FEATURE SELECTION FOR CLASSIFICATION

Department of Electrical and Computer Engineering

ECE 569A Artificial Intelligence

PROJECT REPORT

Submitted by

Alinstein Jose, V00918748

Sai Prakash Reddy Konda, V00901689

Tony Massoud, V00903118

University of Victoria

Date: July 22nd, 2019

Table of Contents

List of Tables

Figure Name **Page Number**

List of Figures

Figure Name **Page Number**

Abstract

Because of the latest improvements in the areas of Computer Science and Technology, it has become easy for the users to quickly find useful or needed information. Data mining can be viewed as an area in artificial intelligence which tends to extract information or patterns from large amounts of data stored in databases/ datasets[1][8]. Recent research on feature selection has been conducted in an attempt to find efficient methods for selecting the better and relevant features.

Feature selection usually involves a combination of search techniques and attributes utility estimation plus evaluation with respect to specific learning schemes. There are several methods to select features in ensemble systems. Genetic algorithms (GA) are one of the most useful methods for feature selection in order to do classification[1].

Feature Selection is one of the main concepts in machine learning which will have a huge impact on the performance of the machine/ model. The data features that we use to train the machine learning models will have a huge influence on the performance you can achieve. Irrelevant or partially relevant features can negatively impact model performance. Feature selection and Data cleaning should be the first and most important step of your model designing.

Everyone must have faced the problem of identifying the related features from a set of data and removing the irrelevant or less important features which do not contribute much to the target variable in order to achieve better accuracy for our model[1].

This project tries to solve the above problem and gives overview of feature selection algorithm implemented for classification using genetic algorithms, which will search the feature space using the idea of evolutionary computation/ various genetic algorithm methods, in order to find the optimal feature subset.

1. Introduction

1.1 Background

Our group consists of 3 graduate students, namely:

Mr. Alinstein Jose: Alin is a graduate student in Dr. Sheng Lu's lab and he is working in areas of Optimization, Machine Learning etc., He is currently taking a graduate course (ECE-503-Optimization for Machine Learning) and is also planning to take CSC-503-Data Mining course.

Mr. Sai Prakash Konda: I am a graduate student in Dr. Daler Rakhmatov's lab, working in the area of Computational Optimization in Ultrasound (Medical) Imaging using Modern Day Learning Techniques. I took ECE-535-Data Analysis and Pattern Recognition course and currently taking ECE-503 with Alin.

Mr. Tony Massoud: Tony is very much interested in the areas of Computer Security and Forensics and is working towards implementation and designing of new methods in those areas using the knowledge of Artificial Intelligence and Machine Learning.

1.2 Motivation

All of us are working and taking courses in the area of Artificial Intelligence, Machine Learning in particular.

In Machine Learning, Feature Selection is a very important concept. In fact, there are various machine learning techniques available today which helps us in calculating and obtaining the features for high dimensional data.

Using genetic algorithms for feature extraction and classification is one among them. This method has advantages compared to the other existing methods. At each level, population with good and healthy features are carried on to the next stage to give new off springs. Weak parents are discarded at each stage. By which, our final output will have output with good features.

Which is the reason we have chosen this project topic. We believe that this project is more related to our area of research and interests. Also, it helps us in our learning, improving and developing the practical knowledge in the area of genetic algorithms.

2. Related Work

2.1 Paper-I

Genetic algorithms in feature selection

This paper uses a genetic algorithm (GA) for the feature selection problem. The method explores the space of possible subsets to obtain the set of features that maximizes the predictive accuracy and minimizes irrelevant attributes. Introduce a multiple correlation in a fitness function used by the GA to evaluate the fitness of each feature subset regarding relationship in its domain.

2.2 Paper-II

Feature selection using genetic algorithm to improve classification in network intrusion detection system

In this paper, they have presented Genetic Algorithm based optimized feature selections for intrusion detection systems. And have used one-point crossover for the Genetic Algorithm parameters instead of two-point crossover used by the previous research as it one-point crossover is faster. For evaluations, we used the NSL-KDD Cup 99 data set and we modified the data set by looking into to the recent attacks, hence making the data set more relevant to the current situations. They have used Random Forest as the classifier because it gave the best results in terms of the classification rate and the training time.

2.3 Paper-III

Feature extraction, feature selection and machine learning for image classification

Madalina et al[4] has implemented some feature extraction, feature selection and machine learning based classification methods for pollen grain recognition and classification from the images. This paper investigates and compares a variety of feature extraction/ selection and machine learning based classifiers with its method. The results of this method outperformed the results of the existing methods.

3. Problem Formulation

3.1 Search Space

The dataset considered in this project is an open-source dataset and is downloaded from UCI Machine learning Repository. It is also available as inbuild MATLAB dataset in Statistics and Machine Learning Toolbox™. It is called the Ionosphere dataset. The size of this dataset is: 351 X 35. Among 35 columns, 34 columns represent a number of features of the dataset and the last column (i.e. 35th column) represents the label for the dataset.

If our dataset has F features, then the search space will be 2^F. We know that, for Ionosphere dataset, F is equal to 34 and so, search space will be equal to 2^{34}.

Our objective function is to minimize the number of features for selection and classification. It determines the efficiency to classify the dataset given, so that we can reduce the feature dimensions of the dataset.

3.2 Encoding

The encoding method is a technique to represent the individuals in a prescribed structure/ format, so that they convey necessary information.

In the above description, we mentioned that our search space will be of size 2^{34}. But the reason we have taken the base value as 2 is that, in this project, we are using binary encoding method. The entries will have values either 0 or 1.

It is considered to be the simplest and most widely used method of representing the individuals. One can comfortably perform boolean operations on the individuals[6]. Here, 1 represents that feature is a better feature and is selected, 0 represents feature is bad and is rejected. For example if the dataset has dimension of 10. If I am only selecting first 6 features from the dataset. Then my encoded array is [1 1 1 1 1 1 0 0 0 0].

3.3 Fitness Function

A fitness function is used to tell how close a given design solution is to the actual/ expected solution. They are used in genetic algorithms to guide the simulations towards the optimal design solutions[8].

The fitness function used for solving the problem depends on method of classifier used. In our project, we have chosen 4 various classification techniques, namely:

1. K-means Nearest Neighbor (KNN) classifier,
2. Logistic Regression classifier,
3. Decision Tree classifier and
4. Support Vector Machine (SVM) classifier.

MATLAB has fitness functions inbuilt for all of these classifiers, namely, fitcknn() for K-means Nearest Neighbor, fitclinear() for Logistic Regression, fitctree() for Decision Tree and fitcsvm() for Support Vector Machine.

Note that all these functions need variables such as feature vector (reduced from 351 X F to 351 x m, where F is equal to 34 and m < 34, depends upon number of 1's in that chromosome) and label of the dataset. We have evaluated our machine learning model *K-fold Cross-Validation(CV)* function in MATLAB because error metric to determine the accuracy of the model is not very reliable as the accuracy obtained for one test set can be very different to the accuracy obtained for a different test set. K-fold Cross-Validation(CV) provides a solution to this problem by dividing the data into folds and ensuring that each fold is used as a testing set at some point. It is known and understood that KNN classifier needs K value (K-stands for number of nearest neighbors) to be inputted along with all the other required (above mentioned) variables.

4. Methodology

4.1 Approach

The block diagram for explaining our approach is as follows:

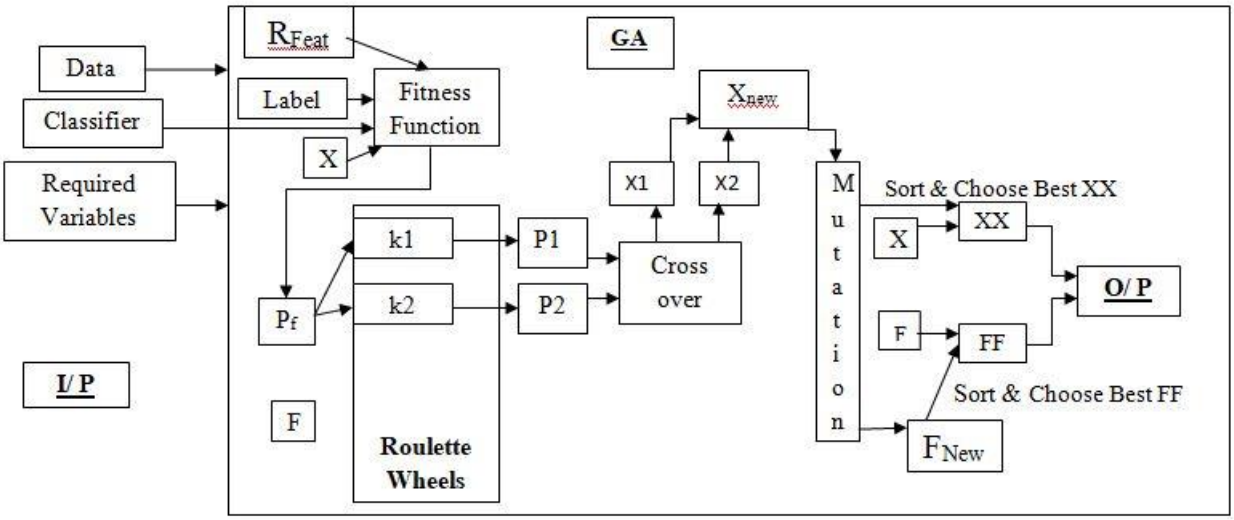

Fig. 4.1: Block Diagram of our Approach

The description of the above blocks in the diagram are as follows:

(i): Data: Input Ionosphere Data (351 X 34).

(ii): Classifier: One among the above mentioned classifiers is chosen.

(iii): Required Variables: We need some external variables as inputs such as number of chromosomes(N = 10), number of generations (T = 20,50,100), mutation (MR = 0.01) Cross Over Ratio (C_{R1} = 0.8) and Number of Crossovers (C_{R2} = 8) etc.,

(iv): R_{Feat}: Reduced feature vector.

(v): X: Initial population Matrix.

(vi): P_f: Probability of fitness function (calculated probabilities for fitness values).

(vii): k1, k2: Intermediate values obtained using roulette wheel.

(viii): P1 and P2: Parents 1 and 2.

(ix): X1 and X2: 2 Outputs obtained from crossover operation.

(x): X_{new}: Merged version of X1 and X2.

(xi): F_{New}: Fitness vector obtained after mutation.

(xii): F: Initial fitness in the beginning.

(xiii): FF: Merged version of F and F_{New}.

The working procedure is explained as follows:

The dataset chosen for this project is Ionosphere data. Its size is 351 X 35, denoted by A. Initial feature vector is formed using A, its size is 351 X 34. The last column is the label of the data and is denoted by the label itself. One should choose a classifier among 4 available options and then, the required variables mentioned above are supplied into the GA block. We initialize matrix X as an initial population matrix. In the first generation, randomly 1's are inserted into it. Number of 1's inserted are: N * D. Based on the 1's present in each row of X, we reduce the size of feature vector row each time. This reduction is done in order to calculate initial fitness function which is of size 1 X 10.

The probabilities of fitness values are calculated and denoted by P_f. Using this P_f, 2 roulette wheels are tuned to select/ come up with 2 indices, denoted by k1 and k2. Now using these values, we can create 2 parents of size 1 X 34 each, whose value will be equal to k1th and k2th row of X. Then a random index is chosen between 1 and 34. This index value serves as a cross over index.

Crossover operation is applied on parents and this process repeats for CR times, where C_{R2} = round(C_{R1}*N). Here, C_{R1} is crossover rate and N is number of chromosomes. C_{R2} is the number of crossovers. Crossover operation outputs two X matrices X1 and X2. They are merged and denoted as X_{new}.

Mutation operation is applied on this X_{new} matrix and F_{New} is calculated from it. F_{New} is the fitness function calculated using updated feature vector, label and X_{new}. X and X_{new} are merged to form XX, F and F_{new} are merged to form FF.

As a final step in the first generation, we should sort the FF in ascending order, choose first 10 values of FF and use them as indices (since these are best values), and accordingly choose rows of XX till N, so that we get new X matrix for next generation and then the best chromosomes. The best chromosome will be the very first row. The very first value of sorted FF will be the best fitness value.

This entire process takes place in 1 generation. This process is repeated for all T generations. At the end of each generation, we get the best fitness value converged/ less than that of the previous iteration, best chromosomes, and new X matrix. Of the last generation, we take the best chromosome as our final feature selection matrix.

After the last generation, plot the best fitness values obtained after each generation. We see that the graph will be a decaying exponential, which tells us that the fitness function is getting better and better as number of generations increases. At the same time, we are getting the best and best X matrix after every iteration.

One can also plot another curve graph using these best fitness values to see how the solution converged.

4.2 Validation

After we are done with the above process for all generations, we got T best fitness values. One can observe all these fitness values in the order they were received. We see that the value at one generation will be lower than or equal to its previous generation. That means, the fitness value at the first generation will be highest and the value at the last generation will be the least.

Which clearly says that the solution is getting converged to zero as generations increases. Which means that our method is accurate and is working/ progressing towards the optimal solution.

Also, after completing the above process for till the last generations, and compare initial number of features (i.e. number of features before the process) and final number of features (i.e. number of features after the process).

Note that in this project, all fitness functions are calculated using/ along with Kfold cross-validations, which makes our fitness function values more accurate.

5. Results and Discussion

5.1 Comparison of Solution to Optimal Solution

The optimal solution will have the least fitness value. In order to compare our solution to the optimal solution, we can compare the fitness values of both the solutions. In that way, we will know, how far is our solution to the optimal solution.

Also, the optimal solution will have the highest number of features reduced. We can compare the feature reduction percentage of both the solutions. In that way, we will know, how better is our solution when compared to the optimal solution.

5.2 Graphs/ Charts/ tables

The table which compares various classifiers is as follows:

Name of the Classifier	Generation: 20		Generation: 50		Generation: 100		
	Number of final features	Percentage of Reduction	Number Of final features	Percentage of Reduction	Number Of final features	Percentage of Reduction	Final (Initial) fitness value
KNN	12	64.70%	9	73.50%	9	73.50%	0.074/ (0.134)
Logistic Regression	19	44.11%	21	38.23%	17	67.14%	0.086/ (0.114)
Decision Tree	18	47.05%	23	32.33%	15	55.88%	0.07/ (0.094)
SVM	23	32.33%	19	44.11%	18	47.05%	0.08/ (0.113)

Table 5.1: Comparison of Various Classifiers, their performance.

KNN:

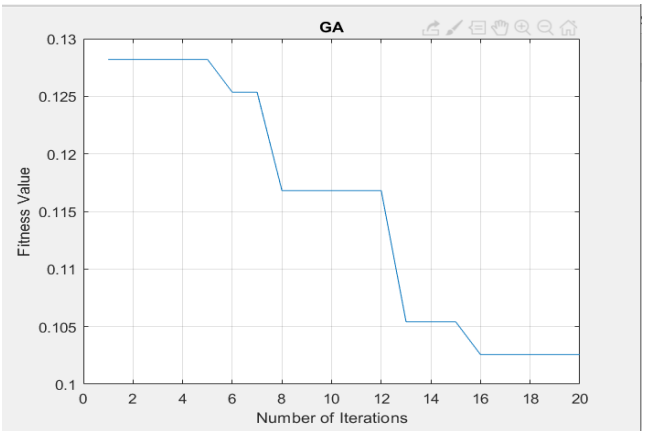

Figure 5.1.1: Plot of Fitness value for 20 generation using KNN Classifier.

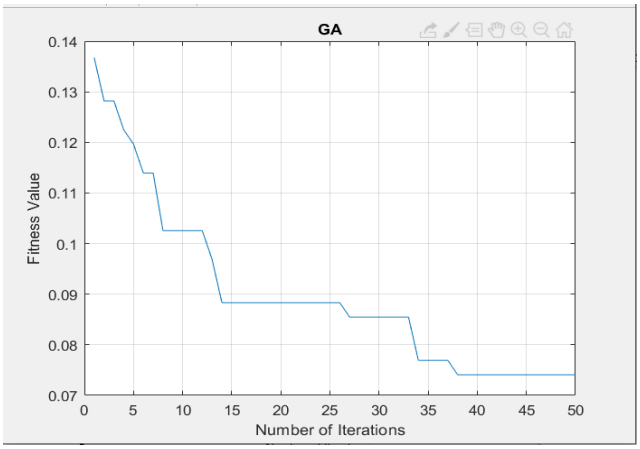

Figure 5.1.2: Plot of Fitness value for 50 generation using KNN Classifier.

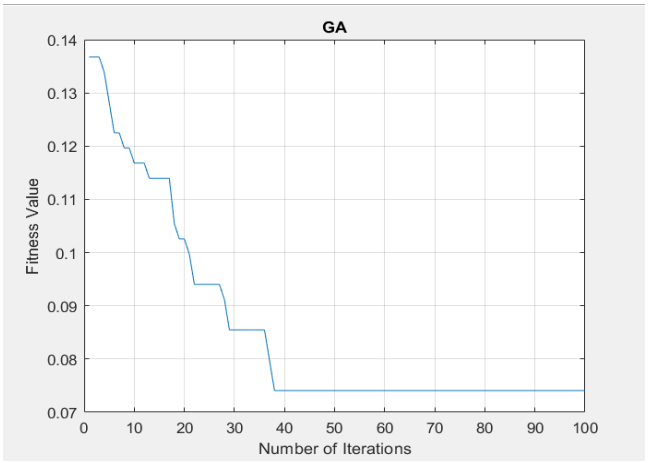

Figure 5.1.3: Plot of Fitness value for 100 generation using KNN Classifier.

Logistic Regression:

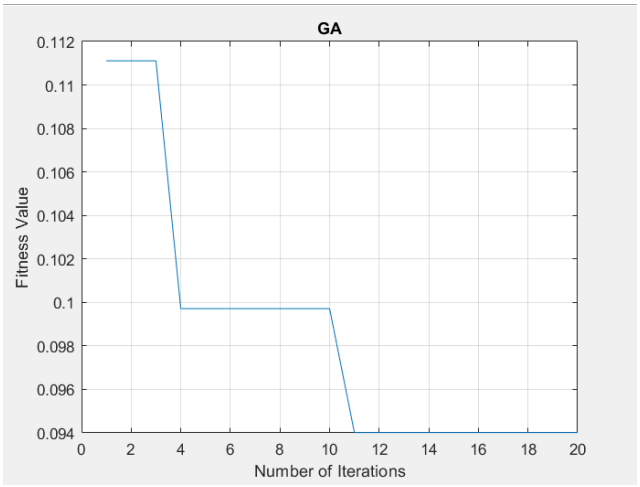

Figure 5.2.1: Plot of Fitness value for 20 generation using logistic regression

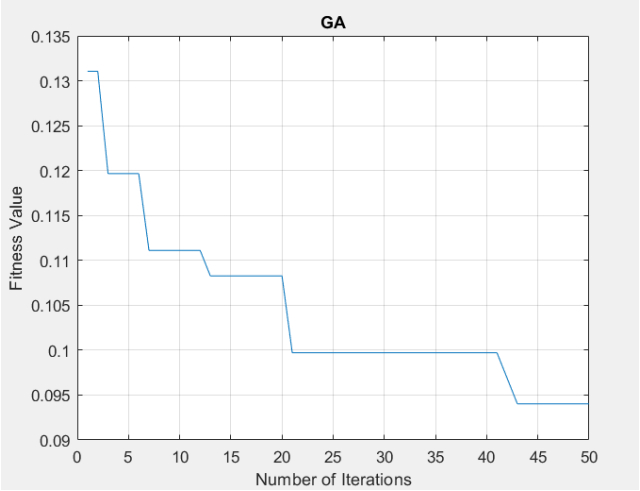

Figure 5.2.2: Plot of Fitness value for 50 generation using logistic regression

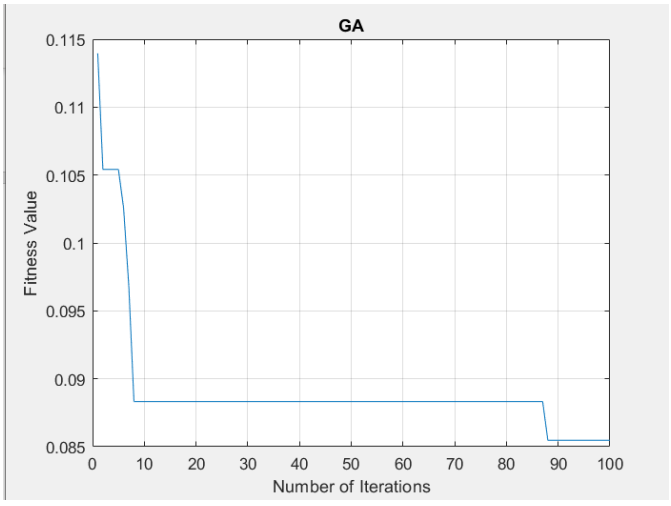

Figure 5.2.3: Plot of Fitness value for 100 generation using logistic regression

Decision Tree:

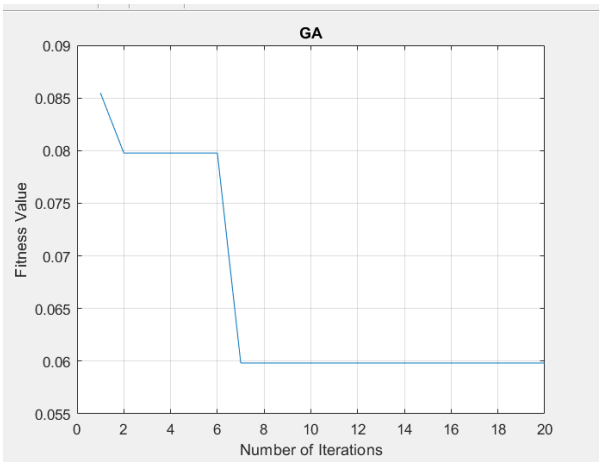

Figure 5.3.1: Plot of Fitness value for 20 generation using Decision Tree

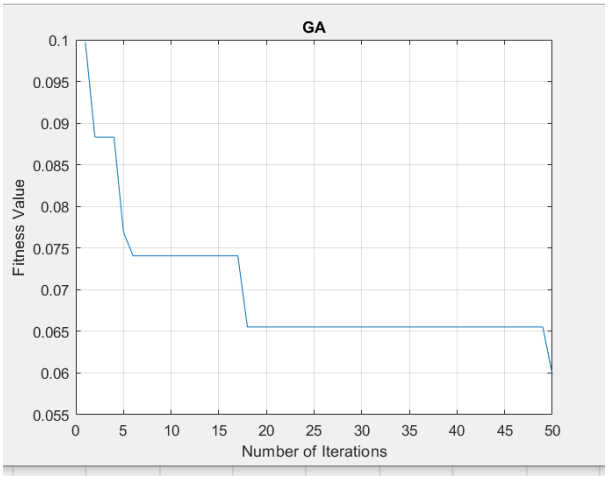

Figure 5.3.2: Plot of Fitness value for 50 generation using Decision Tree

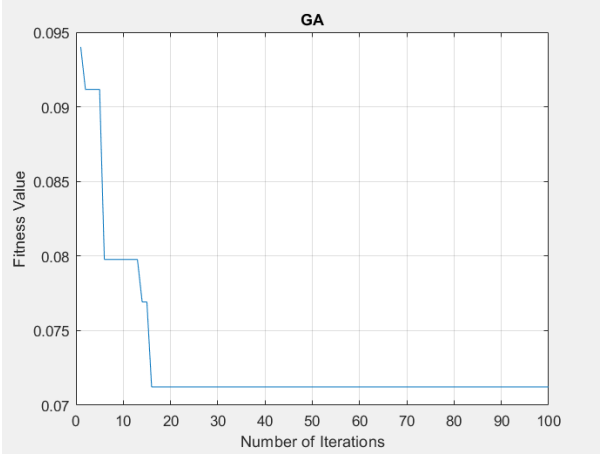

Figure 5.3.3: Plot of Fitness value for 100 generation using Decision Tree

SVM:

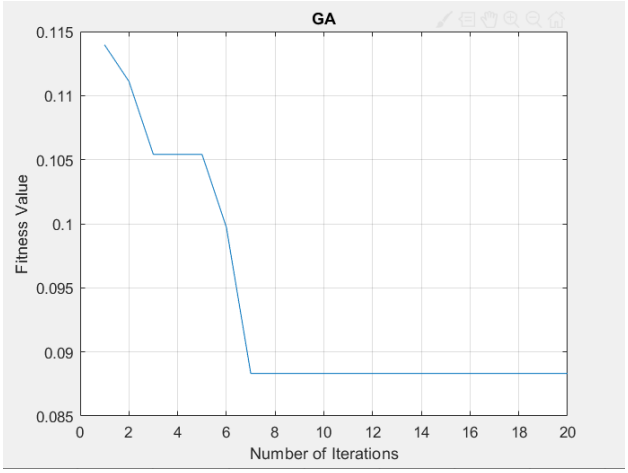

Figure 5.4.1: Plot of Fitness value for 20 generation using SVM

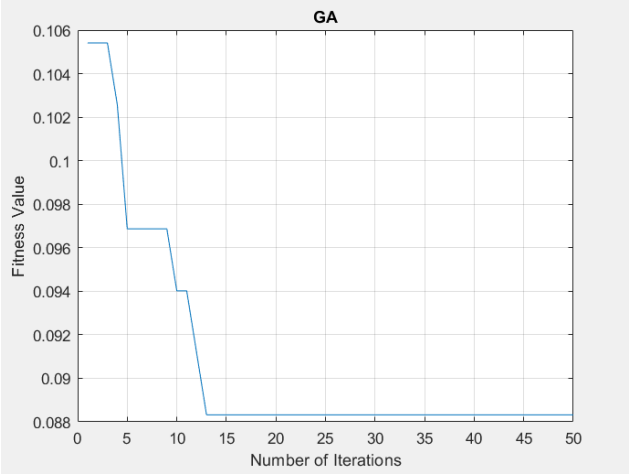

Figure 5.4.2: Plot of Fitness value for 50 generation using SVM

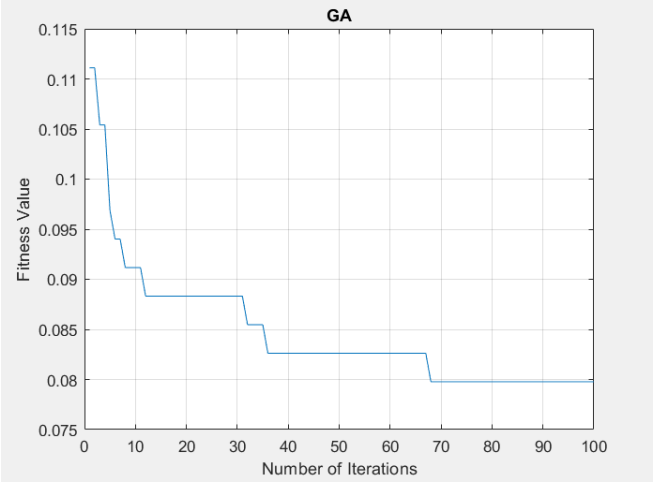

Figure 5.4.3: Plot of Fitness value for 100 generation using SVM

5.3 Justification of Obtained Results

Theoretically speaking, a method is considered to be performing better if it produces at least 10% better result. We see that in this project, up to 73.50% feature reduction is achieved for 50 and 100 generations in KNN. Which means that feature data set contains many redundant features.

At 100 generation we got the following results:

- With the KNN algorithm we were able to reduce from 34 to 9 features, and the fitness value from 0.137 to 0.074.
- With Logistic Regression we were able to reduce from 34 to 17 features and the fitness value from 0.114 to 0.086.
- With Decision tree, we were able to reduce from 34 to 15 features and the fitness value from 0.094 to 0.07.
- With Support Vector Machine we were able to reduce from 34 to 18 features and the fitness value from 0.113 to 0.08.

Applying genetic algorithm on four classifiers we observed that the number of features is reduced drastically as we increase the number generation.

While comparing the graphs of the same classifier we see that the number reduced features or reduction in fitness values are random, or the genetic algorithms give different results while running it each time.

The fitness function values approaching zero means that the classifier is able to classify at higher accuracy and one can say that the obtained solution is approaching the optimal solution.

5.4 Challenges Faced

We faced a few challenges during this project. They are:

(i). Choosing an appropriate dataset: We have tried our algorithm on various datasets such as breast cancer dataset, spam base dataset and hand written digits dataset, etc., Since the size of the datasets is very large, it took more than 40 to 50 minutes to give out the results. So we had to choose a dataset, whose size is reasonable.

(ii). Thermal shutdown of Computer's: Since the previously considered datasets were very large, the system had to do many computations, which resulted in thermal heating of the computer and resulted in thermal shutdown. Some may argue that this problem/ challenge is computer specific and can be resolved by trying it on a different computer. I tried it on my hp laptop Intel Core I7 processor, my group mates tried it on their laptops. But the issue existed, until we worked on a different data set.

(iii). Using UVic's Computers: We thought of using UVic's computers (modern, fast and accurate) and work on earlier datasets. But since this is a course project, we may be asked to execute the results in the class. Hence we have chosen Ionosphere dataset (a reasonable dataset for our course project).

6. Conclusion

In our project, we have run our algorithm at 20, 50 and 100 generations. At the 100 generation we got 73.5%, 47.05%, 67.14% and 55.88% of feature reduction are observed using our genetic algorithm for KNN, SVM, Logistic regression and Decision Tree classifiers respectively. From the observation, we conclude that the genetic algorithm can select the relevant features from the data set and reduce the number of features by removing the redundant features.

7. Future Work

Presently, the algorithm continues for a fixed number of generations. Number of generations is a user-defined input. But this need not be a good measure. Instead, the algorithm can be modified to terminate after a desired level of feature reduction is achieved. That is, making algorithm independent of number of generations, but dependent on number of features reduced.

The algorithm can also be modified to achieve a better final fitness value. As w now, fitness value gives information about how closer the obtained solution is to the ideal/ optimal solution. We can give a new input variable such as desired fitness value and allow the algorithm to continue until the desired value requirement is met. The algorithm can be terminated after the obtained solution will have a fitness value less than or equal to the desired value.

Appendix

Main.m:

```
%  Genetic Algorithm (GA)

clc,
clear,
close all

%% Ionosphere -------------------------------------------------------------

load ionosphere.mat;
feat = f;
label = l;

%---Input-----------------------------------------------------------------
% feat:  feature vector (instances x features)
% label: labelling
% N:     Number of chromosomes
% T:     Maximum number of generations
% CR:    Crossover rate
% MR:    Mutation rate
% selclass:Classification
% *Note: k-value of KNN & k-fold setting can be modified in FitnessFunc.m
%---Output----------------------------------------------------------------
% sFeat: Selected features (instances x features)
% Sf:    Selected feature index
% Nf:    Number of selected features
% curve: Convergence curve
%-------------------------------------------------------------------------

disp("Select the classifier:")
fprintf('\n');
disp("    KNN ---------------------- 1")
disp("    Logistic Regression ----- 2")
disp("    Decision tree ----------- 3")
disp("    SVM --------------------- 4")
fprintf('\n');
selclass = input('Classifier: ');

%%GA
close all; N=10; T=20; CR=0.8; MR=0.01;
[sFeat,Sf,Nf,curve]=GA(feat,label,N,T,CR,MR,selclass);

% Plot convergence curve
figure(); plot(1:T,curve); xlabel('Number of Iterations');
ylabel('Fitness Value');


if selclass == 1

    title('Genetic Algorithm using KNN Classifier');
```

```
    curve_01 = curve;

end


if selclass == 2

    title('Genetic Algorithm using Logistic Regression Classifier');
    curve_02 = curve;

end


if selclass == 3

    title('Genetic Algorithm using Decision Tree Classifier');
    curve_03 = curve;

end


if selclass == 4

    title('Genetic Algorithm using SVM Classifier');
    curve_04 = curve;

end

grid on;

disp("Number of intial features ")
size(feat,2)
disp("Number of features after optimization using GA")
Nf
```

Roulette Wheel.m:

```
function Route=RouletteWheelSelection(Prob)
% Cummulative summation
C=cumsum(Prob);
% Random one value, most probability value [0~1]
P=rand();
% Route wheel
for i=1:length(C)
    if C(i) > P
    Route=i; break;
  end
end
end
```

GA.m:

```
%-------------------------------------------------------------------------%
%  Genetic Algorithm (GA)
%-------------------------------------------------------------------------%

function [sFeat,Sf,Nf,curve]=GA(feat,label,N,T,CR,MR,selclass)
%---Inputs-------------------------------------------------------------------
% feat:  features
% label: labelling
% N:     Number of chromosomes
% T:     Maximum number of generations
% CR:    Crossover rate
% MR:    Mutation rate
%---Outputs------------------------------------------------------------------
% sFeat: Selected features
% Sf:    Selected feature index
% Nf:    Number of selected features
% curve: Convergence curve
%-------------------------------------------------------------------------------

% Objective function
fun=@FitnessFunc;
% Number of dimensions
D=size(feat,2);
% Initial population
X=zeros(N,D); fit=zeros(1,N);
for i=1:N
  for d=1:D
    if rand() > 0.5
      X(i,d)=1;
    end
  end
end
% Number of crossover
CR=round(CR*N);
% Fitness
for i=1:N
  fit(i)=fun(feat,label,X(i,:),selclass);
end
% Pre
X1=zeros(CR,D); X2=zeros(CR,D); Fnew=zeros(1,2*CR); curve=inf; t=1;
figure(1); clf; axis([1 T 0 0.5]); xlabel('Number of Iterations');
ylabel('Fitness Value');


if selclass == 1

    title('Convergence Curve using KNN Classifier');

end


if selclass == 2
```

```
    title('Convergence Curve using Logistic Regression Classifier');

end


if selclass == 3

    title('Convergence Curve using Decision Tree Classifier');

end


if selclass == 4

    title('Convergence Curve using SVM Classifier');

end

grid on;
%---Generations start-------------------------------------------------------
while t <= T
    disp('Current Generation is ')
    t
    % Convert error to accuracy (inverse of fitness)
  Ifit=1-fit;
  % Get probability
  Prob=Ifit/sum(Ifit);
  % {1} Crossover
  for i=1:CR
    % Select two parents
    k1=jRouletteWheelSelection(Prob); k2=jRouletteWheelSelection(Prob);
    % Store parents
    P1=X(k1,:); P2=X(k2,:);
    % Random select one crossover point
    ind=randi([1,D]);
    % Single point crossover between 2 parents
    X1(i,:)=[P1(1:ind),P2(ind+1:D)]; X2(i,:)=[P2(1:ind),P1(ind+1:D)];
  end
  % Union
  Xnew=[X1;X2];
  % {2} Mutation
  for i=1:2*CR
    for d=1:D
      if rand() <= MR
        % Mutate from '0' to '1' or '1' to '0'
        Xnew(i,d)=1-Xnew(i,d);
      end
    end
    % Fitness
    Fnew(i)=fun(feat,label,Xnew(i,:),selclass);
  end
  % Merge population
  XX=[X;Xnew]; FF=[fit,Fnew];
  % Select nPop best solution
```

```
  [FF,idx]=sort(FF); X=XX(idx(1:N),:); fit=FF(1:N);
  % Best chromosome
  Xgb=X(1,:); fitG=fit(1); curve(t)=fitG;
  % Plot convergence curve
  pause(0.000000001); hold on;
  CG=plot(t,fitG,'Color','r','Marker','o'); set(CG,'MarkerSize',5);
  t=t+1;
end
% Select features based on selected index
Pos=1:D; Sf=Pos(Xgb==1); Nf=length(Sf); sFeat=feat(:,Sf);
end
```

Fitness Funtion.m:

```
%-----------------------------Fitness Function-------------------------------
-------------%
function fitness=FitnessFunc(feat,label,X,selclass)
% Parameter setting for k-value of KNN
k=5;
% Parameter setting for number of cross-validation
kfold=2;
% Error rate
fitness=jwrapper(feat(:,X==1),label,k,kfold,selclass);
end

% Perform KNN with k-folds cross-validation
function ER=jwrapper(feat,label,k,kfold,selclass)

switch selclass
    case 1
        %%KNN
        Model=fitcknn(feat,label,'NumNeighbors',k,'Distance','euclidean');
        C=crossval(Model,'KFold',kfold);
        ER=kfoldLoss(C);
    case 2
       % Logistic Classifer
        Model=fitclinear(feat,label,'KFold',kfold,'Learner','logistic');
        numCLModels = numel(Model.Trained);
        %Mdl1 = Model.Trained{1}
        ER = kfoldLoss(Model);
    case 3
        %Decision tree
        Model=fitctree(feat,label);
        C=crossval(Model);
        ER = kfoldLoss(C);

    case 4
        %SVM
        %Model1=fitcsvm(feat,label,'KFold',kfold,'Standardize',true)
        Model1=fitcsvm(feat,label);
        C1=crossval(Model1);
        ER=kfoldLoss(C1);
    end
end
```

References

[1] "Feature Selection Techniques in Machine Learning with Python", Eleanor Roosevelt, October, 2018.

[2] "Feature Selection Methods on Biological Knowledge Discovery and Data Mining", Hanen Mhamdi, Faouzi Mhamdi, December, 2014.

[3] "New Feature Selection Method based on Neural Network and Machine Learning", Nicole Challita, Mohamad Khalil, Pierre Beauseroy, December, 2016.

[4] " Feature Extraction, Feature Selection and Machine Learning for Image Classification", Madalina Cosmina Popescu, Lucian Mircea Sasu, July, 2014.

[5] "Feature Selection Algorithm for Improving the Performance of Classification**",** Kajal Naidu , Aparna Dhenge, Kapil Wankhade, May 2014.

[6] "ECE-569A: Artificial Intelligence - Lecture Notes**",** Zahara Nikdel, May 2019.

[7] "Artificial Intelligence-A Modern Approach (3rd Edition)", Stuart Russel, Peter Norvig, 2010.

[8] " https://en.wikipedia.org/wiki/Fitness_function", Wikipedia Foundation, May 2018.