e-07,
-8.73111971253171e-09,
5.395778988026983e-10]
7.54151496e+13
0.270
[0.0,
-5.535276633566175e-13,
-1.6327570364285394e-13,
8.169446185315879e-11,
-1.541048750325862e-10,
-6.273011181121462e-16,
8.842475596024319e-16,
1.6972374841586552e-05,
-6.886223850605097e-07,
-8.73111971253171e-09,
5.395778988026983e-10]
[0.0,
-1.5150473111618814e-12,
-1.8057805668137127e-13,
2.2215232807251892e-11,
-1.2493223064735554e-10,
-3.435762402141306e-16,
7.612791142165564e-16,
2.0873339540467684e-05,
-3.449774092497498e-06,
-8.945069206735448e-09,
1.7622968359814592e-09]
No mutation
[0.0,
-1.5150473111618814e-12,
-1.8057805668137127e-13,
2.2215232807251892e-11,
-1.2493223064735554e-10,
-3.435762402141306e-16,
7.612791142165564e-16,
2.0873339540467684e-05,
-3.449774092497498e-06,
-8.945069206735448e-09,
1.7622968359814592e-09]
[0.0,
-2.334148918701792e-12,
-2.753942114953018e-13,
6.525352511120046e-11,
-2.279196690327251e-10,
-6.252177599374026e-16,
7.021184057810882e-16,
3.200327951027324e-05,
-1.2503938411834859e-06,
-9.60121798565892e-09,
6.845887974280482e-10]
8.68939111e+13
0.235
[0.0,
-2.334148918701792e-12,
-2.753942114953018e-13,
6.525352511120046e-11,
-2.279196690327251e-10,
-6.252177599374026e-16,
7.021184057810882e-16,
3.200327951027324e-05,
-1.2503938411834859e-06,
-9.60121798565892e-09,
6.845887974280482e-10]
[0.0,
-2.1219855544055539e-12,
-3.875888155716363e-14,
1.3948568672484828e-11,
-2.9567786675081675e-10,
-2.3270563456956695e-15,
6.683408973021359e-16,
1.9727813539279647e-05,
-5.589958036148936e-06,
-9.211929416265328e-09,
2.2622396116387415e-09]
[0.0,
-1.098044603297235e-12,
-4.5026552552221794e-14,
1.6403660045603029e-11,
-4.197528922006843e-10,
-2.1711825570076193e-15,
6.091243632960105e-16,
2.2430202830456702e-05,
-5.242971465323017e-06,
-5.662681122514743e-09,
2.5908626699866625e-09]
[0.0,
-1.9592448566952294e-12,
-2.5688747851181103e-13,
6.786156933467414e-11,
-8.068677616423359e-11,
-4.394621803328867e-15,
7.147105079922777e-16,
3.073781173707569e-05,
-1.7168565101092755e-06,
-8.029449402457141e-09,
1.5049376962665642e-09]
[0.0,
4.212236453507956e-14,
2.2871552395305602e-14,
1.1287163139097512e-10,
-1.9630189148463211e-10,
-2.7966073994683487e-16,
8.595912332544849e-16,
1.9531914964438626e-05,
-1.249156510591671e-06,
-9.07687796997584e-09,
7.436050380036266e-10]
[0.0,
4.898526877668036e-14,
3.226162199524798e-14,
1.5221538682805016e-10,
-2.2969093260059183e-10,
-3.8928949810823143e-16,
1.2396772961568568e-15,
1.6699360349811911e-05,
-9.120282074640079e-07,
-1.2552474179891876e-08,
7.864321775778483e-10]
[0.0,
-7.312022167546701e-13,
-1.2704038868710 59e-13,
1.3044512201563408e-11,
-2.0334876938001115e-10,
-2.899241366099511e-15,
1.0382604666176847e-15,
2.3631786388518215e-05,
-3.132184919896 35e-06,
-7.37598441501899e-09,
2.3927463287666183e-09]
[0.0,
-2.334148918701792e-12,
-2.753942114953018e-13,
6.525352511120046e-11,
-2.279196690327251e-10,
-6.252177599374026e-16,
7.021184057810882e-16,
3.200327951027324e-05,
-1.2503938411834859e-06,
-9.60121798565892e-09,
6.845887974280482e-10]
[0.0,
-3.064814774056653e-12,
-1.941297497284347e-13,
8.354686379359565e-11,
-3.002497678107353e-10,
-8.237022177299102e-16,
7.732564432137759e-16,
2.2810377580591562e-05,
-9.171985857701971e-07,
-1.5208318640831645e-08,
4.881088171879905e-10]
[0.0,
-1.421292349917242e-12,
-1.1718283324746663e-13,
4.324859422412412e-11,
-1.1237490180636792e-10,
-2.195405359484337e-15,
8.742444925980802e-16,
1.8269371434709088e-05,
-3.884351969842127e-06,
-1.3141547514077387e-08,
1.2476940313783013e-09]
[0.0,
-2.1219855544055539e-12,
-3.875888155716363e-14,
1.3948568672484828e-11,
-2.9567786675081675e-10,
-2.3270563456956695e-15,
6.683408973021359e-16,
1.9727813539279647e-05,
-5.589958036148936e-06,
-9.211929416265328e-09,
2.2622396116387415e-09]
[0.0,
-3.2363105413467977e-12,
-4.714833957536693e-14,
1.7149178625838 81e-11,
-2.6487320990611434e-10,
-1.3301913667601918e-15,
3.6062137933017256e-16,
1.4476782658571245e-05,
-6.430118057752502e-06,
-1.0260905964050802e-08,
1.688824505380186e-09]
[0.0,
-6.7777790323820 27e-13,
-1.7594022850155538e-13,
4.636647858481264e-11,
-2.4649120231190903e-10,
-2.194565263080172e-15,
1.0015410759139838e-15,
1.7866470009781395e-05,
-9.001575289727879e-07,
-7.738846785453724e-09,
6.914788518580845e-10]
[0.0,
-2.6129249709980713e-12,
-8.79394209706757e-16,
-2.9244295804725624e-12,
-1.9431248092694786e-10,
-9.066628976774333e-16,
9.467326725524948e-16,
2.0860403561147235e-05,
-5.791945677796173e-06,
-8.751718459937844e-09,
4.84267243039612e-10]
[0.0,
-2.251806840478993e-12,
-1.2867141628568 76e-15,
-2.6024822124625 07e-12,
-9.867219385014834e-11,
-9.729562207713364e-16,
6.374635043507242e-16,
2.8002397484604415e-05,
-5.802381727153233e-06,
-9.980604025226285e-09,
5.635323747257173e-10]


[0.0, -3.0648147740566 53e-12, -1.941297497284347e-13, 8.354686379359565e-11, -3.002497678107353e-10, -8.237022177299102e-16, 7.732564432137759e-16, 2.2810377580591562e-05, -9.171985857701971e-07, -1.5208318640831645e-08, 4.881088171879905e-10]
1.18502667e+12
0.903
[0.0, -3.0648147740566 53e-12, -1.941297497284347e-13, 8.354686379359565e-11, -3.002497678107353e-10, -8.237022177299102e-16, 7.732564432137759e-16, 2.2810377580591562e-05, -9.171985857701971e-07, -1.5208318640831645e-08, 4.881088171879905e-10]
[0.0, -3.0648147740566 53e-12, -1.941297497284347e-13, 8.354686379359565e-11, -3.002497678107353e-10, -8.237022177299102e-16, 7.732564432137759e-16, 2.2810377580591562e-05, -9.171985857701971e-07, -1.5208318640831645e-08, 4.881088171879905e-10]
[0.0, -3.6103856167853637e-12, -2.777049428455554e-13, 1.221006396661 56e-10, -3.3513445879087083e-10, -1.1946944478716 42e-15, 5.150611018982261e-16, 2.7393015500661268e-05, -6.7142506568771 45e-07, -2.2104549806719 02e-08, 4.961344038366025e-10]
[0.0, -2.6484233857307755e-12, -1.6121977610827663e-13, 2.960707494896525e-11, -2.6209315182634 02e-10, -8.0589067083759 95e-16, 5.545928882693489e-16, 1.5863156528500 43e-05, -7.1768879261459 91e-07, -8.0077168944569 95e-09, 3.9404395744106195e-10]
1.21747240e+13
0.087
[0.0, -2.6484233857307755e-12, -1.6121977610827663e-13, 2.960707494896525e-11, -2.6209315182634 02e-10, -8.0589067083759 95e-16, 5.545928882693489e-16, 1.5863156528500 43e-05, -7.1768879261459 91e-07, -8.0077168944569 95e-09, 3.9404395744106195e-10]
[0.0, -3.0648147740566 53e-12, -1.941297497284347e-13, 8.354686379359565e-11, -3.002497678107353e-10, -8.237022177299102e-16, 7.732564432137759e-16, 2.2810377580591562e-05, -9.171985857701971e-07, -1.5208318640831645e-08, 4.881088171879905e-10]
[0.0, -3.8250913298286 54e-12, -2.1557256094407779e-13, 1.0582226833253055e-10, -3.1519909645834057e-10, -7.9691036821711 72e-16, 7.392069258162 42e-16, 3.0078304382613 59e-05, -8.0915263509174 59e-07, -2.1687975594122 148e-08, 5.858109467726537e-10]
[0.0, 4.898526877668036e-14, 3.226162199524798e-14, 1.5221538682805016e-10, -2.2969093260059183e-10, -3.8928949810823143e-16, 1.2396772961568568e-15, 1.669936034981911e-05, -9.120282074640079e-07, -1.2552474179891876e-08, 7.864321775778483e-10]
1.24665397e+14
0.008
[0.0, 4.898526877668036e-14, 3.226162199524798e-14, 1.5221538682805016e-10, -2.2969093260059183e-10, -3.8928949810823143e-16, 1.2396772961568568e-15, 1.669936034981911e-05, -9.120282074640079e-07, -1.2552474179891876e-08, 7.864321775778483e-10]
[0.0, -3.0941583041167325e-12, -1.4494021147333922e-13, 4.893193617197457e-11, -2.9168674182339254e-10, -8.267127773374596e-16, 8.146694667021326e-16, 1.7168119066762928e-05, -9.600270410291194e-07, -5.1282383523898 56e-09, 3.945414326341477e-10]
No mutation
[0.0, -3.0941583041167325e-12, -1.4494021147333922e-13, 4.893193617197457e-11, -2.9168674182339254e-10, -8.267127773374596e-16, 8.146694667021326e-16, 1.7168119066762928e-05, -9.600270410291194e-07, -5.1282383523898 56e-09, 3.945414326341477e-10]
[0.0, -1.0980446032972 35e-12, -4.5026552552221794e-14, 1.6403660456030293e-11, -4.1975289220068 43e-10, -2.1711825570076193e-15, 6.091243632960105e-16, 2.2430202830456702e-05, -5.2429714653230 17e-06, -5.6626811225147 43e-09, 2.590862669986625e-09]
[0.0, -2.6895579968994792e-12, -1.5954318164408026e-13, 9.901956630789 18e-11, -2.5342829924412313e-10, -8.0155499378037 67e-16, 4.901838014610402e-16, 1.753476078784492e-05, -6.7373673268070 62e-07, -1.0256996492903925e-08, 3.703128225392193e-10]
[0.0, -1.4093291991083223e-12, -1.1031845542227329e-13, 1.3012452573233519e-10, -3.6047581134360635e-10, -8.60157190512e-16, 6.11076698494599e-16, 2.4014276812951 09e-05, -7.9342860678401 04e-07, -1.0849200874057677e-08, 4.967313311586191e-10]
[0.0, -1.5150473111618814e-12, -1.8057805668137127e-13, 2.2215232807251892e-11, -1.2493223064735554e-10, -3.4357624021413 06e-16, 7.612791142165564e-16, 2.0873339540467684e-05, -3.4497740924974 98e-06, -8.9450692067354 48e-09, 1.7622968359814592e-09]
[0.0, -3.0979447687972605e-12, -2.4446909400703067e-13, 1.7666051520016975e-10, -2.3153542130797 97e-10, -6.8733217732480 94e-16, 9.824929326112 08e-16, 2.2671510192856516e-05, -9.1691266215008 85e-07, -1.3150776652777117e-08, 4.391250048308186e-10]
No mutation
[0.0, -3.0979447687972605e-12, -2.4446909400703067e-13, 1.7666051520016975e-10, -2.3153542130797 97e-10, -6.8733217732480 94e-16, 9.824929326112 08e-16, 2.2671510192856516e-05, -9.1691266215008 85e-07, -1.3150776652777117e-08, 4.391250048308186e-10]
[0.0, -3.2363105413467977e-12, -4.714833957536693e-14, 1.714917862583881e-11, -2.6487320990611434e-10, -1.3301913667601918e-15, 3.6062137933017256e-16, 1.4476782658571245e-05, -6.4301180577250 2e-06, -1.0260905964050802e-08, 1.688824505380186e-09]
[0.0, -3.0648147740566 53e-12, -1.941297497284347e-13, 8.354686379359565e-11, -3.002497678107353e-10, -8.237022177299102e-16, 7.732564432137759e-16, 2.2810377580591562e-05, -9.171985857701971e-07, -1.5208318640831645e-08, 4.881088171879905e-10]
[0.0, -1.5892266845206808e-12, -2.201539410324046e-13, 5.7883458039849404e-11, -3.874889129193645e-10, -1.099463519506 09e-15, 5.488630925087693e-16, 1.5771613241491735e-05, -7.5427391819561 9e-07, -1.4730891142939849e-08, 6.259419287444952e-10]
[0.0, -2.2518068404789 93e-12, -1.2867141628568 76e-15, -2.6024822124625 07e-12, -9.8672193850148 34e-11, -9.7295622077133 64e-16, 6.374635043507242e-16, 2.8002397484604415e-05, -5.8023817271533 23e-06, -9.9806404252262 85e-09, 5.635323747257173e-10]
[0.0, -2.528646475302289e-12, -1.8138627339053722e-13, 9.893427264160 67e-11, -2.5133254605589 89e-10, -8.1629535608714 95e-16, 5.235250051067119e-16, 1.5229200331181222e-05, -9.8403861308747e-07, -5.7651938039725 48e-09, 3.9467755261679406e-10]
[0.0, -3.7387060887013 98e-12, -1.4267131658462962e-13, 1.182935612167499e-10, -3.4713350382856 54e-10, -8.0642391826240 03e-16, 4.180707970900584e-16, 1.2050343406818081e-05, -1.3659509662320277e-06, -7.6669854882929 84e-09, 4.290423239252747e-10]