FEATURE SELECTION AND COMPRESSION FOR STRESS CLASSIFICATION: EVOLUTIONARY ALGORITHMS FOR NEURAL NETWORK COMPRESSION

David Wade
u6942357@anu.edu.au

Abstract. Neural network classifiers often use large datasets and layers with many units. Training a neural network with many parameters can be computationally expensive and can negatively affect model performance. This paper demonstrates how evolutionary algorithms can be used to select useful features for neural network classifiers, thereby discarding irrelevant input features and producing a classifier that is less computationally expensive to deploy, using a stress classification network as an example.

1 Introduction

Human brain wave readings from thermal electroencephalographic (EEG) sensors can be used to predict whether a person is calm or stressed using artificial neural network models. Using all the thermal-eeg headset sensor readings (and their statistical parameters) to make classifications requires a large input vector, which can negatively affect model performance and lead to longer computation times[1]. This report aims to describe an artificial neural network used to predict human stress levels accurately using a sequential feed forward network, and an extended set of compression techniques including a genetic-based algorithm for feature selection. This is an example of how a genetic algorithm can be used to produce optimal model parameters without having to compromise computational time.

1.1 Background

This report is an *extension* of my previous model and technique to preserve functionality in a neural network compressed based on the hidden units' distinctiveness. This itself is based mostly on T.D. Gedeon and D. Harris' approach described in *Progressive Image Compression*[2]. The technique is developed on a model with the same thermal-eeg dataset and the same classification goal. I compressed the hidden layer of a three-layer feed forward network by constructing a set of vectors with the number of training patterns as its size, each vector includes values equal to the activation of one hidden neuron for each training pattern. Each vector corresponds to one unique hidden neuron, and each hidden neuron's activations are encoded into one vector. My previous model uses a K-means clustering algorithm to describe each hidden unit's "distinctiveness" from each other based on the calculation of the angle between them in the input pattern space.

My extended technique includes the distinctiveness-based compression technique, but also aims to reduce the number of features required to train an accurate model. Achieving this by brute-force (i.e., by training a network on every possible set of input features) is not computationally feasible.

1.2 Proposed technique

My extended model uses an evolutionary or genetic based algorithm to select the optimal features to be used as input in the calm/stress classification problem. Each model is encoded in a "chromosome" representing which features are included as input. A set or *population* of chromosomes are evaluated based upon their model's performance on classifying data in a test set. This produces a testing accuracy or *fitness* which determines how likely a chromosome is to *reproduce*, i.e. have features that determine the chromosomes of a more optimised population, or a *generation*. Each chromosome is also affected by *mutation*, or a small, random variation in its binary encoding. After a set number of generations, with a set number of chromosomes in each population, I aim to produce a population of chromosomes that converge to a model with an optimal set of input features.

1.3 Dataset

This paper reports on the *thermal_eeg* (version 1) dataset. Each data pattern is a summary of time-series data recording a participant's electrical brain activity while watching either a calming or stress-inducing video. The features (columns) in the dataset are statistical measures of the *mean, minimum, maximum, standard deviation, variance, interquartile range, skewness, root mean square, summation, hjorth parameter, hurst exponent, mean of first signal difference, mean of second signal difference, approximate entropy and fuzzy entropy* of each of the headset's fourteen channels' signal readings over each time series.

2 Method

2.1 Evolutionary algorithms for feature selection

Evolutionary algorithms are optimisation algorithms that mimic biological evolution by evaluating a set of solutions based on their *fitness*, or ability to solve a problem effectively. Solutions that meet a fitness criterion have a higher chance of being selected as *parents* that determine the characteristics of the next generation of potential chromosomes. Each new chromosome in a successive generation are randomly altered (mutated) by a small factor, to encourage a more diverse set of solutions in each generation. After several hundred generations, chromosomes that represent poor solutions are eliminated due to having a low fitness and a low chance of reproducing. Chromosomes that represent strong solutions ideally converge, and each chromosome in a final generation should encode a set of features that produce network model with high accuracy[3]. A flowchart representing this process is provided below.

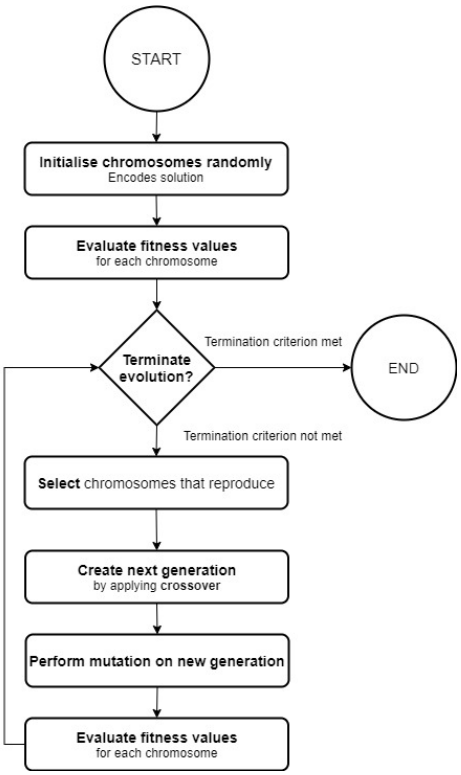

Fig 2.1 logic flow of evolutionary algorithm

2.1.1 Initialisation

Chromosomes are initialised as a vector of random bits of length 210. Each element represents the i^{th} input feature of the dataset. Chromosomes values of 1 indicates an input feature that is included in the model, and values of 0 indicate input features that are not included in the model. In my program, 200 chromosomes are included in every generation is stored in a matrix with 200 rows and 210 columns.

2.1.2 Evaluation

Chromosomes are evaluated upon their performance in a feed-forward classifier neural network with one hidden layer. One network is created for each chromosome and has an input size equal to the number of 1 bits in the chromosome. The hidden layer has a size equal to 66% of the number of input features (rounded down to the closest integer). Each network has two output units for each prediction class (calm, stress).

The network uses a Cross Entropy Loss function to evaluate how close predictions are to the real classes that input patterns represent. A stochastic gradient descent optimiser is used to alter weights and biases to an approximate solution with a learning rate of 0.3. Units in the hidden layer use the Softplus activation function (with a limit of 1 million).

Each network is trained on a dataset 66% the size of the original *thermal_eeg* dataset. Since the dataset is quite small, no batches are necessary during training. To save computation time, each network trains on no more than four epochs. Due to the small size of the dataset, networks generally overfit the training data at around epoch two, and the state of the network is saved as an optimal model at whichever epoch trains a network with the lowest testing loss. The testing *accuracy* of each chromosome's network is saved as the fitness value of that chromosome.

2.1.3 Crossover

Crossover is analogous to biological reproduction, where the fittest individuals in a population determine the traits of the following generation. The chance of a chromosome reproducing are determined by their individual fitness as a percentage of the summation of all chromosomes' fitnesses. Each chromosome is sorted into a probability distribution array of size 200 (the number of chromosomes in each generation), where each element is the index of a chromosome in the parent generation. In my program, a bias is applied such that fitter chromosomes are more highly represented in the probability distribution than their fitness proportion would determine. Parent chromosomes are selected randomly from the distribution. The number of elements corresponding to any chromosome is proportional to their fitness, so that they have a higher chance of being selected as parents to the next generation.

Crossover is applied by creating an offspring chromosome where each gene is randomly selected from parent A or parent B, randomly selected from the chromosome probability distribution. This is a process called *uniform crossover*. This process is repeated for each new chromosome, resulting in a new population that combine the better traits of the previous generation.

2.1.4 Mutation

Mutation introduces further diversity to each generation of new offspring chromosomes. Mutation occurs by inverting bits in a chromosome such that a previously activated feature is disabled, and a previously activated feature is disabled. The number of bits mutated in a chromosome is determined by the mutation probability (set to 5%).

2.1.5 Termination

In my program, new generations keep being produced until 400 generations have passed. Over the whole evolutionary process, any chromosome that produces a model with a higher testing accuracy than the current best model is selected as the best model. Once the evolutionary algorithm terminates, the best performing model and its testing accuracy is reported in the terminal output. The original distinctiveness-based compression technique is then applied to the hidden layer, reducing the hidden layer to 70% of its original size.

3 Results

As a control measure, models produced by the evolutionary algorithm are compared to a networks with no compression. A control network is created with 210 input size, for each feature in the original dataset and a hidden size of 15 0 units. All other hyperparameters of the control network are identical to those used in the evolutionary algorithm. A total of eighty thousand (equal to the product of the number of generations and chromosomes) control models were constructed, and the testing accuracy of the best performing model in each 200 iterations (analogous to generations) were recorded.

In comparison to the control models, the best models in each generation's chromosomal models had slightly smaller testing accuracies. The mean testing accuracy for the control models was 77.08%. The mean testing accuracy for the models with input features selected by evolutionary algorithms was 73.01%. Further statistical parameters are summarised in figure 3.1 and the distribution of test accuracies are summarised in figure 3.2 and 3.3. These results are consistent with the program being run multiple times, within a few percentages.

Test accuracy	ANN Control	Evolutionary algorithm
Mean (%)	77.40	72.89
Min (%)	54.0	52.0
Max (%)	90.0	90.0
Standard deviation	0.0671	0.0754

Fig 3.1 Stress classification network results

3.1 Evaluation

As discussed in my previous compression report, accurate models appear to be highly dependent on the randomly initialised weights and biases of a network. For this reason, multiple models trained on the thermal_eeg dataset with identical hyperparameters rarely converge to an optimal solution. Upwards of 10 models often need to be trained before a model with a testing accuracy over 70% is produced. Evolutionary algorithms are designed to find an approximation of the optimal solution, not the optimal solution itself. My program uses 400 generations to approximate a solution to the stress classification problem due to the physical limitations of the machine the program was tested on. Although the program produces positive results, with some chromosomes producing a high-accuracy model, it is likely many good solutions are not detected in the selection process due to the random initialisation of weights and biases. A more optimal solution might be found if the program was run using many more generations. The parameter controlling the number of generations in my program is named *num_generations* and can be changed to any value.

Chromosomes produced by the final generation of the evolutionary algorithm have an average of 107 features activated in the model they encode. This demonstrates that features selected using an evolutionary algorithm can produce a predictive stress model with equal test accuracy to a model with no feature selection. After distinctiveness-based compression is applied to the hidden layer, the models usually maintains its prior test accuracy, however occasionally

loses accuracy by around 10%. The result of this combined compression technique on the highest performing model selected from the evolutionary cycle is a model with an input size approximately half that of a model with no compression and 210 input features. After applying distinctiveness-based compression, the hidden layer is reduced to 70% of its original size. The resulting compressed model has a testing accuracy usually above 90%, demonstrating that a stress-prediction neural network can be significantly reduced in size using evolutionary algorithms and distinctiveness-based compression, while maintaining a high degree of accuracy.

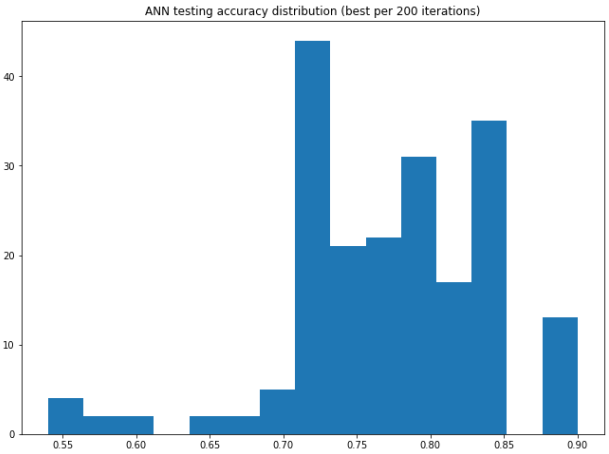

Fig 3.2 Testing accuracy distribution on ANN models

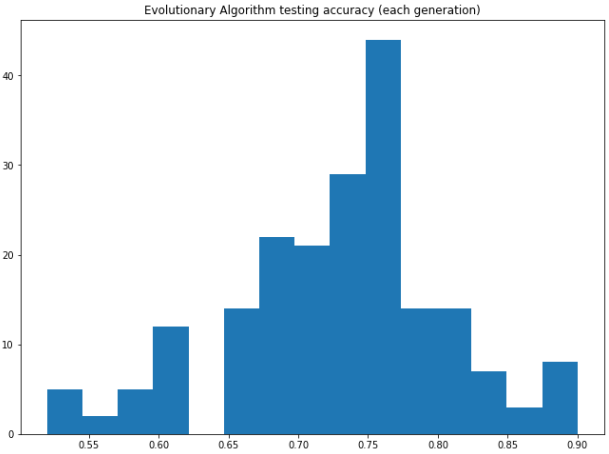

Fig 3.3 Testing accuracy distribution on evolutionary models

4 Conclusion

My program demonstrates that evolutionary algorithms can be used to select features in a dataset that are useful for classification tasks. Evolutionary feature selection and distinctiveness-based compression can be used to reduce the size of a stress-classification network and achieve test accuracy that is at least as good as a network with no compression, resulting in a smaller network that is less computationally expensive to use.

5 References

1. Gedeon, T.D., "Evolutionary Computation: Genetic Algorithms for Feature Selection", Research School of Computer Science, ANU.
2. Gedeon, T.D., & Harris, D, "Progressive Image Compression", School of Computer Science and Engineering, UNSW, Centre for Neural Networks, King's College, London.
3. Gedeon, T.D., & Qin, Zhenyue, "Evolutionary Computation: Genetic Algorithms with Technical Details", Human Centred Computing Laboratory, ANU.