**University of Waterloo**

**Department of Electrical and Computer Engineering**

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

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

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

Discussing the suitability of search algorithm as it relates to the problem at hand as well as the algorithm complexity

Tabu search is a proper algorithm for our feature selection problem. With a dataset contains n features, 2^n possible solutions could be obtained, which is a huge collection of states. By greedy algorithm, local best solutions could be found, but it is hard approach the absolute best solution. However, Tabu search prevents repeated search in a same area. It is also able to explore broader range of features combination and compare all the local best solutions due to the use of memory (e.g. remembering the previous best results).

According to the feature selection problem, a Tabu list of a array of length *n* is needed to hold the status of each feature. For this specific solution, it’s an on/off move instead of swap; same as the long term memory. From our feature selection problem, the move will be either removing a feature or adding in a feature to our current feature list. Our algorithm starts from a solution of random feature set, for each move it chooses the possible moves with best anticipated outcome. We defined our anticipated outcome by using the logistic regression as a cost function, and the difference in the model’s accuracy will be the score we use to find the best anticipated outcome. A Tabu length of will be recorded corresponds to moves feature in Tabu list to prevent greedy search. The Tabu length will deduct by 1 in each iteration, the feature is resumed to respond when Tabu length hits 0 again. In order to rule out the coincidence of a best solution which is already Tabu, the Tabu is released when anticipated outcome is better than all of the ones in long term memory.

**[2 points] Algorithm Scalability:**

When the search space is increased, the Tabu list size will increase by square. For our specific problem of feature selection, the tabu list is a array of length of n, which is the short-term memory. From the long-term memory, we decided to use the default aspiration criteria, which means that if a tabu move becomes admissible if it yields a solution that is better than any obtained solution so far. Since the memory is in an 1\*n matrix. As more features added to the problem, the algorithm cost longer to find the final solution.

We use logistic regression model as cost function. At each Tabu iteration, regression needs to be computed n time. However, these computation during the iteration are independent from each other. Only at the end, synchronization is needed to update taboo list depending on which moves are made. Therefore, regression computation can be multi-threaded, thus scaling the performance depending on CPU cores available.

**[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:

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

Local maxima:

(['concave points\_se', 'fractal\_dimension\_se', 'radius\_worst', 'perimeter\_worst'], 0.965034965034965)

(['concave points\_se', 'fractal\_dimension\_se', 'radius\_worst'], 0.95804195804195802)

(['smoothness\_se', 'compactness\_se', 'concavity\_se', 'concave points\_se', 'symmetry\_se', 'fractal\_dimension\_se', 'radius\_worst', 'texture\_worst'], 0.93706293706293708)

(['symmetry\_se', 'area\_se', 'fractal\_dimension\_mean'], 0.87412587412587417)

The initial score with all 30 features is 67%, the low accuracy was caused by underfitting. We got interference from the useless features.

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

The best result is 97.2%. The score is calculated by returning the mean accuracy on the given test data and labels. During the searching process, it could be seen that some of the local best solutions (listed above) is founded. While the Tabu Search avoids being stuck locally to explore other region, therefore reaching even better solutions. Since Tabu search is not designed to find the absolute best solution, it will generate a solution that is close to the best. The set of features that produces the model with the highest prediction accuracy will be defined as the dominant features. In this case, the dominant features are 'texture\_mean', 'perimeter\_mean', 'smoothness\_mean', 'compactness\_se', 'texture\_worst', 'perimeter\_worst', and 'smoothness\_worst'. Note some of these features overlapped with the previous result achieved by BFS.

Previously, BFS is an uninformed search. Therefore, if an optimal solution is at the bottom of the tree, BFS would have to brute-force traverse all the tree to reach it, which is time inefficient. Currently, the Tabu search is informed. Since Tabu search has “memory” and can track previous results, it can use heuristic to intelligently evaluate subsets of features. Tabu Search only compares to the future better solution instead of the checked best solutions; though it starts from a specific solution, it still maintains a holistic view of the search space. When the search is terminated, even if it does not obtain the best solution, it still reaches one of the optimal solutions. The probability of it reaching to an optimal solution before BFS renders it much more efficient.

**[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. This section should be used for all reports other than the Graph Search Algorithms.

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%

Graph Search Algorithm occupies larger memory than the Tabu Search. With n features problem, graph search may generates up to O(2^n) nodes hold all combinations of the features; while the Tabu Search only needs 2 n-length arrays to hold the status. Thus, the space complexity of Tabu Search is far less than that of the graph search. The graph search wastes space since feature selection problem only used a tree traversal instead of path optimization.

For the Tabu Search, a termination criterion is set: besides finding the Tabu solution, it will stop when the algorithm reaches 2^n times of iteration. Therefore, both the graph search and the Tabu search has the time complexity of O(2^n).

The Tabu Search’s final outcome almost the same as that of the BFS. The Tabu is finding the approximate of the absolute best solutions; where the BFS finds the exact best solution if it exists.

*N.B.: Please, note the report should be a maximum of One pages long. Please, submit your report as well as your code to the appropriate dropbox on Learn.  Your code will weigh 1 point of the grade. Please note that you can use existing libraries*

[*https://github.com/YusufLiu/FeatureSelection*](https://github.com/YusufLiu/FeatureSelection)