**University of Waterloo**

**Department of Electrical and Computer Engineering**

**ECE 457A: Cooperative and Adaptive Algorithms**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Project Title** | Feature Selection using Genetic Algorithm | | |  |
| **Group Number** | 13 | | |
| **Team Member** | *Tian Zhu* | *20517833* | *t27zhu@edu.uwaterloo.ca* |
| *CongHao He* | *20589010* | *chhe@edu.uwaterloo.ca* | *10*  *points* |
| *Yiran Tian* | *20580120* | *y62tian@edu.uwaterloo.ca* |
| *YuCheng Liu* | *20558944* | *y698liu@edu.uwaterloo.ca* |

**[2 points] Algorithm and Problem:**

Our topic is feature selection, and the goal is to find a set of most relevant features that provide best fit for the predictive model. In this case, each feature is one gene in the gene pool, and the combinations of the genes (solutions) are actually the individuals in population. Firstly, the population is generated as the form of binary array. 0 or 1 in the array indicates the existence of each feature. Then, fitness function is thrown to evaluate each model, and only part of the population is kept for further recombination according to the result of evaluation. After the population is halved, crossover happens between individuals to generate new population. Parents are paired due to fitness ranking. Then, mutation happens to prevent local maximum. According to our specific problem, only 10 features can be run at same time due to computation limit, thus the population decreasing is skipped in code.

Moreover, GA is also suitable in this case since feature selection do not require permutation data like Sudoku, making crossover much easier to achieve. In order to diversify the solution, mutation happens randomly. Then the current population is passed to fitness assignment again. If termination criteria (maximumly 100 generations) is not hitten, go back to selection process. to sum up, the Genetic Algorithm is suitable for this Feature selection problem. Since the genes could represent the features, and binary indication of existence is well adapted to each feature. Each individual in the population is exactly solution of the problem, therefore, the formulation of problem is a good match with GA algorithm. GA algorithm tends to evaluate multiple solutions at the same time and has a diversifying process (mutation), thus rule out the possibility of stucking in local best solutions.

**[2 points] Algorithm Scalability:**

Discuss what will happen if you increase your search space.

The search space increases as the number of features increase, yet the run time of the algorithm is unlikely to change. Firstly, if we do one point or n-point mutation, the time for mutation remains similar. Also, the terminating condition is set not to reach a certain level of change in fitness score, but rather number of generations. It would be possible to increase the number of generations as the number of features increase, but in general it is not favorable because that defeats the purpose of approximate solution. It is also possible to increase the population pool as the number of features increase. In this case, the pool size will increase linearly with the number of features. The space complexity is more influenced by the pool size, as the space required is to hold the population pool. Increase on chromosome length (addition of features in our case) is less computationally demanding compared to an increase in population. The time complexity is associated with both pool size and the generation limit. However, both variables can increase logarithmically or near constant at higher numbers.

In general, increasing the terminating generation would only possibly allow the algorithm to search longer in the search space, but not necessarily finding a more optimal solution. This property is very important because for the feature selection problem, the search space increases exponentially as the number of features decrease. To have a relatively constant change in running time and space would allow the solution to be much more applicable to more areas..

**[4 points] Results:**

Share the results achieved by the algorithm w.r.t your problem and dataset. Comments on the solution(s) optimality.

Final solution of GA:

Features Selected: ['texture\_mean', 'radius\_worst', 'texture\_worst', 'area\_worst', 'compactness\_worst']

[('Current', ['texture\_mean', 'radius\_worst', 'texture\_worst', 'area\_worst', 'compactness\_worst'], 0.972027972027972), ('Best', ['texture\_mean', 'radius\_worst', 'texture\_worst', 'area\_worst', 'compactness\_worst'], 0.972027972027972)]

Our problem is to select a set of features that achieve best accuracy of outcomes from all the features provided. For the Genetic Algorithm, the Wisconsin Breast Cancer Dataset is selected to perform the test. This dataset has 30 features, and 2^30 possible solutions are available.

The optimality of the solution depends on multiple parameters: the population size, the crossover rate, the mutation rate and the number of generations to terminate on. When the population size increases, the chance of achieving a global optimum increases due to the increase in search space with the downside of a longer running time. The crossover rate determines the exploration speed. The greater the crossover rate, the faster the exploration (i.e. less probability of premature convergence on a local extremum). The mutation probability determines the exploitation power. It needs to be kept within the range of 1/population\_size and 1/chromosome\_length. This parameter is tuned to the point where the probability of a successful mutation is ⅕. This ensures that the mutation is neither too random to the point where it is a random walk nor too slow to the point where unnecessary processing power is wasted. The number of generations needs to be set a optimum number such that the algorithm does not terminate too early nor waste computing power.

For the GA, another factor that affects the optimality of the solution is the initial condition. Different initial feature set would provide different results. In the desired world of infinite population pool, this will not be a problem. In the real world however, it would be impossible to have a large pool size due to computation restraints. However, it is possible to achieve optimal solution simply by running the GA multiple times. In our GA solution, we used the rank based approach in selecting the parent pool. This is because the fitness scores, i.e. the accuracy of the solution, are very close to each other. Therefore, it is better to attribute a heavier importance to the fitness scores to ensure the distinction between the different solutions.

The final solution that GA provides can be achieved using a space of O(population\_size) and the running time is around O(population\_size \* generation\_number). These two variables are only affected by the solution space in lower dimensions (such as 1 or 2) and will remain relatively constant for higher dimensions. Therefore, the GA is great for feature selection in that even if the number of available features is high, the time and space utilize will be approaching a constant. The result solution can be optimum if adequate number of runs are tested on different initial solution. For our result example, the optimal solution was found after several runs of GA on random initial solutions.

**[1 points] Performance Comparison [if applicable]:**

Comment on the algorithm performance and the optimality of the solution achieved compared to the results you previously achieved.

Regression accuracy achieved is similar to the other method used previously, 97.2%. Although the accuracy is similar, the features subset found that achieves it is different to the one found by BFS or Tabu for example. This shows the explorative nature of GA algorithm, which is capable of finding different extrema maximizing the score %.

GA uses coding of parameters to form solutions, and it does not modify parameters (specifically features here) themselves. Comparing to BFS, GA can be set up with a random start. Comparing to Tabu and SA, GA does not need information other than objective function and fitness function.

Final solution of Tabu:

(['texture\_mean', 'perimeter\_mean', 'smoothness\_mean', 'compactness\_se', 'texture\_worst', 'perimeter\_worst', 'smoothness\_worst'], 0.972027972027972)

Final solution of BFS:

['area\_worst', 'compactness\_worst', 'concavity\_se', 'perimeter\_worst', 'radius\_worst', 'texture\_worst'] 97.2%

Final solution of SA:

('Best', ['radius\_mean', 'perimeter\_mean', 'area\_mean', 'perimeter\_se', 'area\_se', 'smoothness\_se', 'concavity\_se', 'concave points\_se', 'symmetry\_se', 'fractal\_dimension\_se', 'texture\_worst', 'compactness\_worst', 'concavity\_worst', 'concave points\_worst', 'symmetry\_worst'], 0.9790209790209791)]

Final solution of GA:

Features Selected: ['texture\_mean', 'radius\_worst', 'texture\_worst', 'area\_worst', 'compactness\_worst']

[('Current', ['texture\_mean', 'radius\_worst', 'texture\_worst', 'area\_worst', 'compactness\_worst'], 0.972027972027972), ('Best', ['texture\_mean', 'radius\_worst', 'texture\_worst', 'area\_worst', 'compactness\_worst'], 0.972027972027972)]