Genetic Algorithms

MDL Project

Final Deadline : 12th March, 11:55pm

1 Genetic Algorithm

Genetic algorithm is often used in parameter selection and obtaining optimal solutions. You have been given coefficients of features(a vector of size 11) corresponding to an overfit model and your task is to apply genetic algorithm in order to reduce the overfitting i.e. generalize the model so that the model performs better on unseen data. To help you with the formulation of your fitness function, you will be allowed to query and check how your coefficients perform on the training and the validation set.

The output(of your vector) will be calculated using:

$$f(x) = \sum_{i=0}^{10} w_i * x_i$$

where x_i represents the features, and w_i represents the coefficients submitted by you.

MSE, which will be returned for the validation and the train data, will be calculated using:

$$MSE = \sum_{x \in data} ((y - f(x))^2)/N$$

For the sake of simplicity, the weights you submit should satisfy:

$$w_i \leq 10$$
$$w_i \geq -10$$

2 Server Details

You will be provided with a file client.py. It has one function named *get_errors* that you can use to query the server. Import this client into the GA code you write and call the function by supplying your team key and weight vector. The team key will be sent directly through an email. Please note that there is a daily limit of 200 requests to the server which means total functions calls allowed are 200. This limit might change in the due course of project. To check your daily usage, visit `http://10.4.21.156/getusage`

2.1 Available Functions

- *get_errors*
  params:
  - ID => String: The secret team ID
  - Vector => Python List: The 11D weight vector

  returns: 2 element python array of the form [train error, validation error]

2.2 Workflow

A typical workflow will be as follows:

- Start by reading the overfit array from the text file provided with the assignment itself.
- Use the *get_errors* function to guide the learning in Genetic Algorithm.
- Submit the final 10 weight vectors with your submission(in descending order of your preference as described below).

2.3 Essential Notes

- Please be mindful of the daily usage limit.
- Please do not share your secret ID with anyone. If someone exhausts your daily quota, there is nothing we can do.
- The secret ID will be sent through an email.

3 Format of Report

- Summary of your Genetic Algorithm with all the steps, also mention if you have made any major changes to the base genetic algorithm.
- Three diagrams that showcase three iterations of your algorithm. Refer Fig. 1 that illustrates one iteration of a genetic algorithm. You can choose a population of size minimum 7 to draw the diagram and then proceed with the iterations. The initial population(corresponding to the 1st diagram) need not be your very first population you used and can be intermediate, but the diagrams should be for 3 successive/consecutive iterations. The vectors cannot be random and should be the ones generated by your algorithm.
- Explain your fitness function.
- Explain your crossover function.
- How exactly did you apply the mutations(if any).
- What were your hyperparameters like pool size, splitting point for creating new genes, etc and why did you choose those parameters?
- Statistical information like number of iterations to converge, etc.
- What heuristics did you apply, mention the ones that didn't work too.
- Mention the train error and validation error that you were able to achieve. Explain why this performs well on an unseen data with theoretically sound points. Avoid vague arguments.
- Anything else that you used, tricks, brute force, etc. that we should be aware of.

4 Deliverables

Submit a zip file named <**TeamNumber**>**.zip** which on unzipping returns a folder named <**TeamNumber**> that contains the following files,

- A report named **Report.pdf** that follows the format as mentioned in Section 3.

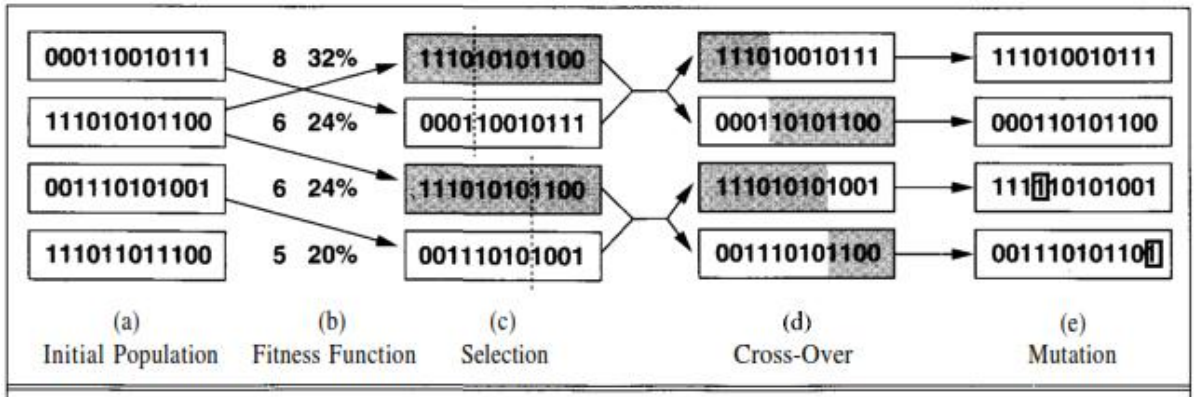

Figure 20.16 The genetic algorithm. In (a), we have an initial population of 4 individuals. They are scored by the fitness function in (b); the top individual scores an 8 and the bottom scores a 5. It works out that the top individual has a 32% chance of being chosen on each selection. In (c), selection has given us two pairs of mates, and the cross-over points (dotted lines) have been chosen. Notice that one individual mates twice; one not at all. In (d), we see the new offspring, generated by cross-over of their parents' genes. Finally, in (e), mutation has changed the two bits surrounded by boxes. This gives us the population for the next generation.

Figure 1: Sample Iteration of Genetic Algorithm

- A python file named **main.py** that contains the source code of your Genetic Algorithm. The code, on executing should generate a file named **output.txt** which contains **ten** vectors of size 11. The MSE will be computed using these ten vectors on an unseen test dataset which decides your leaderboard position. The lesser the average MSE is, the higher your position is in the leaderboard.
- A text file named **output.txt** which contains **ten** 11-D vectors in the following format,
  - The vectors should be arranged in the order you want us to consider the vectors. This means the first vector is the first best vector and the tenth vector is the least best vector according to you.
  - Each vector should be a list of 11 floats
  - Append all the vectors into a single list of lists called answer
  - Each element in answer is a list of floats
  - The answer list would look like,

```
[
    [w00, w01, ......],
    [w10, w11, ......],
    .
    .
    .
    [w80, w81, ......],
    [w90, w91, ......]
]
```

  - Use json.dump to put the answer array into a file called output.txt. Code should be like,

```
with open('output.txt', 'w') as write_file:
    json.dump(answer, write_file)
```

  - Make sure you are able to use json.load to construct back the original array from output file. Since we have automated evaluations, failure to be able to reconstruct the original array from output.txt file would lead to **zero** points in the leaderboard.

- A folder named **generations** which contains ten files namely generations_1.txt, generations_2.txt, generations_3.txt, generations_4.txt, generations_5.txt, generations_6.txt, generations_7.txt, generations_8.txt, generations_9.txt, generations_10.txt. Each of these files should contain the intermediate population after selection(if any), crossover(if any) and mutation(if any) for every generation for that particular vector. For ex: generations_1.txt should contain all the intermediate population(please label them) for every generation corresponding to the first 11-D vector.

The directory structure on unzipping <**TeamNumber**>**.zip** should look like,

- <TeamNumber>
  - Report.pdf
  - main.py
  - output.txt
  - generations
    - generations_1.txt
    - generations_2.txt
    - generations_3.txt
    - generations_4.txt
    - generations_5.txt
    - generations_6.txt
    - generations_7.txt
    - generations_8.txt
    - generations_9.txt
    - generations_10.txt

5 Evaluation Criteria

- Marks Distribution
  1. Intermediate Evaluation (March 5th): 10%
  2. Basic Code, Report and Viva: 40%
  3. Relative preformance on the test set(on the basis of rank among all the teams): 50%
  4. Your weights will be tested on a completely unseen dataset, lower the MSE, higher your position on the leaderboard.
- All coding questions have to be done in Python3 only.
- You are expected to write vectorized code.
- Plagiarism will be penalized heavily.
- No deadline extensions.
- Manual evaluations will be held regarding which further details will be announced later.