Artificial Intelligence

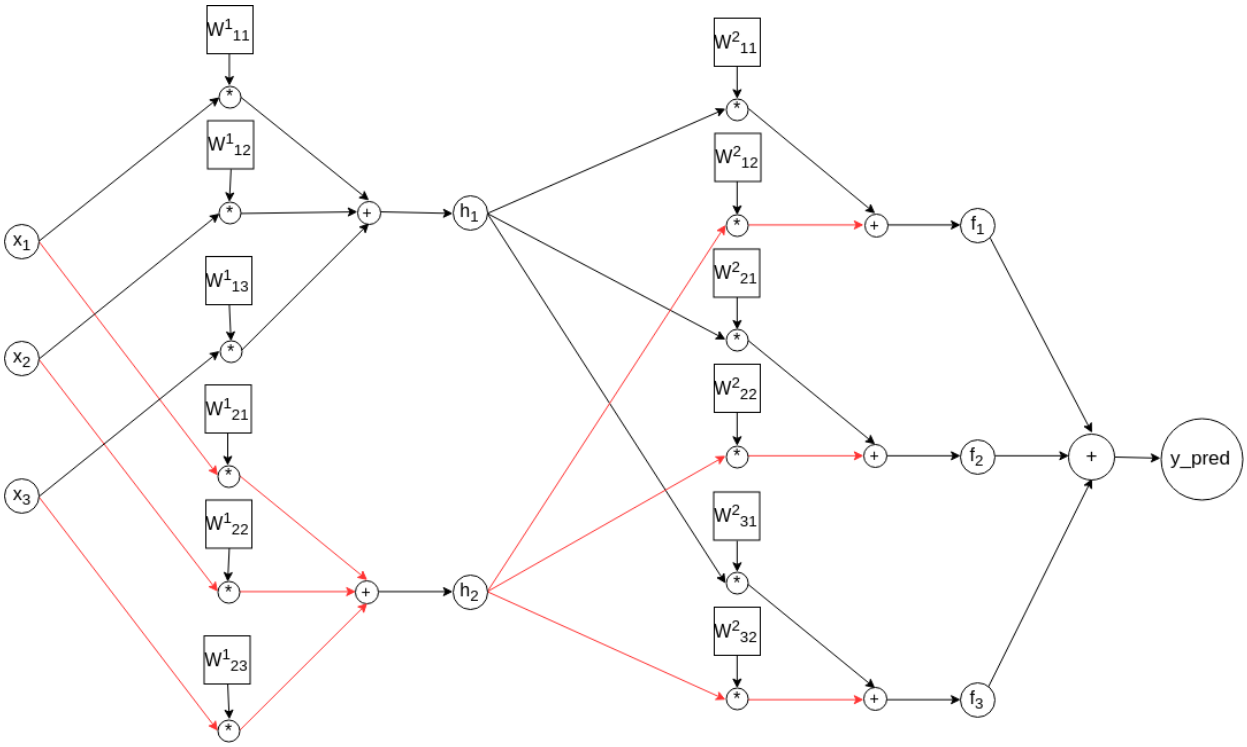
ASSIGNMENT 2 - EVOLUTIONARY PROCESS OPTIMIZER
Spring 2018 - Ozyegin University

Deadline for Code and Report Submission: 28.March.2018, Wednesday 23:55

by Okan Tunalı
okan.tunali@ozu.edu.tr

In this assignment, you will implement Genetic Algorithm to optimize following process.

The process involves performing some series of mathematical operations on the given diagram. An array of real numbers $\mathbf{x} = [x_1, x_2, x_3]$ is given to some parameters called **weights** denoted as W; and after performing that operations illustrated in the figure, system will produce a real valued output, called $\mathbf{y}_{\mathbf{predict}}$. Note that the color of the arrows has no functionality, it is just to make the process visually tracable.

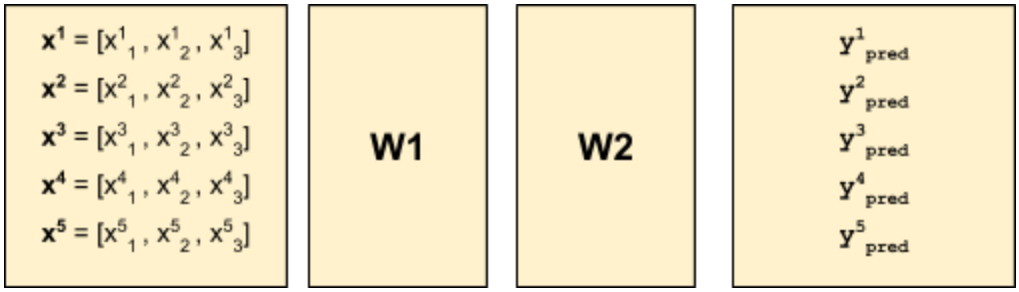


Sample calculations are as follows:

$$h_1 = (W^1_{11} * x_1) + (W^1_{12} + x_2) + (W^1_{13} + x_3)$$
$$f_1 = (W^2_{11} * h_1) + (W^2_{12} + h_2)$$
$$y_{pred} = f_1 + f_2 + f_3$$

If you notice, process can easily be converted into a linear algebra representation - matrix multiplications (*highly recommended to reduce complexity*).

In the following example, we have an input having N training instances where N equals 5. So the input **X** is a [5 x 3] matrix having 5 training instances and the calculated output **y** is a [5 x 1] vector. **y** is calculated by performing aforementioned multiplications and summations with corresponding weights.



Genetic Algorithm as optimizer:

In this assignment, you are going to find the most convenient weight values (12 weights) by applying genetic algorithm. The value of each weight is represented by a gene. A chromosome will consist of 12 genes - corresponds the value of weights. Your algorithm will start with a population - a set of chromosomes - as candidate solutions for your problem.

You should implement the five phases of a genetic algorithm:

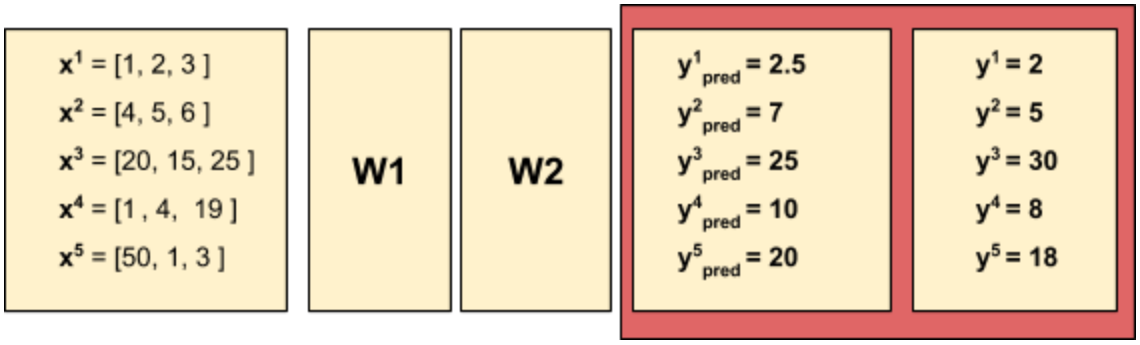
1. Initial Population
2. Fitness Function
3. Selection
4. Crossover
5. Mutation

By intuituon, “survival of the fittest” principle must be embedded, however in this study, **the gene *Population size* stays the same**. When the weakest individual die, it is replaced. For that reason, it is normal to have similar individuals in your population as they demonstrate higher fitness.

- Hierarchy: *Genes → Individuals (Choromosomes) → Population*

The system has only linear components; so output must also be a linear combination of inputs. **You will calculate your real output / target as the *mean of rows***. See the example:

- $y^1 = mean(\mathbf{x^1}) = (1+2+3)/3 = 2$ *where* x_{11}*: 1* x_{12}*: 2 and* x_{13}*: 3*



Given the inputs and an individual of population (gene string of weights, which is not shown in the figure), system produced close but inaccurate results. *It seems like it is on the good track of fitness.*

Fitness measure for evaluation

Our fitness measure for the individuals of the population will be sum of root squared error, which is $\sum_{i}^{\#Row} \sqrt{(y^i - y^i{}_{pred})^2}$.

Implementation notes:

- The input matrix should be [N x 3] where $N \geq 10$, and population matrix should have size of [M x 12] where $M \geq 20$
    - *Benchmark: A 4 year old laptop with Intel® Core™ i5-4210U can easily handle 100 rows of input with a population having 200 individuals.*

- *Input matrix* ***X*** should have real values in the range of [-5,5].

- Your weights / genes should have values in the range of [-1,1]. Their initial values and mutated genes will be generated from this range.

- Algorithm should randomize floating numbers (***not integers***). Random generator's distribution is up to you: it can be gaussian or uniform distribution. Mean of the generation ranges may not be zero; given that range, you are free to play.

- Stopping condition for the evolution may be a iteration limit like 2000 or a condition such as "until fittest individual provides fitness error at most 0.05". But in the end your algorithm has to make progress as you will plot its progress of fitness over iterations for the genes (see example plot).

- There should be a line to print which generation is being created (number) and each of the population members' fitness levels (see example). Not necessarily after each iteration, it may be after each T number of iterations.

- You may have as much as helper methods or classes you like. Clean coding is appreciated.

- *It is highly recommended to improvise and research for the related methods to reach faster convergence ability and escaping from sub-optimal solutions to be used in Mutation and CrossOver.*

- *You are not allowed to use linear algebra solvers or implement any kind of close form solutions like* $\hat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$.

- Please add comments to your methods and pick explanatory names for the variables.

- You may be inspired from the source codes you found but it is strictly forbidden to plagiarize.

Notes for the Assignment Report:

- You do not have page limit.
- Figures are great to explain; but their legend, label or title is insufficient to show your work. They are helpers and they require some text.
- Please to not copy/paste your code snippets. On the other hand, their formal/non-formal pseudocode with explanation would be perfect.
- ***There has to be a graph showing your individuals' decreasing fitness error with comments on it.***
- "It was not asked, so I did not write" is the approach of freshmen. Please feel free to add and improvise any related content to show your work.
- If you do not write your code from scratch, please include your references at the end of the report.

Example:

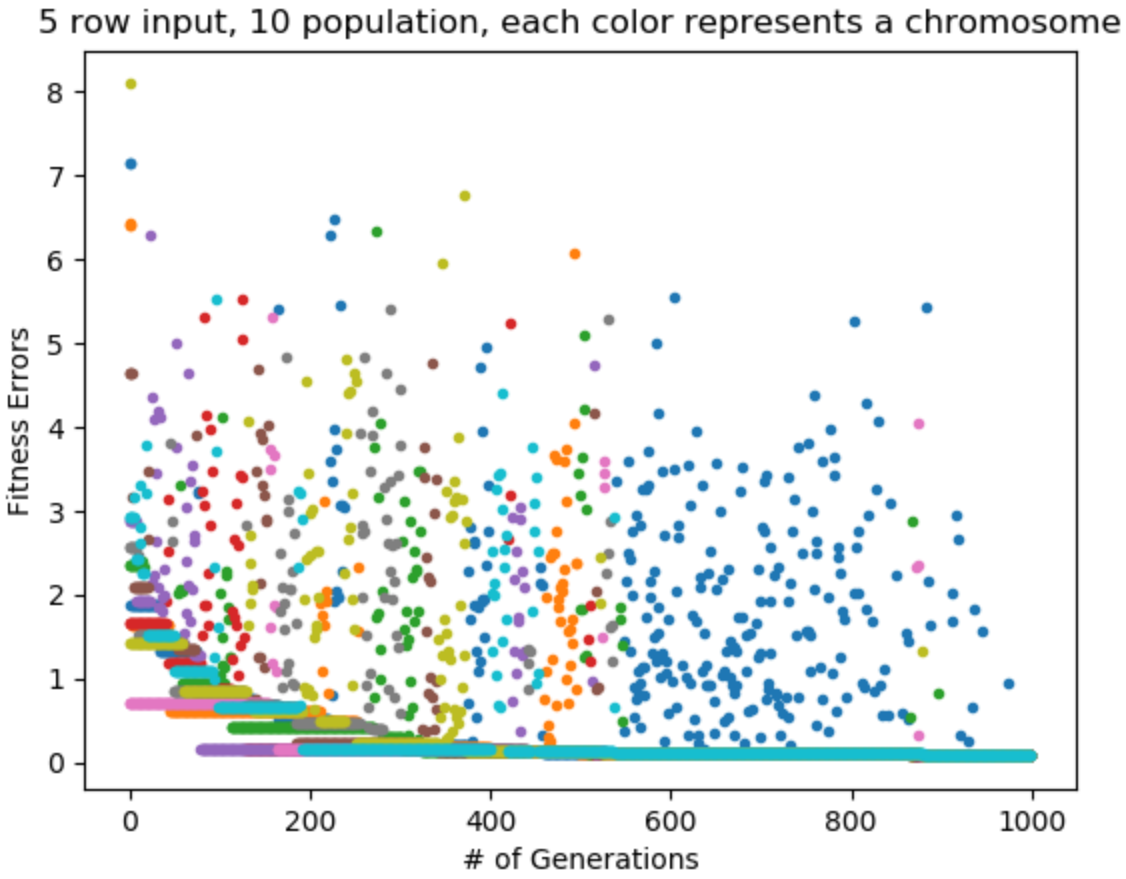
The setting is as follows:

- Input matrix **X** is [5x3] so N = 5 and the matrix of weights, the weights population is [10x12] because M = 10. Note that N <10 and M < 20 just for this example to prevent it to occupy too much space.
- Limit of iterations is set to 1000 and T = 200. So in each 200 generations, it prints out the errors of chromosomes.
- In the weights matrix, I rounded them up to 2 decimal points.

```
Input Matrix:
 [-0.72607689 -1.04935171  1.62234085]
 [-1.42469534 -0.12198307 -1.22963098]
 [ 0.87323313 -0.01011584 -0.57608823]
 [-0.97611884  0.51950684  1.11433109]
 [-1.50486662  1.73845095 -0.42632182]


Generation:  1
Fitness errors of chromosomes
[1.181, 2.562, 2.89, 1.749, 3.143, 2.419, 3.558, 2.032, 3.435, 1.692]


Generation:  200
Fitness errors of chromosomes
[0.32, 1.52, 0.47, 0.24, 0.46, 0.53, 0.46, 0.51, 0.5, 0.11]


Generation:  400
Fitness errors of chromosomes
[0.11, 0.11, 0.11, 0.11, 1.21, 0.11, 0.11, 0.11, 0.11, 0.11]


Generation:  600
Fitness errors of chromosomes
[0.92, 0.07, 0.07, 0.07, 0.07, 0.07, 0.07, 0.07, 0.07, 0.07]


Generation:  800
Fitness errors of chromosomes
[1.72, 0.03, 0.03, 0.03, 0.03, 0.03, 0.03, 0.03, 0.03, 0.03]


Generation:  1000
Fitness errors of chromosomes
[0.03, 0.03, 0.03, 0.03, 0.03, 0.03, 0.03, 0.03, 0.03, 0.03]

Weights/Chromosomes Matrix | fitness errors
[ 0.92  0.47  0.99  0.37  0.83  0.29  0.6   0.38 -0.79  0.13  0.45 -0.28] 0.03
[ 0.92  0.47  0.99  0.37  0.83  0.29  0.6   0.38 -0.79  0.13  0.45 -0.28] 0.03
[ 0.92  0.47  0.99  0.37  0.83  0.29  0.6   0.38 -0.79  0.13  0.45 -0.28] 0.03
[ 0.92  0.47  0.99  0.37  0.83  0.29  0.6   0.38 -0.79  0.13  0.45 -0.28] 0.03
[ 0.92  0.47  0.99  0.37  0.83  0.29  0.6   0.38 -0.79  0.13  0.45 -0.28] 0.03
[ 0.92  0.47  0.99  0.37  0.83  0.29  0.6   0.38 -0.79  0.13  0.45 -0.28] 0.03
[ 0.92  0.47  0.99  0.37  0.83  0.29  0.6   0.38 -0.79  0.13  0.45 -0.28] 0.03
[ 0.92  0.47  0.99  0.37  0.83  0.29  0.6   0.38 -0.79  0.13  0.45 -0.28] 0.03
[ 0.92  0.47  0.99  0.37  0.83  0.29  0.6   0.38 -0.79  0.13  0.45 -0.28] 0.03
[ 0.92  0.47  0.99  0.37  0.83  0.29  0.6   0.38 -0.79  0.13  0.45 -0.28] 0.03
```

```
[ y  | y_pred of fittest]
 [-0.051 -0.038]
 [-0.925 -0.899]
 [ 0.096  0.093]
 [ 0.219  0.207]
 [-0.064 -0.082]
```

You need to have the following decrease in errors as a trend. For sure, crossover and mutation causes jumps in the error but that is a conscience part of the algorithm.



Another example for 100 rows of input and a population of 50 chromosomes: