An Evolutionary Approach to Feature Selection and Network Optimisation

Justin Topfer

Research School of Computer Science, ANU
u4842542@anu.edu.au

Abstract: Given a data set and a classification problem, researchers are faced with the problem of identifying the useful features in that data set and then optimizing the hyper-parameters of their classifier. By building on the work of Irani *et al.*, we show that a Support Vector Machine search algorithm can quickly identify key features in a data set, and using these features, a similar optimization algorithm can then be used to tune a three hidden layer, Artificial Neural Network classifier [1]. This method allows for a quick classification solutions to be found, and can easily be applied to new and diverse problems. This method was applied to data from the study *Towards Effective Music Therapy For Mental Health Care Using Machine Learning* [2] and we show that feature selection with the search algorithm improves classification with an un-optimized classifier from 83.3% (with loss of 0.036) to 97.2% (loss of 0.001). After optimizing the network on accuracy we reported classification accuracies of 99.31% with loss 0.000043 (using stratified 5-fold cross validation) and after optimizing for loss we reported loss values of 0.000014 with accuracy of 97.9%. These classification results were less accurate than in the original paper, however only five features were used from the data set.

Keywords: Evolutionary Algorithm, Genetic Algorithm, Feature Selection, Artificial Neural Networks, Support Vector Machine, Bio-Inspired Computing

1 Introduction

Given a data set and a classification problem, researchers are faced with the task of identifying the useful features in that data set and then optimizing the hyper-parameters of their classifier. An efficient method of reducing a data set is attractive, because identifying high-value features can reduce data collection costs, training times and as Rahman *et al.* found, can improve classification results versus training on the entire data set [3]. Optimizing a classifier can also be an expensive task that requires significant domain knowledge, so a search and optimization method that could be easily applied to different problems would be very useful. Genetic Algorithms (GA's) have found much use in search and optimization problems, so they were an obvious method to choose.

A GA is a metaheuristic and is a subset of a larger class of evolutionary algorithms. GA's have found much use in search and optimization problems, especially when the search space is not well defined or understood [4]. In this paper, we describe a system of two GA's for addressing the problem of data selection and classifier optimization.

The first GA is a search algorithm using a Support Vector Machine (SVM) to evaluate search results. An SVM is a classifier that attempts to construct a hyperplane between classes so as to maximize the distance between the classes and the plane [5]. An SVM classifier was chosen because it trains much faster than other classifiers like Artificial Neural Networks, and this was an attract property given that the algorithm would have to search a large data set. A pre-processed data set was the input into this algorithm, and a subset of high-value features was identified for further classification.

The second GA is used to optimize the hyper-parameters of a three hidden layer Artificial Neural Network (ANN). An ANN is a computing methodology that is inspired by the structure of biological brains and has been used widely in classification, image recognition, regression and other prediction tasks. One of the strengths of ANN's is their many tune-able hyper-parameters. Given sufficient tweaking of these parameters, ANN's can find excellent solutions to problems. However, finding the best settings is a time consuming process, so this optimization algorithm aimed to optimize the hyper-parameters of the ANN automatically.

We used data presented by Rahman *et al.* in their paper *Towards Effective Music Therapy for Mental Health Care Using Machine Learning Tools: Human Effective Reasoning and Music Genres.* Rahman *et al.* collected physiological signals from 24 participants while they listened to music which was classed as stressful or calm. We analysed 210 features extracted from the 14 channels of an EEG headset worn by participants during the experiment [6].

2 Data

A 144 x 212 database was provided. Of the 212 features, 210 were extracted from the 14 channels of an EEG. For each channel, 15 different statistical summaries were provided such as the mean, skewness and the Hurst exponent. The last two features were the class labels (calm or stressful, labeled 1 or 2) and the participant number. The classes were

balanced, with 72 records from each class and the participant numbers were also balanced, with 6 records per participant.

2.1 Data Preprocessing

Many features of the database were on very different scales, so some data standardisation process was desirable. Initially data was put on a 0-1 scale, but this was found to reduce naive SVM classification accuracy from 81% to 54.5%. We hypothesise this standard exacerbated the influence of outliers. A -1 to 1 scale had a similar effect, so a z-score scale was used (i.e standard deviations from the mean).

Many records contained significant outliers. 771 data points were found to be more than 3 standard deviations from their respective feature means and 34 of these points were more than 8 standard deviations away. Simply removing records with the more extreme values did improve accuracy, however imputing the mean feature value for each outlier was deemed preferable because it resulted in a similar accuracy without changing the balance of the database or removing valid information.

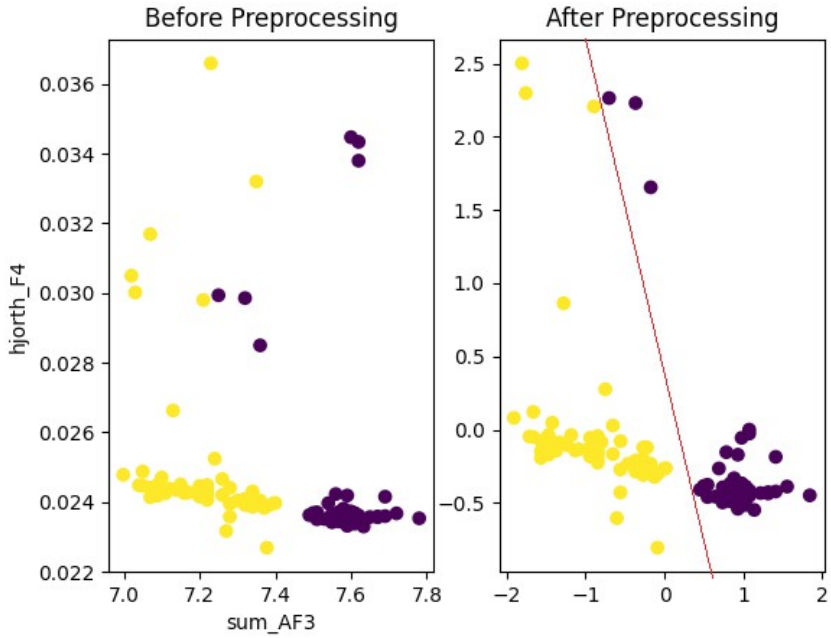

Fig 1. Classes are easily separated after preprocessing for some high-value features

Some features in the data set were found to give nearly 100% classification accuracies by themselves. To allow for optimization on accuracy these features had to be removed from the data set, otherwise the GA would stall at perfect or near perfect accuracy. In total seventeen features were removed (see appendix one for a list). These features were identified by running the SVM search algorithm with a max feature count of 1 (I.e searching for the best feature) and removing any single feature that resulted in accuracy above 95%. After removing these seventeen features we were left with a 144x193 data base for classification.

2.2 Data Splitting

Stratified 5-fold cross validation was used to evaluate classification both in the SVM search algorithm and the ANN optimization algorithm. This method involves splitting the data set into five 'folds', and then training the data set on four folds while holding one separate for testing. This process is then repeated five times, so that every fold can be tested upon and the average accuracy or loss of the five tests is then reported. The folds were also stratified - meaning that the classes were balanced in each fold to avoid bias from class representation.

Stratified 5-fold cross validation was selected because the data base is small – it has only 144 records. Cross validation has the advantage of training on, and testing with, all of the data so it is ideal for smaller data sets.

3 Methodology

The method comprised of two steps. Firstly, an SVM feature selection algorithm searched through the data set to identify a selection of high-value features. Then, using these high-value features, an optimization algorithm would

optimize the hyper-parameters of a three hidden layer neural network to find the best parameters to maximize classification accuracy or to minimize classification loss.

3.1 SVM Search Algorithm

This algorithm is a genetic algorithm that takes in a large data set and outputs a selection of features which it has found to give the highest accuracy. The search algorithm has the following steps:

1. Creation of binary encoded chromosomes
2. Evaluation of chromosomes
3. Selection of the best chromosomes
4. Two-point crossover
5. Mutation of genes
6. Trimming

Steps 2-5 would be repeated for a certain number of epochs, or until the algorithm had converged.

3. 2 Creation

A population of 150 binary encoded chromosomes were created. A binary encoded chromosome is a string of 0s and 1s of the same length as the feature space to be explored (193 in this case), where a 1 at position 10 would indicate that the 10^{th} feature should be used for evaluation. The chromosomes would have a set number of randomly selected features enabled. Typically, one feature was enabled at creation because the algorithm tended to be expansionary – it would prefer to have more features enabled than less.

150 was found to be a good population size because it would ensure convergence in under thirty generations. This number is going to be dependent upon the size of the data base being explored. Given how fast the search process was for this data set, it did not impact results very much.

3.3 Evaluation using Linear SVM

After creation, the chromosomes would be evaluated using a linear SVM. An SVM attempts to find a hyperplane, or decision boundary that best separates the classes. The features that the chromosomes carried would be passed to the classifier where stratified 5-fold cross validation would be used to train and evaluate the accuracy of the classifier. The accuracy for a given chromosome was then used as it's measure of 'fitness', which when ranked against it's peers would indicate if it should survive or be replaced.

The parameters of the SVM were determined by performing a grid search using Radial Basis Function (RBF) and Linear kernels. The Linear kernel has a single hyper-parameter, C. The C parameter trades off misclassifcation of training examples with simplicity of the decision plane, so a high C can lead to over fitting while a low C results in poor accuracy. The RBF kernel has two hyper-parameters, C and gamma. Gamma defines how much influence a single training example has [7].

To determine which kernel and what hyper-parameter settings to use, a grid search was conducted using the two kernels, C values of 0.1, 1, 10, 100 and 1000 and gamma values of 1, 0.1, 0.01, 0.001 and 0.0001. This grid search was conducted on 5 random features from the data set (so as to be similar to the search function) and using a 70/30 training and testing data split and optimizing for accuracy. A linear kernel with a C value of 0.1 was found to be optimal and resulted in a testing accuracy of 86% on this data subset. The confusion matrix for the testing data set is shown below:

	Predicted = 0	Predicted = 1
Actual = 0	22	3
Actual = 1	3	16

Fig 2. Confusion matrix for optimum SVM settings

3.4 Selection

Each chromosome was now ranked depending upon its classification accuracy and the best 45% of the chromosomes were kept. 45% was chosen as the cutoff by examining this parameters effect upon diversity, where diversity was simply the number of unique chromosomes in each generation. To test this, seven trials were run using the search algorithm with cutoff values ranging from 0.25 to 0.5 and searching the feature space with a population of 150 hundred chromosomes over thirty generations. The average diversity within a population for each trial was recorded below:

Cutoff Value	Average Population Diversity
0.2	82.5
0.25	88.4
0.3	79.2
0.35	84.3
0.4	90.2
0.45	100.7
0.5	97.6

Fig 3. Testing selection cutoff value

A cutoff of 45% resulted in the greatest average diversity, so this value was used for the model. Logically, it makes sense that a larger cutoff will result in higher diversity due to how we have implemented crossover. The fittest chromosome is combined with all other chromosomes first, before the next fittest is combined and so on. So a larger cutoff value means we widen the accuracy difference between fittest chromosome and the chromosome it is combined with last.

3.5 Crossover

The surviving chromosomes were then combined using two-point crossover. Two cuts were randomly made somewhere along a pair of parent chromosomes and the middle section of each was swapped, to create two new chromosomes.

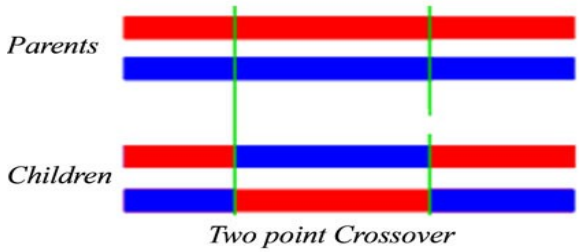

Fig 4. Two Point Crossover [8]

Spears and Anand's found, that given a population size over twenty, GA's with crossover outperform those without [9]. We found a population size of 100 to be ideal, so were confident that crossover would improve the model.

The order of parent chromosomes was not random. The chromosome with the highest fitness was combined with every subsequent chromosome first, before the second fittest chromosome was combined with the others and so forth. Crossover continued until a certain number of chromosomes had been made.

3.6 Mutation

After crossover, 1-5% of genes in the chromosomes were mutated (changed from a 1 to a 0 and vice versa). Other researchers have suggested a starting mutation rate of 0.5-1%, however we found for this application a higher mutation rate was better and inspired greater searching of the feature space [10]. The feature selection algorithm tends to find a good selection of features and then latch onto them. So a higher mutation rate can input more diversity into the system.

3.7 Trimming

Because classifiers tend to perform better with more information, chromosomes had a tendency to simply get larger without fully exploring the data base for a given number of features. To combat this, if a chromosome had more than five features enabled, then features would be randomly selected and turned off until the total number of features was equal to five.

3.8 Optimization of the NN classifier

Once the SVM search algorithm has converged and selected a subset of high-value features, the optimization algorithm can use these features to find ideal hyper-parameters for tuning a classifier. We chose to use a three hidden layer, ANN as our classifier. This model was chosen because it has several tune-able hyper-parameter's for optimization. The parameters we chose to tune were the number of neurons in the first and second hidden layers, the learning rate and the number of epochs to train for.

The optimization algorithm is a GA that takes in a data set as input and outputs the best combination of hyper-parameters that it found within a certain time frame. It can optimize on loss or accuracy and has the following steps:

1 – Creation of custom encoded chromosomes
2 – Evaluation using an ANN
3 – Selection
4 – Custom crossover
5 – Mutation

3.9 Creation

Custom encoding was used for chromosome representation. Each chromosome was a list of four numbers which described the four tune-able parameters: the number of units in the first hidden layer, the number of units in the second hidden layer, the learning rate and the number of epochs to train over. The values for these chromosomes were initially chosen randomly from within certain limits, and kept within these limits while the algorithm progressed. This was done to ensure that the values were always valid and sensible.

The number of units within each hidden layer was limited to be between one and fourty. This range gave significant scope for unexpected findings while still being reasonable. Learning rate was limited to between 0.001 and 0.3 for similar reason. Number of epochs was limited to 150 due to processing requirements, however this limit was rarely approached by the algorithm.

Initially the population sizes were kept to a smaller number of around twenty and the algorithm ran over thirty generations. However, although there are only four tune-able parameters there is a large number of possible combinations of these parameters. So a larger population size of 100 chromosomes evolved over 30 generations was found to give better results.

3.10 Evaluation using an ANN

In the same manner as the SVM search algorithm, an ANN was used to evaluate the chromosomes for the optimizer. A three hidden layer ANN was chosen because this schema gives an extra tune-able parameter over a two layer ANN and it remains a realistic design for the relatively simple nature of the classification problem. It would be possible to give the optimization algorithm complete control over the structure of the ANN, however that was not needed for this problem.

Rectified linear units were used as the activation function on the internal hidden layers. This function was used because it does not have the vanishing gradient effects of the sigmoid function, and is the default activation function for multilayer perceptrons [11].

Adaptive Momentum Estimation (Adam) was used as the optimizer. Adam combines ideas from momentum and gradient descent optimizers. Adam makes use of an exponentially decaying average of past gradients and has an adaptive learning rate for each feature [12]. This means it can escape local minima while still converging to an answer quickly.

Cross entropy was used to calculate loss. Cross entropy is the measure of the distance between two probability distributions and is widely used when optimizing classification models [13].

Stratified 5-fold cross validation was used to train and evaluate the ANN. The average accuracy or average loss from this validation was then used to rank the chromosomes.

3.11 Selection

Similar to the search algorithm, the best 45% of chromosomes were selected for crossover. As found with the search algorithm, the optimization algorithm crossover technique combines the best performing chromosome with each subsequent chromosome, so a wider selection was expected to be better.

3.12 Custom Crossover

Two point crossover could not be used because the chromosomes are not binary encoded. So, for crossover in this algorithm we chose to simply take the average value of the two chromosomes. Although simplistic, this approach resulted in acceptable diversity and convergence.

In the same manner as the search algorithm, the highest fitness was combined with every subsequent chromosome first, before the second fittest and so on.

3.13 Mutation

Similar to the search algorithm, we found that a higher mutation rate led to a better result. Because there are only four genes in each chromosome, we found that a higher mutation rate of 15% was useful.

When a gene was mutated, a new random value was drawn for it from within the possible limits.

Results

The SVM search algorithm was found to rapidly converge on a solution with accuracy of 95.1% using the features 'iqr_F4', 'sum_T8', 'hjorth_F8' and 'hurst_T7'. Although a maximum of five features was possible, the algorithm only found four which combined with the very quick convergence suggests that the search area may be too constrained.

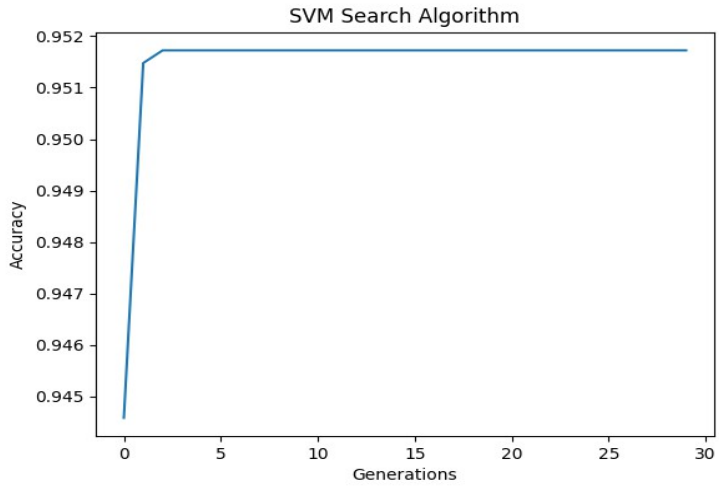

Fig 5. Search algorithm performance

However when examining the population diversity, we see that for every generation of 150 chromosomes there are approximately 95 unique chromosomes, so perhaps the uniqueness of chromosomes within a generation is not the problem but rather the search is limited to some subset of features. We also see diversity begin to oscillate after 20 generations, which indicates the algorithm has converged towards some solution.

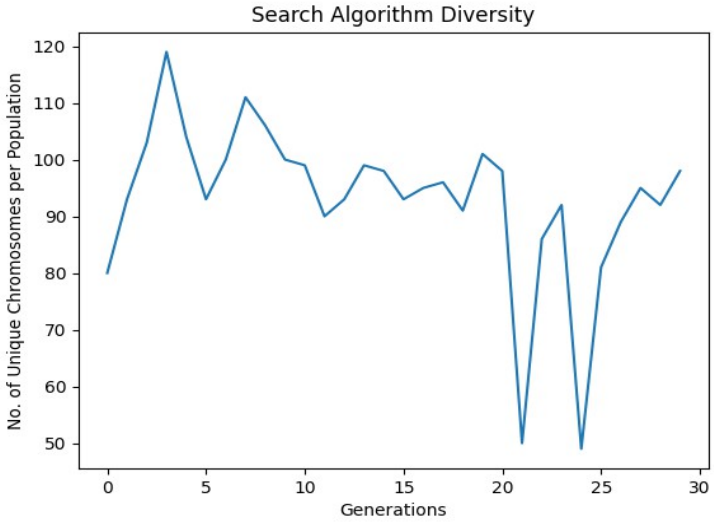

Fig 6. Search algorithms diversity

Feature selection with the search algorithm also increased classification with an un-optimized neural network from 83.3% with a loss of 0.036 to 97.2% and a loss of 0.001.

After these features were selected by the search algorithm they were passed to the optimization algorithm for classification. The optimization algorithm could optimize on classification accuracy or on classification loss.

Prior to optimization, the un-optimized three layer neural network had 10 units in each hidden layer, a learning rate of 0.1 and a training time of 150 epochs and recorded an accuracy of 97.2% with a loss of 0.036 when using stratified 5-fold cross validation. After optimization on accuracy, a maximum accuracy of 99.31% was reached with loss of 0.000043. This represents a significant improvement in accuracy and loss over the un-optimized network. The best results were achieved with a 15 units in the first hidden layer, 27 in the second hidden layer, a learning rate of 0.09 and a training time of 113 epochs.

However similar to the search algorithm, the optimization algorithm rapidly converged on a solution and then did not make any improvement. It's unclear if this is a good characteristic or not, however one would expect more oscillation.

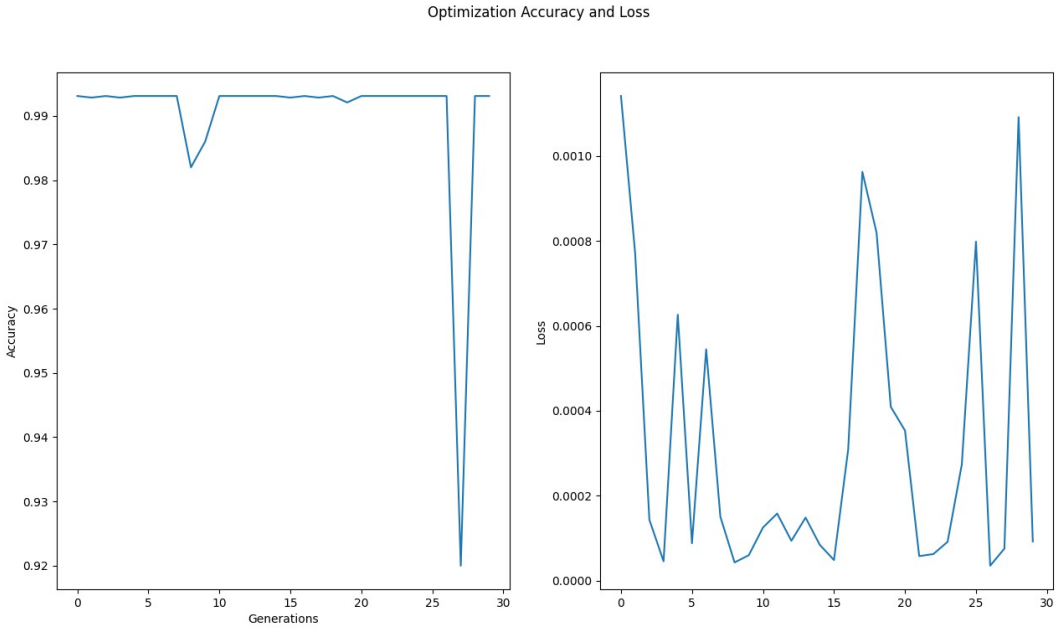

Fig 7. Accuracy and loss when optimizing for accuracy

When optimizing on loss, a minimum loss of 0.000014 was recorded with an accuracy of 97.9%. Interestingly, when optimizing on loss the algorithm seems to traverse the error space more smoothly, although it does take much longer to converge. Due to processing constraints the algorithm could only be ran for thirty generations, but it would be interesting to see the results from a longer time frame.

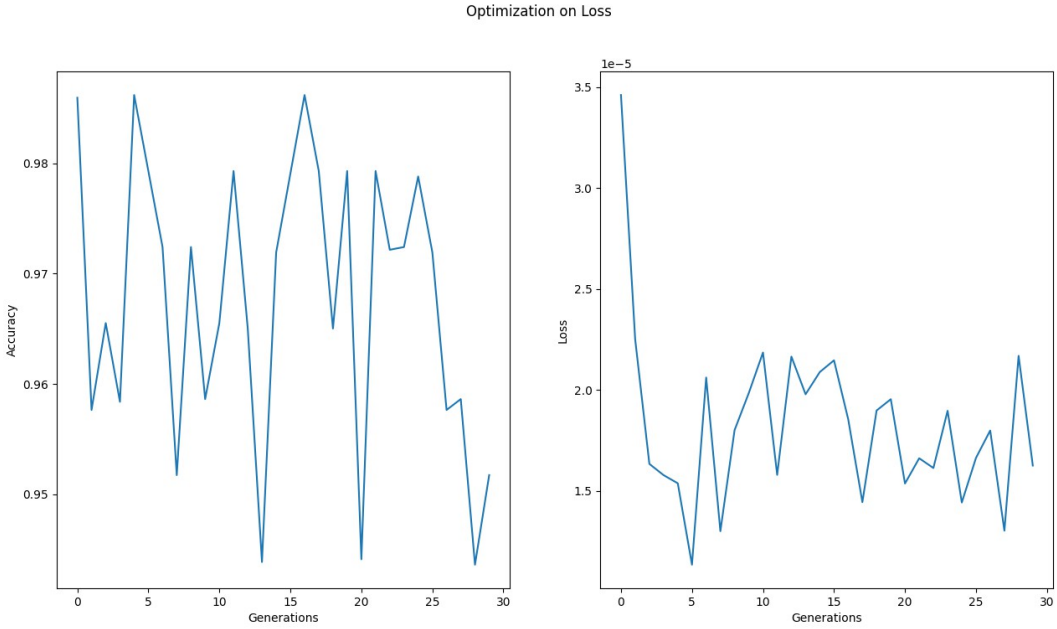

Fig 8. Accuracy and Loss when optimizing on loss

Conclusion

In this paper, we showed how search and optimization algorithms could be used to reduce a data set and optimize an ANN classifier. We described the structure of the genetic algorithms, illustrated how the algorithms could explore a feature space to identify possible solutions and identified issues with the proposed models. We showed that the search algorithm reduced the data set significantly and improved accuracy on un-optimized classifiers, and the optimization algorithm resulted in significantly improved classification accuracy and loss.

In future work, we would like to explore better methods for describing the composition of populations of chromosomes. Simply reporting accuracy for each generation does not give sufficient information on what is happening within the population to identify and solve issues. Better reporting would allow us to develop further methods for increasing population diversity and would result in more robust searching of the error space.

We were also unsatisfied with the method of crossover used in the optimization algorithm. Although easy to understand and implement, we feel that crossover by simply using the average value of the two chromosomes may lead to premature convergence and a stagnant system. Experimentation with other methods would be ideal. In a similar vein, it would be worth experimenting with having other ANN parameters, such as the activation functions and optimization functions, as part of the optimization algorithm. Discerning a way to implement crossover with these functions is the main issue here.

Appendix One

Removed features - sum_AF3, hjorth_P7, sum_P7, rms_F8, mean_P8, sum_P8, sum_T7, sum_F8, mean_F8, sum_O1, sum_FC6, sum_O2, mean_FC5, sum_FC5, sum_AF4, rms_F7, sum_F7

References

1. Irani R., Nasrollah K., Dhall A., Moeslund T., Gedeon T. *Thermal Super-Pixels for Bimodal Stress Recognition,* pp1
2. Rahman J, Gedeon T, Caldwell S, Jones R, J*in Z, Towards Effective Music Therapy for Mental Health Care Using Machine Learning Tools: Human Affective Reasoning and Music Genres,* JAISCR, 2021, Vol. 11, No. 1, pp. 5-20
3. Rahman J, Gedeon T, Caldwell S, Jones R, Jin Z, *Towards Effective Music Therapy for Mental Health Care Using Machine Learning Tools: Human Affective Reasoning and Music Genres,* JAISCR, 2021, Vol. 11, No. 1, pp. 18
4.
5. Mitchell, Melanie, *An Introduction to Genetic Algorithms.* Cambridge, MA: MIT Press (1996). pp. 2-5
6. Cortez C, Vapnik V, *Support-Vector Networks,* Machine Learning, 20, 273-297 (1995) pp. 273
7. Rahman J, Gedeon T, Caldwell S, Jones R, Jin Z, *Towards Effective Music Therapy for Mental Health Care Using Machine Learning Tools: Human Affective Reasoning and Music Genres,* JAISCR, 2021, Vol. 11, No. 1, pp. 7
8. https://scikit-learn.org/stable/modules/svm.html#svm-kernels, accessed 31/05/2021
9. Spears W., Anand V., *A Study of Crossover Operators in Genetic Programming,* ISMIS 1991: Methodologies for Intelligent Systems, pp409-418
10. Askali M, Azouaoui A, Nouh S, Belkasmi M, *On the Computing of the Minimum Distance of Linear Block Codes by Heuristic Methods*, International Journal of Communications Network and System Sciences, Vol. 5, No. 11, 2012, pp. 2
11. Gupta D., Ghafir S., *An Overview of Methods, Maintaining Diversity in Genetic Algorithms,* International Journal of Emerging Technology and Advanced Engineering, Vol 2, Issue 5, May 2012
12. Brownlee J, *How to Fix the Vanishing Gradient Problem Using the ReLU,* https://machinelearningmastery.com
13. https://machinelearningjourney.com/index.php/2021/01/09/adam-optimizer/
14. Brownlee J, *An Introduction to Cross Entropy for Machine Learning,* https://machinelearningmastery.com