Spherical Bounding Classifier using CGP Generated Transforms

WK Wong  Basil Andy Lease
Curtin University Malaysia Miri, Malaysia
E-mail: weikitt.w@curtin.edu.my

Abstract. A Cartesian Genetic Programming (CGP) approach was applied to generate data projection equations (transforms) to project original N-dimensional features space into new 3-dimensional coordinate space followed by a spherical bounding for classification. The solution finding mechanism differs slightly from conventional CGP based machine learning in four ways. Firstly, inputs were weighted to introduce more flexibility for solution finding. Secondly, a dual mutation sequence was introduced to encourage faster convergence. Thirdly, chromosome vector included a 3-dimensional coordinate point and spherical bounding mechanism for classification. Fourthly, the probability of selecting either of the input types (feature input, node sequence, or constant) are made approximately equal to ensure that the input types have equal probability to be in the active equation nodes. The best classification rate on test datasets achieved using 10 fold cross validation was 98.57% (Wisconsin Breast Cancer dataset), 87.78% (Heart disease dataset) and 80.5% (PTMA Indian diabetes dataset).

1. Introduction
Machine Learning (ML) algorithms are algorithms that cluster data into specific groups and can generally be divided into supervised and unsupervised learning. These algorithms play an important role in data mining, processing of information and image processing to classify generated data into either of the classes. Data Classification algorithms have been in constant development with several algorithms such as ANN and SVM among the popular machine learning methods applied. In supervised learning, a set of known labels are provided in which the algorithm fits the estimated output based on the target labels.

Genetic Programming (GP) based ML has several advantages compared to other ML techniques. The first advantage is computational speed. Since most genetic programming machine learning approach outputs equations, this would be ideal. The second advantage is portability. The training may be applied to a desktop environment but targeted for an embedded system. Hence, the structure of the classifier algorithm must be highly portable for application in various target embedded system. The third advantage is interpretability and visualisation of generated data in feature space. This would enable further developers to optimise the classification by tweaking specific parameters.

GP in ML algorithms has been applied and explored previously in other research work mainly utilising a variation of K-Nearest Neighbors (KNN) type of learning mechanism to distinguish the target from outlier class. Most research harness the ability of GP in generating specific solutions in projecting the existing features in a new feature space. Other GP based ML includes projecting the original features into a single dimension followed by a threshold mechanism for classification.
Another type of classifier that functions relatively well is the Support Vector Data Descriptor (SVDD) which projects original feature sets into higher dimension followed by bounding using a hyperspherical distance for classification. This research seeks to explore the spherical bounding classification in SVDD with GP as the data projection mechanism.

The objective of conducting this research is to report the performance of CGP generated transform and further classified using a spherical bounding system. The paper consists of 4 sections. In section 2, several related works will be discussed especially those related to CGP, the primary genetic programming engine. Section 3 will briefly discuss the mutation strategies and mechanism for generating the solutions to transform the original feature set into new feature space. The results and benchmarking will be presented in Section 3. Finally, the analysis and future work to improve this current algorithm will be discussed in Section 4 of this paper.

2. Literature Review
In this section, several related works and the fundamental concepts of these methods will be briefly discussed. Cartesian Genetic Programming (CGP) is a form of Genetic Programming or evolution-based generation of computer programs functions which uses a directed graph to represent a program/function. Each node in the program represents a module of the program/function. CGP was initially proposed to be a solution finding algorithm for an electronic circuit in [1] but was later expanded to other problem [2]. Variations of CGP such as the SMCGP [3] and Embedded CGP [4] were later proposed showing faster solution finding on benchmark problems. Due to the capability of CGP to be able to generate complex function, it has been applied to generate Artificial Neural Networks (ANN) [5] and other classification functions using generated rule sets. CGP may be used to generate complex functions by mutating the chromosome and therefore suitable to be applied in generating complex classification rules. Unlike conventional genetic algorithm, CGP does not apply crossover functions in as they were shown not contributing to solution finding. Hence, only mutation function was applied, most commonly a 1+n mutation strategy as discussed in [1].

Projections of original features into new feature sets were often applied in to reduce the dimensions of features. This is often performed due to the 'Curse of Dimensionality'. Several examples include Principal Component Analysis (PCA), Independent Component Analysis (ICA) and Multidimensional Scaling (MDS). These methods were often applied together with some form of machine learning algorithm such as SVM or ANN for classification purposes. These methods act as dataset filters to reduce noise in the projected feature sets and hence serves as preprocessing prior to classification. In [6], the MDS was applied together with SVM on benchmark datasets. In [7], MDS was applied using a genetic algorithm to project data into lower dimensions while keeping dissimilarities of the sets analysed.

Geometrical classifiers such as SVM and Support Vector Data Descriptor (SVDD) functions on the principle of using geometry to classify the original dataset. SVM uses quadratic programming to select hyperplanes such that the distance between the nearest vectors (support vectors) to the boundary plane selected (target/outlier) is optimally distanced. A weight can be applied to reduce 'slack' variables during training in the soft handover regions during the training phase. 'Slack' variables refer to the misclassification during training phases. SVDD is a version of SVM which uses hyperspheres as classification mechanism instead of hyperplanes. Both algorithms apply kernel functions to project the original features into higher dimensions prior to the hyperplane/hypersphere selection, most notably the RBF kernel functions. Both SVM and SVDD have been applied in various classification work that is consistent with benchmark results.

Genetic programming approach has been applied in various research work such as [8–12]. Most genetic programming classifiers, such as [8–10] utilise a form of KNN classifiers as the underlying classification mechanism. KNN classifiers have a disadvantage when handling soft boundaries due to multiple reference point at the boundary. A spherical bounding system could be a better
solution in handling soft boundaries in classification.

Other Genetic programming approaches include projecting existing features into a 1-dimensional data such as [11, 12] in which classification is performed using a piecewise function. These can be further benchmarked with the results obtained.

3. Methodology

Three datasets were used as the benchmark test for the evolutionary classifier which is the Wisconsin Breast Cancer (WBC) dataset, Heart Disease dataset, and PIMA Indian Diabetes dataset. These datasets were selected due to the binary class in nature and suitable for the current research work (number of features below 40). The details of the datasets can be found in [13]. The classifications performance are tested on 10 fold cross validation which means 90% of the data are applied for training procedure, and 10% were reserved for testing. The details of the classifier are shown in Table 1. As shown, the dataset contains missing values. The missing values are replaced with 0. Various research works have excluded the missing value instances from training and testing such as [11] and hence robustness of classifier to deal with noisy data may not be shown. However, for this research work, all the instances will be included to test the robustness of the algorithm with regards to noisy data. The values are normalised using min-max normalisation where values are normalised to [0, 1].

Similar to conventional CGP [1], the algorithm applied does not include crossover and only involves the 1+4 mutation strategy. The equations for each dimension were generated based on 100 nodes where each node represents two inputs and a function operator.

The classifier essentially maps an N-dimension feature set to a target 3-dimensional coordinate system output i.e. \( f_n : \mathbb{R}^N \rightarrow \mathbb{R}^3 \). The chromosome vector, \( v \) represents the chromosome segment. For each instance of the dataset, a new projected coordinate is generated \((x_m, y_m, z_m)\) by decoding from the chromosome, \( v \). The reference point \((x_t, y_t, z_t)\) is generated from the chromosome, \( v \) segments as shown in Eqs. (1) to (3). Eqs. (4) to (6) shows the segment \( s \) of the chromosome decodes into the 3 dimensional coordinate, \((x_m, y_m, z_m)\) where \( m = 1, 2, \ldots, i \) where \( i \) represents the \( i^{th} \) instances of the dataset. The original dataset \( F = \{ f_1, f_2, \ldots, f_N \} \).

The first 300 segments \( v_1, \ldots, v_{300} \) of \( v \) represent the function and the two inputs for each node. The subsequent 200 segments of the chromosome \( v_{301}, \ldots, v_{500} \) vector \( v \) are decoded into weightage coefficient, \( w_n \) that are multiplied to the inputs. Table 1 show the list of functions and their respective operands where \( w_1 \) and \( w_2 \) represents the weightage, \( w_n \) to the dataset input.

A program length of 100 nodes is allowed in the equation generation in which one node represents two weighted features as a function. Three types of inputs were decoded as the operand to the functions as shown in Table 1. The three possible types of inputs are constants, dataset values and output from previous nodes. The constant input is added and not available in the original CGP [1] as it will be helpful for evolving coordinates near the spherical boundary. The types of inputs are decoded from the chromosome vector, \( v \) values by using lookup approach in which specific range corresponds to either of the input types.

The inputs to the node function (int1 and int2) are multiplied by the respective weight \( w_1 \) and \( w_2 \) as shown in Table 1. The weight to the node input is generated by taking chromosome and dividing the value by 300. Hence, a reduction of the actual input value was applied with a maximum value weight, \( w_n \) equals to 1.

\[
x_t = v_{501} \\
y_t = v_{1004} \\
z_t = v_{1507}
\]
\[ x_m = f_n(v_1, \ldots, v_{500}, f_1, \ldots, f_n) \]  
(4)

\[ y_m = f_n(v_{503}, \ldots, v_{1003}, f_1, \ldots, f_n) \]  
(5)

\[ z_m = f_n(v_{1006}, \ldots, v_{1506}, f_1, \ldots, f_n) \]  
(6)

\[ \text{int} = \begin{cases} 
\text{feature input} & 0 < vi \leq 100 \\
\text{node output} & 100 < vi \leq 200 \\
\text{constant} & 200 < vi \leq 300 
\end{cases} \]  
(7)

where \( f_n \) represents the normalized dataset values \\
\((x_t, y_t, z_t)\) represents the reference point \\
\((x_m, y_m, z_m)\) represents the projected coordinates

The chromosome vector integer value is decoded into the function, various nodes output reference using the directed acyclic graph system where the values are continuously shifted until a valid range is obtained. The CGP equations project the features into new features spaces. A spherical boundary is introduced from the 3-dimensional coordinate \((x_t, y_t, z_t)\). If the projected features lie within the boundary spherical from a selected reference point, it will be classified as the target. The Euclidean distance, \(d_m\) from the selected reference coordinate \((x_t, y_t, z_t)\) is expressed in Eq. (8).

\[ d_m = \sqrt{(x_m - x_t)^2 + (y_m - y_t)^2 + (z_m - z_t)^2} \]  
(8)

Where \(x_m, y_m, z_m\) are the generated values from the CGP generated projection functions and \(x_t, y_t, z_t\) are the selected reference coordinate. \(d_m\) represents the distance of new projected coordinates of instances \(m\) in the normalised dataset values.

The outlier/target class is determined by the following, if \((d_m \leq T_H)\), \((T_H\) is the threshold), then the feature set will be classified as the target class. Else, the features will be classified as outlier class. Table 1 shows the 12 functions, their respective function arity and the equivalent function number that may be applied in each node. For the unary operation, only the first input is considered and the second integer (int2) will be ignored.

In various CGP algorithms, the probability of selecting between the types of inputs (original feature input, node sequence, constants) is not equal. In this current research, the probability of selecting either of the input types is required to be made constant as shown in Eq. (7). As expressed in Eq. (7), the input to functions, int to the nodes can be converted to a feature input, node sequence or constant using the piecewise function in which the probability of function as either type of input is almost equal. The chromosome sections are mutated by replacing the existing value with a uniform distributed random generated integer between \([1, 300]\).

To acquire the feature input and the node output, an acyclic function (circular shift function) was applied in which the value of the chromosome vector segment value \(vi\) is rotated in a circular shift method until a valid value is acquired. The constant value \([0, 1.00]\) is acquired by subtracting the value by 200 and further division by 100.

The optimisation minimises the fitness score as shown in Eq. (9). The mutation to acquire the fitness with the highest fitness was performed using a two-level mutation mode - coarse and
Table 1: Function Operator

| Function | Arity | Integer representation |
|----------|-------|-------------------------|
| \( (w_1 \times \text{int1}) + (w_2 \times \text{int2}) \) | Binary | 1 |
| \( (w_1 \times \text{int1}) - (w_2 \times \text{int2}) \) | Binary | 2 |
| \( (w_1 \times \text{int1}) \times (w_2 \times \text{int2}) \) | Binary | 3 |
| \( (w_1 \times \text{int1})/(w_2 \times \text{int2}) \) | Binary | 4 |
| \( \exp(w_1 \times \text{int1}) \) | Unary | 5 |
| \(- (w_1 \times \text{int1}) \) | Unary | 6 |
| \( (w_1 \times \text{int1})^2 \) | Unary | 7 |
| \( \sqrt{w_1 \times \text{int1}} \) | Unary | 8 |
| \( \cos(w_1 \times \text{int1}) \) | Unary | 9 |
| \( \sin(w_1 \times \text{int1}) \) | Unary | 10 |
| \( \min(w_1 \times \text{int1}, w_2 \times \text{int2}) \) | Unary | 11 |
| \( \max(w_1 \times \text{int1}, w_2 \times \text{int2}) \) | Unary | 12 |

where \( w_1 \) and \( w_2 \) are the multipliers to the input \( \text{int1} \) and \( \text{int2} \).

Fine mutation to accelerate convergence. In Coarse mutation mode (fitness > \(-0.4\)), multiple chromosome sections will be mutated while in fine mutation mode fitness < \(-0.4\), only a single active segment of the chromosome will be mutated as per normal CGP implementation [2]. Active segment refers to the segment which affects the output value. Algorithm 1 shows the overall pseudo code for the evolutionary algorithm using the 1+4 mutation strategy. 1+4 mutation strategy refers to the mutation in which 4 alternative solutions are generated and compared to the existing best solutions. If any better solutions were acquired, the best solution in the pool would be replaced with one of the alternatives.

In the coarse mutation mechanism, the algorithm decides which segment of the chromosome vector \( v \) to be mutated based generation of enabler vector \( v_1 \). \( v_1 \) is a vector of random generated precision numbers [0,1] of the same length as chromosome vector \( v \). The random number generation is uniformly distributed.

The chromosome segment \( v \) is only replaced with a uniformly distributed random integer value [1,300] if the enabler \( v_1 \) segment is above a specified range. In the entire research, a threshold value of 0.80 was applied to the mutation enabler. A mutation in the \( n^{th} \) segment of chromosome \( v \) is applied when the \( n^{th} \) segment of \( v_1 \) (> 0.8). This setting causes approximately 80% of the segments enabled for mutation. The lower the threshold value, the higher the mutation rate using the coarse mutation scheme thereby accelerating converges to a possible solution.

For the fine mutation mode, only an active segment is mutated in the chromosome. Active segment refers to the sections which contribute to the projection equations.

The fitness score for optimisation is the sensitivity *specificity ratio as shown in Eq. (9). The optimisation is based on minimising the score.

\[
\text{Score} = \frac{-TP \times TN}{(TN + TP + FP + FN)} \tag{9}
\]

Where \( TP \) is true positive sets, \( TN \) is true negative sets, \( FP \) is false positive and \( FN \) is the false negative sets.
Algorithm 1 Pseudo code for the proposed CGP

1: Initialisation : Start with initial random ∈ [1,300]
2: while gen = 1:2000 do
3:   for fitness > -0.4 (coarse mutation mode) using 1+4 mutation strategy do
4:       Generate enabler vector v1
5:       for i = 1 : 1507 do
6:         if v1(i) ≥ 0.80 then
7:           replace alternate solutions v(i) with rand(1,300)
8:         end if
9:     end for
10:    end for
11:    for fitness < -0.4 (fine mutation mode) using 1+4 mutation strategy do
12:       Select an active chromosome section
13:       Mutate active chromosome section v(i)
14:    end for
15: end while

4. Results
The projection and classification are performed using 10-fold cross validation (90% of the instances for training and 10% of the instances for testing). The algorithm stops running and assumes the best solution after 2000 generations (gen). The CGP ran for 15 times for each type of dataset, and the average specified is acquired by averaging for all the testing/training classification rates. Table 2 and Table 3 shows the classification rate of the algorithm on the 3 benchmark datasets. It is noteworthy that the partitioning of the testing and training datasets were different in some of the research work and thus a direct comparison might not be applicable. The average values are from acquired by calculating the mean of the recognition rate from 15 set of optimisation for each dataset.

Results on the Benchmark test (Tables 4 to 6) showed the testing results were competitive as compared to the other benchmarking results. Among the three benchmarking datasets, the heart disease dataset contained the highest number of missing data but gave the best results as compared to the others. The difference between the best and the average classification on performance training/testing shows the level of inconsistency in generating best solutions. There was no conclusion as to which threshold on spherical distance was better (0.5 or 1.00).

A high value of spherical distance threshold for classifying projected coordinate as target/outlier (e.g. 1000) was not considered as most of the outputs of the functions would generate a precision output [0,1] unless divided by a precision output. However, this will only require more computations. Due to this reason, only two threshold values for target/outlier classification was considered which was 1.00 and 0.5. Table 2 shows results of distance threshold value (≤ 0.5) while Table 3 shows the results of distance threshold value (≤ 1.00).

Benchmarking is performed by comparing the results with other algorithms. Tables 4 to 6) shows the benchmarking with other research work. The benchmark results in Tables 4 to 6) were updated from work presented in [11]. The results are based on test datasets in which different approaches were applied hence not directly comparable. The results are shown in which
Table 2: Results of training and testing based on 0.5 distance from reference point

| Dataset  | Best Training (%) | Best Testing (%) | Average Training (%) | Average Testing (%) |
|----------|-------------------|------------------|-----------------------|---------------------|
| Cancer   | 97.14             | 98.57            | 96.46                 | 96.63               |
| Diabetes | 75.83             | 80.51            | 72.48                 | 77.62               |
| h. Disease | 80.30            | 80.43            | 78.54                 | 77.25               |

Table 3: Results of training and testing based on 1.0 distance from reference point

| Dataset  | Best Training (%) | Best Testing (%) | Average Training (%) | Average Testing (%) |
|----------|-------------------|------------------|-----------------------|---------------------|
| Cancer   | 97.40             | 98.57            | 96.71                 | 96.09               |
| Diabetes | 76.41             | 80.50            | 70.65                 | 70.50               |
| h. Disease | 80.10            | 84.78            | 78.77                 | 75.57               |

Table 4: Benchmarking of current work with other research work on Wisconsin Breast Cancer dataset

| Method                                         | Accuracy % |
|------------------------------------------------|------------|
| Radial Basis Function Networks [14]            | 49.8       |
| Probabilistic Neural Networks [14]             | 49.8       |
| ANN (Back Propagation) [14]                    | 51.9       |
| Recurrent Neural Network [14]                  | 52.7       |
| Competitive Neural network [14]                | 74.5       |
| Support Vector Machine [15]^{1}                | 96.9       |
| Memetic Pareto ANN [16]                        | 98.1       |
| ANN (Back Propagation) [16]                    | 98.1       |
| Genetic Programming [17]^{2}                   | 98.2       |
| Support Vector Machine [18]^{2}                | 98.4       |
| MT-CGP [11]                                    | 99.3       |
| Data projection using CGP^x (best)             | 98.57      |
| Data projection using CGP^y (best)             | 98.57      |

x represents the results with 0.5 distance threshold from the reference point and y represents the 1.0 distance threshold value from the reference point.

^{1} Based on distance 1.0 from \((x_t, y_t, z_t)\)

^{2} Based on distance 0.5 from \((x_t, y_t, z_t)\)

^{3} Results are based on leave-one-out validation

^{4} Results are based on 10 fold validation

^{5} This paper does not use the pre-defined training and validation split
Table 5: Benchmarking of current work with other research work on Pima Indians Diabetes dataset

| Method                                      | Accuracy % |
|---------------------------------------------|------------|
| Self-generating Neural Tree (SGNT) [19]     | 68.6       |
| Learning Vector Quantization (LVQ) [19]     | 69.3       |
| 1-Nearest Neighbor [19]                     | 69.8       |
| Self-generating Prototypes (SGP2) [19]      | 71.9       |
| Linear Genetic Programming [12]             | 72.2       |
| k-Nearest Neighbor [19]                     | 72.4       |
| Self-generating Prototypes (SGP1) [19]      | 72.9       |
| Gaussian mixture models [19]                | 72.9       |
| Neural Network [12]                         | 75.9       |
| Infix Form Genetic Programming [20]\(^3\)   | 77.6       |
| Support Vector Machine [18]\(^2\)           | 77.6       |
| Principal Curve Classifier [21]             | 78.2       |
| MT-CGP [11]                                 | 79.2       |
| Data projection using CGP\(^x\) (best)      | 80.51      |
| Data projection using CGP\(^y\) (best)      | 80.50      |

Table 6: Benchmarking of current work with other research work on Heart disease dataset

| Method                                      | Accuracy % |
|---------------------------------------------|------------|
| Neural Network [13]                         | 80.3       |
| Linear GP [12]                              | 81.3       |
| Support Vector Machine [18]\(^2\)           | 83.2       |
| Infix Form GP [20]                           | 84.0       |
| MT-CGP [11]                                 | 85.3       |
| Data projection using CGP\(^x\) (best)      | 80.43      |
| Data projection using CGP\(^y\) (best)      | 84.78      |

The best solution for Cancer datasets are as expressed using Eqs. (10) to (12). The equations project existing feature (f1-f7) into new 3-D coordinates \((x_m, y_m, z_m)\). The classification is based on the distances from the reference point \((x_t, y_t, z_t)\) in which a Euclidean distance of \(> 0.5\) is classified as the target. The equations will produce a 96.99% accuracy over the entire breast cancer dataset when validated i.e. 678 correct classified sample out of 699 samples of breast cancer datasets. The reference point \((x_t, y_t, z_t)\) selected was \((0.1500, 0.3233, 0.2167)\) and the threshold to classify as target class is \((\leq 0.5)\) from the reference point.

\[
x = \frac{0.0352}{0.9833 \times f5}
\]

\(^{a}\) Based on distance 0.5 from \((x_t, y_t, z_t)\)
\(^{b}\) Based on distance 1.0 from \((x_t, y_t, z_t)\)
\(^{1}\) Results are based on leave-one-out validation
\(^{2}\) Results are based on 10 fold validation
\(^{3}\) This paper does not use the pre-defined training and validation split
The data projection equations for heart disease dataset are as expressed by Eqs. (13) to (16). The equations will produce an overall accuracy of 79.78% over the entire heart disease dataset. The reference point \((x_t, y_t, z_t)\) is \((0.7666, 0.1866, 0.6700)\), and the threshold distance is 1.0 from the reference point.

\[
x = e^{0.17666 \times f6} \tag{13}
\]

\[
y = 0.6133 \times f22 + 0.15146 \tag{14}
\]

\[
z = 0.5600 \times f32 + (0.04261 \times f5 - 0.1467 \times n_{14}) \tag{15}
\]

\[
n_{14} = \max [0.0396441, 0.23666 \times f24] \tag{16}
\]

The data projection equations for PIMA Indian diabetes dataset are as expressed by Eqs. (17) to (19). The equations will produce an overall accuracy of 76.30% over the entire dataset. The reference point \((x_t, y_t, z_t)\) is \((0.3700, 0.0071, 0.0800)\) and the threshold distance of < 0.5 from the reference point is classified as target class.

\[
x = 0.3990 \times f6 \times f1 \tag{17}
\]

\[
y = \sqrt{0.183 \times f2} \tag{18}
\]

\[
z = (0.71 \times f8)^2 \tag{19}
\]

The visual representation of the data projection using the Eqs. (10) to (19) is shown in Fig. 1-3 for the respective datasets. The 'red' coloured spheres represent an 'outlier' labelled class while the 'blue' coloured spheres indicate a 'target' labelled class. The green sphere represents the reference point selected in which the bounding is performed. The point may overlap as in the case of Fig. 3.

The processing time using the equation projection method is averaged at 4.6567 msecs per feature set computation using an Intel(R) Core(TM) i3-2348M CPU @ 2.30GHz, 2300 MHz, processor without using any parallel processor features. As a comparison, a similar single hidden layer Artificial Neural Network (Matlab toolbox) using softmax transfer function takes an average 9.8856 msecs per feature set. The projections are acquired after 2000 generations of mutation using 1+4 mutation strategies (mutation stops at exactly 2000 generations). In comparison to the similar work reported [11], a maximum of 10,000,000 evaluations were considered. In [11], the minimal evaluations/generations are 7,000 for breast cancer dataset, 35,000 for diabetes and 25,000 for heart disease dataset. The overall summary of the performance of best solutions is presented in 7.
Figure 1: New 3-dimensional coordinates for projected data on Breast cancer dataset

Figure 2: New 3-dimensional coordinates for projected data on Diabetes disease dataset

Figure 3: New 3-dimensional coordinates for projected data on Heart disease dataset
5. Discussion and Future Work

The proposed method introduced in this research has several desirable qualities. The first quality is that possesses a low computation type of machine learning method. However, it needs to be highlighted that equations generated may be long equations and hence consume more processing as compared to other machine learning. In this research work, only 3-dimensional projections are considered with an allowable maximum of 100 nodes per dimension. Since the chromosome length is fixed, the possibility of bloating (the solution become increasingly long due to inactive nodes) is low. A lower allowable maximum node will limit the complexity of the equations generated.

On the benchmark datasets, solutions generated have an average of 10% of the maximum nodes allowed (100). In the future, the penalty factor may be introduced to biased towards a solution with fewer nodes selected. The best solutions generated on the specified datasets are generally acceptable in terms of computational complexity, and hence, the method is well suited for deployment in an embedded system due to lightweight processing. In terms of portability of the solutions, the equations generated can easily be ported to any machine for deployment regardless of varying programming platform and machine. Unlike KNN classifiers proposed in [8–10], the bounding mechanism provides a crisp boundary to handle soft boundary cases.

The proposed machine learning algorithm is the non-‘black box’ (i.e. white box approach) approach to learning. This provides high interpretability for improving the ML algorithm. The weights to the input feature in the respective equations can be further optimised using gradient approach for fine-tuning.

Further research can be carried out to explore higher dimensions in data projections. From the perspective of complexity, the higher the dimensions applied, the longer the chromosome length and hence the more computation required for classification. Essentially, there is no limit to the number of dimensions that can be applied and hence, Hyperdimensional projection can be considered to explore highly complex datasets. However, projection to higher dimensions than the original dataset dimensions may be counter-productive due to the curse of ‘dimensionality’. Hence, further work may include projections into dimensions 2 to N in which N is the feature dimension of the original feature set.

Concerning classification mechanism, various bounding mechanism such as multi-points reference and hyperplanes may even be considered. 'No free lunch theorem' dictates that there is no single best algorithm for all ML problem. Hence, there exist plenty of exploration on the function sets in the pool that can be included for solution generation. Theoretically, any function can be generated if the pool of functions given is sufficient.

6. Conclusion

The proposed method is justified from the results of the classification as shown in Section 4. In summary, these are the contributions from this research:

1. The projection functions incorporate three types of inputs (node sequence input, feature inputs and constants). The constants are to enable the near boundary to reach the desired label correctly. The probability of receiving either of the inputs in each category is approximately equal.
2. A projected coordinate system yields competitive results as compared to other methods. However, more work needs to be implemented to ensure that the projection does not overfit to training sets. Results for the validation and testing are based on missing dataset values.

3. The research conducted present a bounding mechanism for classification as compared to existing methods of using K-NN [8], [9], [10] mechanism and 1-dimensional piecewise function [11], [12]. The reference point was also generated from which the classification boundaries were formed.

4. A mix mutation mechanism (coarse and fine mutation) is introduced. In coarse mutation (approximately 90% of chromosome segments were mutated in which the segments are selected based on uniform distributed generation integers. In fine mutation mode, only one segment is mutated in each generation.

References
[1] Miller J F, Thomson P and Fogarty T 1997 Designing electronic circuits using evolutionary algorithms. arithmetic circuits: A case study
[2] Miller J F 1999 An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2 (Morgan Kaufmann Publishers Inc.) pp 1135-1142
[3] Harding S, Miller J F and Banzhaf W 2010 Genetic Programming and Evolvable Machines 11 397-439
[4] Walker J A and Miller J F 2004 Evolution and acquisition of modules in cartesian genetic programming European Conference on Genetic Programming (Springer) pp 187-197
[5] Khan M M, Ahmad A M, Khan G M and Miller J F 2013 Neurocomputing 121 274–289
[6] Wang X I, Zhang L y, Dong C w and Rui X p 2012 A multi-dimensional visualization method combining mds and svm Natural Computation (ICNC). 2012 Eighth International Conference on (IEEE) pp 430–439
[7] Dzidolik ait 

et al. 1994
[8] Janghel R, Shukla A, Tiwari R and Kala R 2010 Intelligent decision support system for breast cancer International Conference in Swarm Intelligence (Springer) pp 351–358
[9] Liu H, Zhang R, Luan F, Yao X, Liu M, Hu Z and Fan B T 2003 Journal of Chemical Information and Computer Sciences 43 900–907
[10] Abbass H A 2002 Artificial intelligence in Medicine 25 265–281
[11] Guo P F, Bhattacharya P and Kharma N 2010 Automated synthesis of feature functions for pattern detection Electrical and Computer Engineering (CICEE), 2010 23rd Canadian Conference on (IEEE) pp 1–4
[12] Hao P Y, Tsai L B and Lin M S 2008 A new support vector classification algorithm with parametric-margin model Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on (IEEE) pp 420–425
[13] Fayed H A, Hashem S R and Aitaya A F 2007 Pattern Recognition 40 1498–1509
[14] Oltean M and Groşan C 2003 Solving classification problems using inx form genetic programming International Symposium on Intelligent Data Analysis (Springer) pp 242–253
[15] Chang K y and Ghosh J 1998 Principal curve classifier-a nonlinear approach to pattern classification Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on vol 1 (IEEE) pp 695–700