Machine Learning

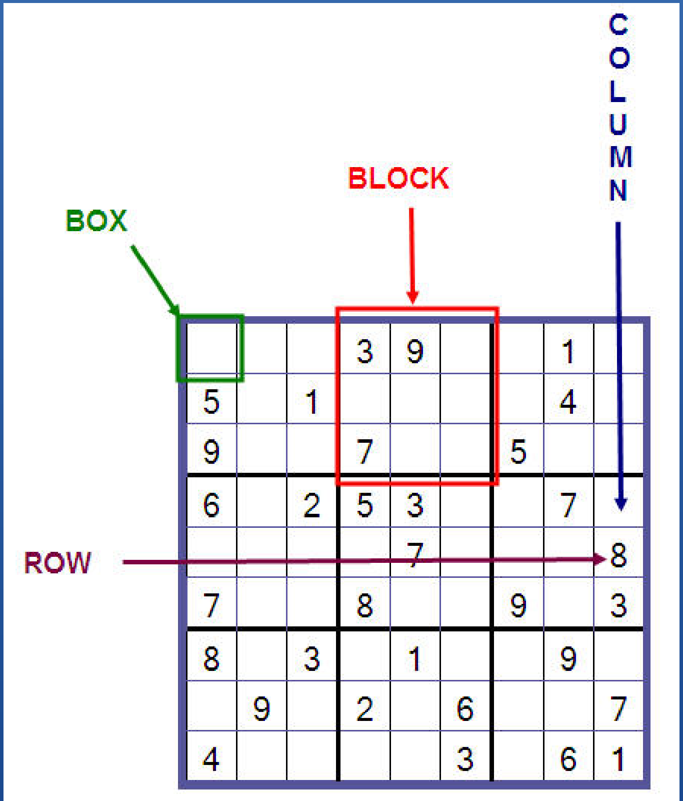
Using Genetic Algorithms
to
solve Sudoku Puzzle

By Achyut and Sizen

Introduction

- Mathematical puzzle invented by Howard Garnes in 1979.
- It is a 9x9 grid
- Total of 81 positions that are divided into 3x3 sub-grids each.
- Presented with numbers at random locations and some empty positions.
- We have to fill in these empty positions with numbers that follow certain rules.

Sudoku



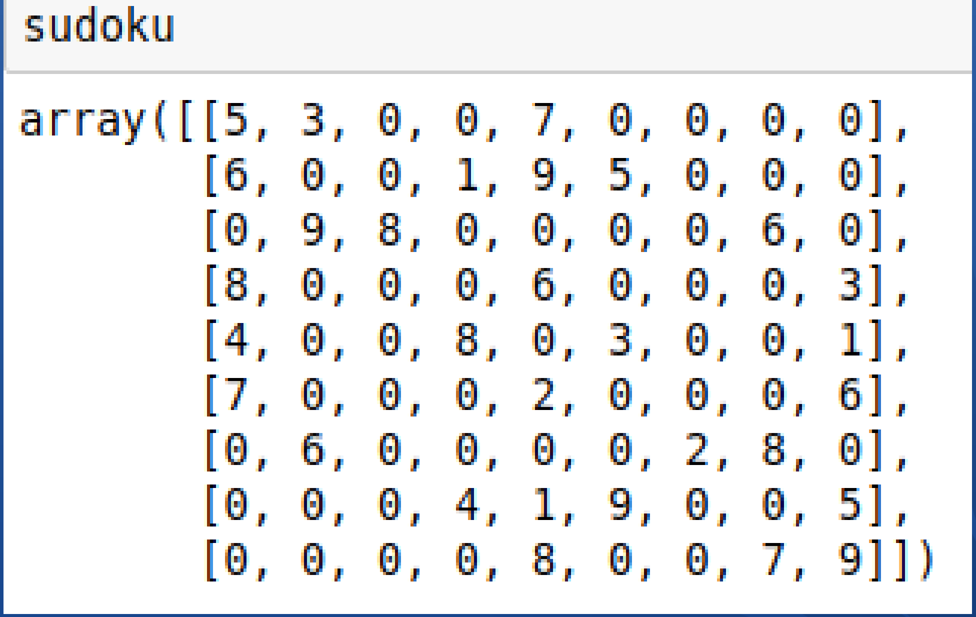
			3	9			1	
5		1					4	
9			7			5		
6		2	5	3			7	
				7				8
7			8			9		3
8		3		1			9	
	9		2		6			7
4					3		6	1

Rules

- Each row must have numbers from 1 to 9 , no repetitions.
- Each column must have numbers from 1 to 9 , no repetitions.
- Each sub-grid must have numbers from 1 to 9 , no repetitions.

Numpy version of Sudoku

- Multi-dimensional numpy array to represent sudoku.



```
sudoku
```

```
array([[5, 3, 0, 0, 7, 0, 0, 0, 0],
       [6, 0, 0, 1, 9, 5, 0, 0, 0],
       [0, 9, 8, 0, 0, 0, 0, 6, 0],
       [8, 0, 0, 0, 6, 0, 0, 0, 3],
       [4, 0, 0, 8, 0, 3, 0, 0, 1],
       [7, 0, 0, 0, 2, 0, 0, 0, 6],
       [0, 6, 0, 0, 0, 0, 2, 8, 0],
       [0, 0, 0, 4, 1, 9, 0, 0, 5],
       [0, 0, 0, 0, 8, 0, 0, 7, 9]])
```

Population

- Size of a genome = Number of zeros in the sudoku.
- Initialized 100 genomes each of size 51.
- Tried running the GA with variety of population sizes and got different results.

Fitness function

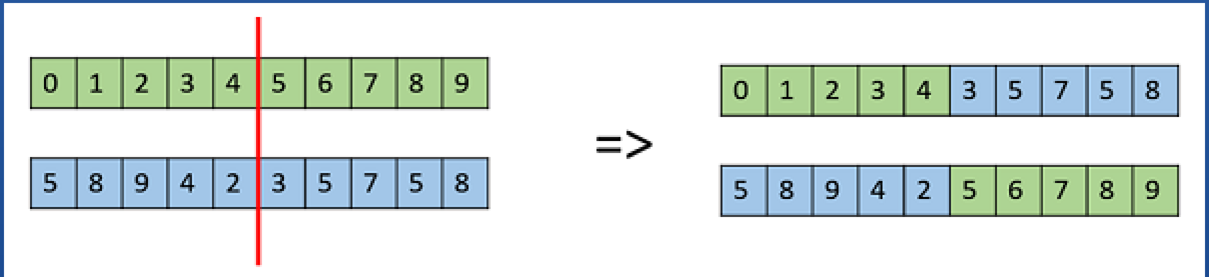
- Performance of GA is dependent on the fitness function.
- Based on the rules of the Sudoku.
- Each unique integer in row, column and a sub-grid gets a score of +1.
- Perfect fitness - 243
- R1 = [1,2,3,4,5,6,7,8,9] ; Score = 9
- C2 = [1,2,3,5,5,7,7,8,9] ; Score = 7

Selection method

- Initially started with Roulette Selection
- Problem with roulette is that it tries to converge too fast which is not desirable for this type of problem.
- Sudoku is a hard GA problem that has many local minima
- Next thing was to implement a Tournament Selection.
- Selected only half of my fittest parent genes.
- Replaced the bottom half with the children.

Cross-over

- Performed a one-point crossover
- Randomly selecting one point within specified ranges and swapping the genes with one another.
- Also tried two-point and uniform crossover but one point worked fine. The other two dragged the score even down.



0 1 2 3 4 5 6 7 8 9

5 8 9 4 2 3 5 7 5 8

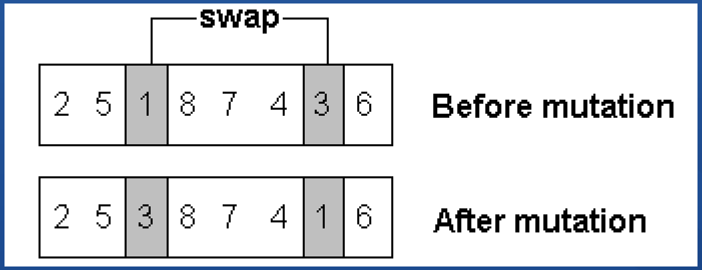
=>

0 1 2 3 4 3 5 7 5 8

5 8 9 4 2 5 6 7 8 9

Mutation

- Tried different mutation techniques.
  - One point swap mutation
  - Two point swap mutation
  - Inversion mutation
  - Five point mutation (worst)



2 5 1 8 7 4 3 6 Before mutation

2 5 3 8 7 4 1 6 After mutation

Results and Conclusion

- After performing the experiment:
  - With Roulette selection
    - The best population size – 5000
    - Iterations – 20
    - Cross-over – one point
    - Mutation - 1 point random swap
  - With Tournament Selection
    - Parameters remained the same but with better fitness scores.

If we had time...

- Implement a restart heuristic
  - Whenever GA gets stuck at the local minima, store the top few solutions and and re-run the GA automatically.
- Redefine the fitness function.
- Think if GA is really needed to solve your problem. Sudokus can be solved with other techniques that take less time to implement.
- Backtracking, constraint propagation, stochastic search etc.