Evolutionary Algorithms and Binary Classification

Safee Azam

Research School of Computer Science,
The Australian National University,
u7047994,
safee.azam@anu.edu.au

Abstract. Modern neural networks used for classification, typically utilise functions in the output layer that are thought to yield probabilities. It was shown around three decades ago, that using these values as probabilities is not correct, as the network was not explicitly trained on them. As a consequence, the sacrosanct classification threshold of 0.5 for binary classification problems may not be so untouchable. We use an Evolutionary Algorithm to not only optimise the typical hyperparameters of a network, but to also optimise this classification threshold. We show on a stress recognition task, that our EA identifies a network that performs better than one hand-picked. Our results are not as good as other basic methods, however, our dataset is small and challenging so we do not believe the results to be conclusive.

1 Introduction

Neural networks are increasingly being used in classification problems. In the output layer, a softmax function is often used in the output layer to obtain "probabilities" of class membership for a given network input. The class with the highest of these "probabilities" is then assigned as the output class. This has become standard practice with modern neural networks used for classification.

Another well known technique in computer science, which has also benefitted from the increased availability of comptation, and interest in artificial intelligence techniques is that of Evolutionary Algorithms [7]. EA's mimic the natural selection process, where "fit" individuals can exchange and pass-on their characteristics to their offspring, with some mutations. After many generations of this process, the resulting individuals are very likely to be more suited for their environments.

We will provide more details on the above mentioned techniques below.

1.1 Probabilities as Confidences

An important question to ask however, is whether these "probabilities" actually reflect the confidence of the prediction. Kogan [3], pointed out 30 years ago, that this was not the case. They showed that networks trained for categorisation, can not also be used to derive confidence scores. That is the "probabilities" typically are obtained in the output are not true probabilities and can not be used as such. One would instead need to train a network on the class membership probabilities explicitly, to obtain the true probabilities.

Adding to this finding, Milne et al. [2], pointed out that consequently, the ubiquitous threshold of 0.5 for binary classification is not somehow unchallengeable. This threshold can be varied and adjusted in order to minimise false results while retaining correct results. In their example [2], they were particularly interested in minimising false positives and so they adjusted the threshold value accordingly. We will apply this technique to our dataset of interest.

We also point out that recent work has also brought this problem to light. Guo et al. [5], refer to this as calibration. They define calibration as "predictive probability estimates representative of the true correctness likelihood" [5] and show that modern state of the art networks, are poorly calibrated. Bayesian models are a natural step to obtain model/prediction confidences and this has been implemented by [6], who show improvements in their confidence scores over traditional methods.

1.2 Evolutionary Algorithms for Hyperparameter Selection

EAs have many applications, including variable optimisation, structural design or improvement where an existing working solution is in place [7]. Our application of interest is variable optimisation, specifically, we wish to optimise the hyperparameters of our network using EAs. In this case, the "individuals" are each set of hyperparameters, their "fitness" is the accuracy of the network, mutations occur by randomly altering these hyperparameters, and offspring are produced by selecting new individuals from the set of parent hyperparameters.

1.3 Dataset

Sharma et al. [4] created a dataset with the aim of working on stress recognition. Stress is a major concern to a person's health, physical, mental and social [4]. Stress research is thus beneficial to society give the potential improvements it can make to health and the subsequent benefits that that promotes. Its non invasive recognition is an important part of this research.

For the creation of the dataset [4], video data in the visible spectrum (RGB) and the thermal spectrum was collected. This was obtained by having 35 subjects watch stressed and non-stressed film clips while being recorded with an RGB and a thermal camera. Subsequent surveys verified the subjects state when watching the clips, and the data was validated accordingly.

Irani et al. [1] have processed and used this data in their model for stress recognition and which we will also use. This dataset contains 20 observations for 31 subjects for a total of 620 observations. For each observation we have the five top principal components of the RGB image and thermal image. These features are large floating values. There are only two labels, calm and stressful, with exactly half of the observations falling into each category. We describe some distirbutional statistics for the data in the table below.

Table 1. Mean and Standard Deviation of top RBG and Thermal Components by label.

Label	RGB Component 1 (AVE)	RGB Component 2 (STD)	Thermal Component 1 (AVE)	Thermal Component 1 (STD)
Calm	1,107,870	393,509	263,022	61,420
Stressful	1,112,094	396,873	262,540	58,189

Looking at table 1, we see the different scales the values appear to be on, or rather, the lack of direct meaning of the value from the principal component analysis. This indicates that normalisation of the data will be necessary to stop the network being influenced by only the magnitudes of its inputs.

From this cursory look, we also find little difference in component value by label. The difference in means for both components between calm and stressful is much less than 1%. This is a small and challenging dataset and we would do well to classify better than 50%.

2 Method

It is clear that the technique of adjusting the threshold of classification would be useful for stress recognition. It may be that we would want to be quite sure that the subject is stressful, and so considering this as the "positive" case, we would want to minimise the false positives of our model. That is we would prefer to label someone stressful as calm, that to mislabel a calm person as stressful. It is easy to see how making a mistake like this in the real world may increase the stress of a calm person!

Furthermore, we wish to use an Evolutionary Algorithm, to optimise the hyperparameters of our network as well as the classification threshold, which we call θ.

In our previous work, we determined via manual exploration an optimal network and value for θ. We will use this to compare with the resulting optimal network form our EA.

2.1 Data preprocessing

The component values in the dataset vary wildly and appear to be of different scales. For them to be able to be used best by the network, we normalise each of these ten features so that they have zero mean and unit standard deviation.

Consequently we do a simple 80:20 split of the data for training and testing for each of our models examined as part of the EA. For more rigorous results, we conduct a 10-fold cross-validation on our baseline model and best model from the EA. That is we withold 10% of the data fitting the model to the rest and testing on the witheld portion, we do this 10 times and calculate summary statistics of our testing accuracy.

2.2 Baseline Network Architecture

Our baseline is a simple network with two hidden layers, as desribed below.

- First hidden layer with 24 neurons and a ReLu activation.
- Second hidden layer with 12 neurons and a ReLu activation.
- Output layer of 1 neuron with a sigmoid activation.
- $\theta = 0.6$

2.3 Evolutionary Algorithm

We use an EA to optimise the hyperparameters of our network. We will use the same architecture as our baseline, however, we allow to EA to choose the following:

- Number of neurons in first hidden layer: ranging from 4 to 36, increments of 4.
- Number of neurons in second hidden layer: ranging from 4 to 36, increments of 4.
- Activation function used for both hidden layers: ReLu, Sigmoid or Hyperbolic Tan.
- Learning rate of gradient descent algorithm: 0.1, 0.01 or 0.001.
- θ: ranging from 0.3 to 0.7, increments of 0.05.

Note that there 7290 possible combinations of these hyperparameters. If each training iteration took only 10 seconds, it would take 20 hours to calculate all of them. Whilst, this is still feasbile, our dataset is relatively small, so the benefit of EA will still be clear in our case, however, will be much more obvious with large datasets.

We now describe our EA:

1. Initialise 25 random individuals by randomly selecting one of each of the above hyperparameters.
2. Train each individual for 10 000 epochs on the training set and evaluate on the testing set.
3. Retain the top 4 individuals by test accuracy.
4. Mate (or crossover) all combinations (six of them) of two individuals (parents), by choosing one of the hyperparameters from each parent. Do this twice for each parent combination so there are 12 offspring in the next generation.
5. Randomly mutate each of the new offspring by re-picking a value for a randomly chosen hyperparameter.
6. Repeat steps 2 to 5.

We run this scheme for 7 generations, so we would have explored around 100 individuals (which aren't guaranteed to be unique). With our EA we hope to find the optimal hyperparameter selection by only exploring a very small subset of all options.

3 Results

We report the top individual from each generation below.

Table 2. Top individual by generation

Generation	Layer 1 Neurons	Layer 2 Neurons	Activation	Learning Rate	θ	Accuracy (%)
1	16	16	Tanh	0.1	0.7	54.2
2	32	36	Tanh	0.001	0.5	51.7
3	4	36	Tanh	0.1	0.7	52.5
4	36	36	Sigmoid	0.1	0.5	53.5
5	36	12	Sigmoid	0.001	0.5	55.8
6	4	12	Sigmoid	0.1	0.5	51.7
7	16	12	Sigmoid	0.1	0.5	53.3

We find that there seems to be little learning and improvement from generation to generation. Indeed the second highest accuracy was achieved after the first generation which was randomly generated. We are not sure if the hyperparameters of the EA itself are to blame or if we are already approaching the highest test accuracy on what is a challenging dataset.

There are however, lessons we can learn regarding hyperparameter selection. We note that in no generation was a model using ReLu the top performer. This is different to our previous finding that ReLu was the best activation function. We also note that $\theta = 0.5$ for five out of seven generations. It seems that the cannonical value of θ was found to still be the best in this problem.

3.1 Comparisons

Typically, we would take individuals from the last generation to examine further, however, we note that the top individual from generaton 5 was stronger than those from the last generation and the strongest overall. Therefore we compare this individual with our baseline model running a 10-fold cross validation see table 3 below.

We find that our best EA network has on average a higher accuracy than our baseline. This suggests that the EA was able to find a better model than we were able to "manually". We note however, the low accuracy of both models, barely being above 50%. We believe this is due to the small dataset size and the limited number of features.

We further compare with [1] in which their most basic model exploited the full data and acheived an accuracy of 61%. Their more sophisticated models were able to achieve an accuracy of 89%. Clearly our simple models cannot perform at the level of the more sophisticated models, however, our model found with an EA was within 10% of their simplest model.

Table 3. Results of cross validation for baseline and best network from EA.

Model	Mean Accuracy	Std of Accuracy
Baseline	49.5	1.7
Best EA Network	52.8	5.2

4 Conclusion

We used a simple EA to optimise the hyperparameters for a stress recognition problem. One of these parameters was the binary threshold of classification, θ. We found that our EA struggled to improve and find a model with accuracy much greater than 50%. We also found that θ was almost always identified as 0.5. We do not hold strong weight in these results however, as our dataset was very small and challenging. Further work should apply our EA to explore choices of θ over larger datasets.

References

1. Irani, R., Nasrollahi, K., Dhall, A., Moeslund, T., Gedeon, T.: Thermal Super-Pixels for Bimodal Stress Recognition. 2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA), pp. 1-6 (2016)
2. Milne, L., Gedeon, T., Skidmore, A.: Classifying Dry Sclerophyll Forest from Augmented Satellite Data: Comparing Neural Network, Decision Tree & Maximum Likelihood, Proceedings 6th Australian Conference on Neural Networks, pp. 160-163, (1995)
3. Kogan: Neural networks trained for classification can not be used for scoring, IEEE Trans. on Neural Networks, (1991)
4. Sharma, N., Dhall, A., Gedeon, T., Goecke, R.: Thermal spatio-temporal data for stress recognition, EURASIP Journal on Image and Video Processing, (2014)
5. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.: On Calibration of Modern Neural Networks, Proceedings of the 34 th International Conference on Machine Learning, (2017)
6. Mandelbaum, A., Weinshall, D.: Distance-based Confidence Score for Neural Network Classifiers, (2017)
7. Sloss, A., Gustafson, S.: 2019 Evolutionary Algorithms Review, (2019)