Evolutionary Hyperparameter Optimisation for Improving Constructive Cascade Network Performance on Alcoholism Classification with EEG Data

Tristan Smith

[1] Research School of Computer Science, Australian National University
u6949592@anu.edu.au

Abstract. This paper investigates the viability of using a genetic algorithm to improve the performance of the casper architecture, a type of constructive cascade network, on the classification of alcoholism based on statistical features extracted from EEG scans. It finds that a genetic algorithm has the potential to find a hyperparameter configuration that yields better accuracy than the default parameters outlined in casper's introduction by Gedeon et. al[1], and those found with manual experimentation. The best performing chromosome found by the method used in this paper yielded an average accuracy of 77.97% and a median accuracy of 78.03% as opposed to the best combination of default and experimental parameters giving an average accuracy of 75.78 and a median accuracy of 77.26, resulting in the potential for a marginal gain in performance in exchange for a significant time investment.

Keywords: EEG, Alcoholism, Machine Learning, Cascade Network, Genetic Algorithm

1 Introduction

Constructive cascade networks, first introduced by Fahlman and Lebiere in the Cascade Correlation (cascor) algorithm[3], are a method that adaptively increases the number of hidden neurons as training progresses, thereby eliminating the need to experimentally determine a good network topology for a given task. Treadgold and Gedeon have shown a more sophisticated approach to the premise of the cascor algorithm titled 'casper', that improves on the limitations of cascor for certain classification problems[1] by using resilient backpropagation, training new neurons with a greater learning rate than existing ones rather than freezing existing weights. Casper uses a set of hyperparameters to control how often new hidden neurons are added during training, the initial relationship between the learning rates of previous and new parameters, and a coefficient controlling the influence of a simulated annealing weight decay process carried out during optimization. In the paper introducing casper, values were provided for parameters controlling the initial learning rates, but two user defined parameters were also given. The aim of this investigation is to explore the effectiveness of using a genetic algorithm to search for values that these hyperparameters may take on which improve casper's performance without the need for extensive experimentation on identifying alcoholic subjects using features extracted from a set of EEG recordings, as compared to those outlined by Gedeon et. al. (and found by user experimentation in the case of the two user controlled values). This investigation also seeks to compare the performance of the improved casper model as compared to a regular feed forward network with a similar number of hidden neurons.

1.1 Motivation

Electroencephalography (EEG) is a process in which an array of electrodes is used to record electrical activity on a subject's scalp. Analysing the recorded electrical activity in the frequency domain can allow deductions to be made regarding the subject's neural activity and cognitive state[6]. Apart from having significant practical use across biology, computing and medicine, EEG data presents a strong characterisation of complex real world data with non-trivial levels of noise and variance, making it a viable way to evaluate the robustness of machine learning techniques.

Evaluating how well casper performs on EEG data with hyperparameters found using an evolutionary search in comparison to both the regular parameters and a regular feed forward neural network can provide an informative perspective on the viability of using methods that reduce the need for trial and error in complex real world problems. It should be explored whether combining casper and evolutionary algorithms can be used to either solve such problems directly or inform the design of a more effective solution, reducing the amount of experimentation required to choose a suitable model.

1.2 Dataset and Preprocessing

This investigation uses a set of features extracted from recordings belonging to a dataset from UCI, which is an EEG dataset from the Neurodynamics Laboratory at the State University of New York. The original dataset used 122 subjects, 77 of whom were diagnosed with alcoholism along with 45 non-alcoholic subjects[2]. A number of trials were conducted for each subject, recording their neural activity for a brief duration in response to a set of 5 different stimuli. A set of 192 features for each example consisting of the mean frequency values taken from a length 256 time series for each frequency band (alpha, beta and gamma) on each of 64 electrodes was used. The casper and feed forward networks

were trained to classify whether or not a feature set belonged to a person with alcoholism. The extracted features dataset contains 11057 trials, of which 7053 were from subjects with alcoholism. The data was normalised using scaling before being used with any models.

2 Method

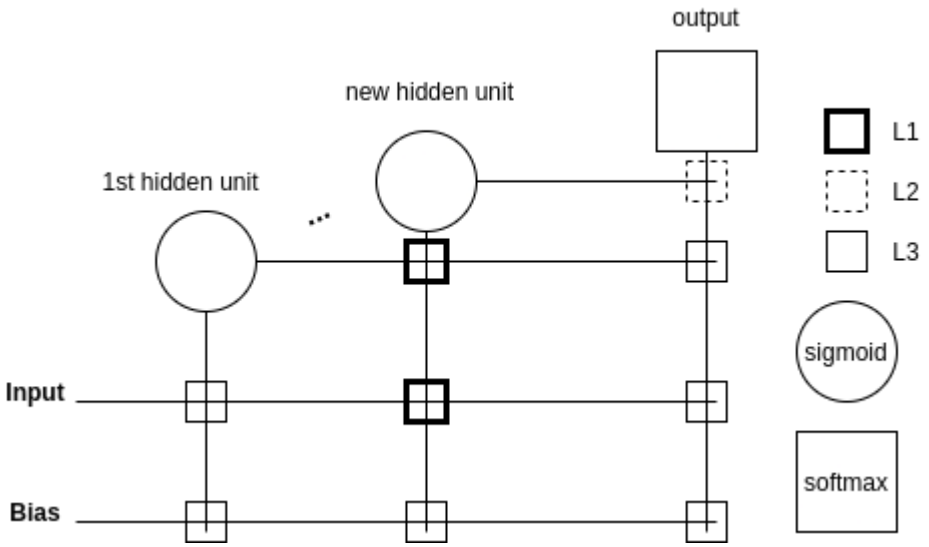
2.1 Casper Network Topology

The topology of networks produced by the casper algorithm is such that each layer has a direct connection to every successive layer, hence each layer contributes directly to the final output, as well as influencing the activation of each proceeding hidden unit. The casper network trained on the EEG features used a sigmoid function for hidden unit activations, as outlined by Treadgold and Gedeon[1], with a cross entropy loss function, hence the output layer uses softmax activation.

2.2 Casper Training Algorithm

The casper algorithm begins with a minimal network consisting of the input and output layer, along with a single hidden neuron. After reducing the training error for a number of epochs given by 15 + P * N, where P is a user defined parameter and N is the number of hidden neurons added thus far. If the residual error has been reduced by at least 1%, a new hidden neuron is added to the network. The network parameters are then split into 3 regions: L1, L2 and L3, where the L1 weights have a comparatively high learning rate, the L3 weights have the lowest learning rate, and L2 has a learning rate a few times greater than L3.

Figure 1. Casper network when a new hidden neuron is added.



The baseline values used for training were those detailed by Gedeon et. al, where L1 = 0.2, L2 = 0.005 and L3 = 0.001. The P value was set to 10, which was experimentally determined to give good results.

2.3 SARprop

Simulated annealing Rprop (SARprop) is a backpropagation algorithm devised by Treadgold and Gedeon[4] as an improvement to the Resilient backpropagation (Rprop) algorithm, and has been observed to improve the generalisation of casper[1]. It uses a simulated annealing term to achieve a regularisation effect on training the newest hidden neuron that attenuates as more time passes before the next one is added. The casper network trained on the EEG features used SARprop with the same gradient outlined in the original casper paper[1] of

$$\delta E/\delta w_{ij} = \delta E/\delta w_{ij} - k^* \text{ sign}(w_{ij})^* w_{ij}^2 \ * 2^{-0.01*HEpoch}$$

Where k is a user defined parameter and HEpoch is the number of epochs since the last addition of a hidden neuron. The casper network trained on the EEG features used a k value of 0.025, which was determined experimentally. The (SA)Rprop values used to train the casper network are the same as in Gedeon et. al[1]: $\eta^+ = 1.2$, $\eta^- = 0.5$, $\Delta_{max} = 50$, $\Delta_{min} = 1 \times 10^{-6}$.

2.4 Genetic Algorithm

A genetic algorithm was used to evolve a population of chromosomes with six genes corresponding to the following of casper's hyperparameters: The values A and P in an expression of the form A + P * N used to determine the number of epochs between adding hidden units, the L1, L2, and L3 learning rates, and the parameter k used in SARprop. Genomes are represented as an array of six values. The first four of these values are integers, the first two being the values for A and P, and the second two being a number representing how many times larger L1 is than L2 and how many times L2 is larger than L1. This was done to maintain the relationship L1 >> L2 > L3 [1]. The final two values are floating point representations of the L1 and k parameters. Thus the genotype of the baseline parameters expressed as a chromosome would take the form

[15, 10, 40, 5, 0.001, 0.025]

The algorithm followed the typical procedure of starting with a random population, evaluating each individual's fitness, then selecting and breeding individuals to create the next generation. Fitness was determined by training a casper model with the encoded parameters on the whole dataset using three fold cross validation. The data was split into three groups and the model was trained three times, using two of the groups for training and one for testing. An individual's fitness was measured as the average testing accuracy over the 3 training runs. This was done to ensure that the model could be evaluated while minimising the time taken as well as the interference posed by the inherent stochasticity in training the model. Due to the time expense posed by the fitness function, a 'hall of fame' style selection method was employed to remove the need to evaluate the fitness of the whole population at each generation. At each generation, the population is ranked by their fitness and the worst performing 25% of individuals are discarded. The top 75% are then put through a binary tournament style selection procedure. A set of pairs the size of 50% of the population is selected, then the fittest of each pair is bred at random with another of the winners. The crossover procedure generates two offspring, hence the bottom 25% of the population are replaced with the new offspring while the top 75% are retained into the next generation. Uniform crossover is used to generate new offspring, with each offspring having a 50% chance to inherit either parent's copy of a gene for each of its genes. Offspring are randomly mutated in accordance with a mutation rate initially set to 0.05. This gradually reduces with each generation by:

$$m_{t+1} = m_t - m_0 / (1.1 * n)$$

Where t is the current generation, n is the number of generations to be executed, and m is the mutation rate. This is done so that exploitation is favored more as evolution progresses. After reproduction, each new offspring's genes are given the chance to mutate according to the mutation rate. A gene is mutated by setting it to a randomly selected value that falls within a predefined range for that parameter (see Table 1.)

Table 1. Range of possible values for each parameter

Parameter	A	P	L1 multiplier	L2 multiplier	L3	k
Minimum	1	2	20	2	0.001	0.01
Maximum	30	20	100	10	0.01	0.1

3 Results

3.1 Genetic Algorithm Evolution Process

The genetic algorithm was run with a population size of 96 individuals for 40 generations. The highest performing chromosome given by this process produced the following hyperparameters: A = 30, P = 2, L1 ≈ 0.39, L2 ≈ 0.015, L3 ≈ 0.003. Several smaller trials were also run with a population size of 16 individuals for 5 generations, and with 96 individuals for 5 generations. Only the largest trial was able to produce a chromosome that performed better than the baseline parameters.

3.2 Feed Forward Network Topology

In a paper focused on feature extraction using the same dataset, Yao et. al used a 3 layer feed forward network as a classifier for their extracted features[2]. This investigation therefore used a 3 hidden layer configuration for the feed forward network. A general heuristic for the number of neurons in the final hidden layer of a feed forward network is the number of expected higher order features in the data. The features in this data are split into 3 channels for alpha, beta and gamma brainwaves; different mental states are associated with differing ratios of activity for these frequencies[6], hence the third hidden layer was given 3 hidden neurons. The behaviour of casper was used to inform the number of hidden neurons, as the aim was to make a comparison between casper and a feed forward network with roughly the same number of hidden neurons. Thus each layer has 3 hidden neurons for a total of 9 hidden neurons, which is the median number of hidden units added to the baseline casper model for this problem. The hidden neurons used a sigmoid activation function, and the network was optimised using RMSprop.

3.3 Comparing Performance

The models were evaluated using k fold cross validation with k = 10. Each casper network was trained for 300 epochs, while the feed forward network was trained for 1000 epochs. This was done 10 times for each of the k test sets. All input data was shuffled and normalised before training. Their accuracy was recorded at the end of training for each of the k folds for both the training and testing set.

Table 2. Mean, median and variance for training set accuracy.

Network	Mean	Median	Variance
Casper Baseline	77.27	81.98	22.53
Casper GA	80.13	77.71	27.12
Feed Forward	75.86	76.37	6.48

Table 3. Mean, median and variance for testing set accuracy.

Network	Mean	Median	Variance
Casper Baseline	75.78	77.26	25.53
Casper GA	77.97	78.03	31.46
Feed Forward	75.47	75.67	6.98

Table 4. Mean, median and variance for number of hidden neurons added by casper

Network	Mean	Median	Variance
Casper Baseline	7.92	9	8.91
Casper GA	6.52	7	4.45

3.4 Relation to Aims

The casper network given by the most successful genome from the evolution process shows a marginal improvement over the one given by the baseline parameters, and over a feedforward network with a comparable number of hidden neurons. This demonstrates the possibility for a genetic algorithm to be used to find a set of hyperparameters that can result in the tuning of the casper architecture for a specific problem. Not only did the configuration found by the genetic algorithm improve casper's average accuracy, it also reduced the number of hidden neurons required to achieve similar performance. The feed forward network performed slightly worse than both casper models and was slower to converge, but shows that the behaviour of both casper models is comparable to a similarly sized standard technique. Given that it takes approximately 10 seconds to evaluate the fitness of a single individual, the genetic algorithm can take several hours to run for a larger population size and number of generations. It took 4.5 hours using GPU acceleration on an NVIDIA GTX 980Ti to train the models in order for the genetic algorithm to complete 40 generations with 96 individuals. For this reason the use of casper with standard parameters or a feed forward model would be preferable in cases where an acceptable score can be achieved with the standard casper hyperparameters and manual experimentation, or where the character of the problem is known to an extent that is far enough to infer a satisfactory architecture for a feed forward model. A contributing factor to the difficulty in finding a well performing individual is the significant noisiness in the fitness function resulting from the stochastic nature of casper training due in part to random parameter initialisation and to the weight decay process in SARProp.

4 Outlook and Potential Improvements

It has been shown that genetic algorithms have potential to improve the performance of casper on a particular problem, although that improvement was marginal, and would not be worth the time expense in most circumstances. In cases where the fitness function of a genetic algorithm is noisy, increasing the population size can improve performance[5]. In order to make the use of a genetic algorithm that evaluates casper models by training them more worthwhile with respect to the time investment versus the performance gain in the result, a much larger trial with more individuals in the population and a larger number of generations should be carried out. Other considerations that may improve the algorithm used in this investigation include using two point crossover for reproduction, as this could be configured to keep the parameters relevant to the L1, L2, and L3 learning rates together during crossover, which may give better results given that those three genes are interdependent. Other measures that may increase the algorithm's resistance to noise could be increasing selection pressure by restricting the individuals that reproduce to the top 25% rather than a subset of the top 75%.

References

1. Treadgold, N. and Gedeon, T., 1997. A cascade network algorithm employing Progressive RPROP. Biological and Artificial Computation: From Neuroscience to Technology,.
2. Yao, Y., Plested, J. and Gedeon, T., 2018. Deep Feature Learning and Visualization for EEG Recording Using Autoencoders. Neural Information Processing,.
3. Fahlman, S.E. and Lebiere, C.: The cascade-correlation learning architecture, In: Advances in Neural Information Processing Systems, Vol. 2, Morgan-Kauffman, Los Altos, 1990.
4. Treadgold, N. and Gedeon, T., 1998. Simulated annealing and weight decay in adaptive learning: the SARPROP algorithm. IEEE Transactions on Neural Networks, 9(4).
5. Grushin, A,, 2004. The Effects of Static Fitness Function Noise Upon the Performance of Genetic Algorithms. 7th Joint Conference on Information Sciences, pp. 275-278
6. Koudelková, Z. and Strmiska, M., 2018. Introduction to the identification of brain waves based on their frequency. MATEC Web of Conferences, 210, p.05012.