CS 550 -- Machine Learning

Homework #3

Due: 10:40 (class time), December 18, 2018

Sevil Çalışkan

21701423

1. Introduction

In this homework, our job was to design and implement a genetic algorithm based approach for cost sensitive learning, in which the misclassification cost is considered together with the cost of feature extraction.

I have selected decision trees as classification algorithm to start with. I have selected it since it is a simple one, with a small calculation time considering the size of the training and test sets, iteration number and size of the population in genetic algorithm. Also, it gives good test results with the given data and does not require classes to be balanced.

I have defined a bit string representation to indicate what features are selected and created different parent with the help of this representation. The bit string 100101 indicate that the 1st, 4th, and 6th features are selected, and the remaining ones are not. So, bit string representation of selected features have a length of 21, first digit implying if the first feature is selected or not (1 or 0) and so on. 21st digit is the multiplication of 19th and 20th, since it can be used only if the cost for both is paid. One example of the feature representations or parents can be seen below:

1 0 0 1 0 0 1 0 0 1 0 1 0 0 1 1 0 1 1 0 0

Finding an appropriate fitness function that guides the genetic algorithm was a crucial part. Function should include both the misclassification cost and the cost of extracting the selected features. To do so, firstly I have calculated total cost of a parent, cost of selected features added to the misclassification error of test set with selected features. If the total cost of a parent small, it should be a better fit since it has either small cost of features or small misclassification error or both. So, it means it is the solution which classifiers better with less total cost. Converting total cost to fitness means that, parents with a small total cost should have higher fitness. We can reach that by taking the (-1) power of total cost. Also, converting this fitness into probability makes more sense when choosing parent for next generations. So, I have used those fitness values to calculate probabilities to be chosen, before creating next generations.

$$Fitness(Parent_i) = \frac{1}{missclassification\ error_i + cost\ of\ selected\ features_i}$$

Algorithm starts with generating a population randomly, size being a parameter. So, it creates a random binary population matrix, size*21, each row stands for a parent. Then, for each parent, train and test sets are created with chosen features. Those sets are then fed into to the classification algoritm, which is a decision tree. I have used MATLAB function *fitctree* to fit a tree and *predict* function to get the predicted labels. *confusionmat* function calculates the confusion matrix given real labels and predicted labels. Using this confusion matrix, the number of misclassified observations is calculated by summing all the values except the ones on the diagonal. This is misclassification error. For total cost, binary representation of the selected features for the parent is multiplied elementwise with the cost array, so the costs of the chosen features are multiplied by 1 and others by 0. Summation of result array is the cost of selected features. Summing misclassification error and cost of features, total cost is calculated. With the total cost, fitness values of each parent are calculated and fed into selecting algorithm. Selection algorithm chooses new parents from the population matrix for the next

generation accordingly with the fitness values, with parameter crossover rate. In my implementation, population size*crossover rate is rounded to next even number. This is number of parents to be crossed over. Population size - number of parents to be crossed over gives the number of parents to be moved to the next generation directly. Until this number is reached, parents are chosen with tournament selection. Two parents are randomly selected, then the fittest one among these two is selected and deleted from the population matrix. These two steps are repeated until the number is reached. Parents left are shuffled and crossed over from a random point with single point crossover. Then new generation is mutated with a mutation rate parameter. Number of genes (or features in our case) to be mutated are calculated by total number of genes * mutation rate. Until this number is reached, random numbers are generated between 1 and total number of genes and those genes on the random numbers are mutated. Those steps are repeated until a number of iteration parameter is reached.

I have chosen parameters of the algorithm with trial and error. I have tried some different values and changed them following the results getting better or worse. I have set the iteration number to 100, since after it, results do not get better. I have chosen crossover rate as 0.5 since it make sense letting half of the parents to get to the next generation and half to be cross overed. I have set mutation rate to 0.01 since when it is high, the odds of fittest individuals also get mutated increases but we still want to get the diversity that the operation brings.

2. Pseudocode

Pseudocode of the algorithm is given below:

// This algorithm tries to choose best features set of the dataset for better classification accuracy with smaller cost for features extracted, given the costs of the features.

Set **colNum** to column number of train (or test) data. //*For number of features.*

Set **popNum** to colNum^2 *//For population size.*

Set **crRate** to 0.5 *//For cross over rate of population.*

Set **mRate** to 0.01 *//For mutation rate.*

Set **iterations** to 100 *//For mutation rate.*

Set **pop** to randomly generated binary matrix with size **popNum*colNum** and set **pop**(column 21) = **pop**(columns 19) ***pop**(column 20) (row = popNum, columns = colNum) *//create a random population at first.*

Set i = 0

While (i < iterations)

> Create train and test sets for everyone in the population (every row in **pop** matrix) with chosen features and store them in lists **newtrain** and **newtest**
>
> Calculate number of misclassified observations with selected features for every feature set in population.
>
> > Fit decision tree with each element of **newtrain** and the real labels of train set
> >
> > Predict labels for every element of **newtest**
> >
> > Create confusion matrix with predicted labels and test labels and calculate number of misclassified observations as **missccost** (array of length popNum / population size) by summing everything in the matrix except the diagonal for everyone in the population.

Calculate cost of selected features for everyone in population as **fcost** (array of length popNum / population size) by multiplying costs and feature representations element-wise and then summing them for everyone in the population

Add **fcost** and **missccost** to calculate **totalCost** (array of length popNum / population size) for everyone in the population

Calculate fitness values of every feature set in the population as 1 divided by **totalCost** and calculated probabilities as fitness divided by total fitness of population as **fit** (array of length popNum / population size) for everyone.

Hold the population in another variable named **popB**

Update pop with new generation with tournament selection, cross-over and mutation operations.

Round **popNum*crRate** to next even number as **crNum** //*For number of parents to be crossed over*

Calculate **popNum** - **crNum** as **select** *//For the number of parents to be moved to the next generation directly*

j = 1

Create **newPop** matrix at size **popNum*colNum**

While (j < select)

Create two random numbers between 1 and **popNum** which are not equal //Select two parents randomly

Put the one with the highest **fit** value among the two parents at the index of those random numbers in **pop** matrix to **newPop** matrix

Delete selected parent from the **pop** matrix

j = j + 1

End

Cross-over the pop matrix *//left parents for new generation*

Shuffle **pop** matrix

For each ordered pair of the shuffled **pop** matrix

Create a random number between 1 and **colNum** as **rnd**

Change columns (features) of the first parent from **rnd** to **colNum** with second parent *//Single point crossover*

Put new parents to **newPop** matrix

End

Mutate some genes form **newPop** matrix

Calculate total number of genes to be mutated by **popNum * colNum * mRate** as **mNum**

m = 0

While (m < nNum)

Generate random numbers between 1 and **mNum** as **rnd**

Mutate rndth gene on **newPop**

End

End

pop = **newPop**

pop(column 21) = **pop**(columns 19) ***pop**(column 20)

i = i + 1

End

3. Results

After 100 iteration with the given parameters, my algorithm finds the feature set with the least cost as below:

0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 1 1 1

So, it chooses 3rd, 8th, 15th, 16th, 17th, 19th and 20th features. Since cost for 19th and 20th features are already paid, we also get the 21st feature. Names of the selected features are onthyroxine, thyroidsurgery, hypopituitary, psych, TSH, TT4, T4U. Total cost of the selected set is 76.70 with cost of selected features is 52.70 and misclassification cost is 24. Train set accuracy of the selected features with decision tree algorithm is 0.9992 and test set accuracy is 0.9930.