CLASSIFYING SCREEN SIZE FROM EYE GAZE SEARCH DATA: NEURAL NETWORK CONSTRUCTION WITH GENETIC ALGORITHM TO SELECT FEATURES AND DETERMINE THRESHOLD BOUNDARIES

Kangjie Yin
Research School of Computer Science, Australian National University
u6015236@anu.edu.au

Abstract

Due to the convenience of mobile devices, more and more people would like to use mobile devices. Which have smaller screen sizes, to search for information. However, users may behave differently from small screen size and large screen size, and the search performance may also vary. We use a data set of eye-tracking and build a neural network to predict the screen size in terms of user search behaviour and search performance, and we determine the threshold boundaries to guarantee that some entries could be labelled correctly. After that, we use genetic algorithm to do feature selection to improve the performance of the neural network. We found that the neural network increases the accuracy to some degree. This suggests trying some other neural networks and genetic algorithm operators.

1. Introduction

Nowadays, people are used to doing searching on the Internet. Searching is not only could be done with personal computers or laptops, but also could be done with various mobile devices which have smaller screen sizes. For instance, iPads and mobile phones are commonly used to browse web sites, reply emails, and do many things which could also be done on desktops. However, users may behave differently when they are facing screens of different sizes. Aula, Räihä, and Majaranta (2005) point out that the eye gaze reflects how elements on the web pages receive attention, because eye-tracking helps us understand user attention in a deeper way. Kim, Thomas, Sankaranarayana, Gedeon, and Yoon (1995) explore the relationships between search performance and users' search behaviours. They use eye-tracking to explore user behaviour and performance and focus on whether screen sizes will influence user search behaviours.

Therefore, reversely, is it possible to predict the screen size in terms of user behaviour and performance? To investigate this, we reuse the eye gaze search dataset, apply genetic algorithm to select significant features, and build a classification neural network to classify the screen size based on user search

behaviour and performance.

2. Method

The original dataset consists of search performance (4 attributes), search behaviour (22 attributes), screen size attribute and task attribute. Consider that our aim is to predict the screen size based on user search behaviour and performance, all search performance and search behaviour attributes are used as input data. In the meanwhile, the screen size attribute is the target label in our classification model.

2.1 Pre-process data set

There are three tabs in the original data set while we only use the **All data** tab. The number of search performance attributes and that of search behaviour attributes add up to 26, among which 25 attributes are numeric and 1 attribute is statistical. Consider that our goal is to predict the screen size using search behaviour and search performance attributes, the **task** attribute will not be taken into consideration, and we remove it.

The statistical attribute is **Strategy** attribute. To be specific, it is a categorical attribute since the data value can only be one of DF (depth-first strategy), BF (breadth-first strategy), and MX (mixed strategy). Some method like a decision tree can handle categorical inputs directly with no need to encode them. Consider that a neural network can only work with numerical inputs, we need to encode **Strategy** first.

One-hot encoding method is applied here to convert category values into numeric values and then binarize integer variables (Okada, Ohzeki, &Taguchi, 2019). With one-hot encoding, each categorical value is represented as an array-like of integers, with one single valid value 1. Therefore, the **Strategy** attribute could be encoded as Table 1 shows. In this way, to gain more reasonable data, we not only convert categorical values into numerical but also avoid ordinal relationships between encoded values (Browniee, 2017; Qiao, Yang, &Wu, 2019).

Table 1. Encoded Strategy attribute

BF	[1, 0, 0]
DF	[0, 1, 0]
MX	[0, 0, 1]

After encoding completed, we need to split data set into a training set, a validation set, and a testing set. The original data set consists of data from 32 different subjects, and each subject is involved in 20 rows of data. Among these 20 rows, 10 rows are labelled with size **L** while the rest are labelled with size **s**. Therefore, we reserve the data of the first 20 subjects as training data, the next 3 subjects are validation data, and the other 9 subjects become testing data. In this way, we guarantee rows with label **L** accounts for 50% in all data sets.

2.2 Genetic Algorithm

After pre-processing, the number of attributes becomes 28. Next step is to adapt genetic algorithm as the feature selection method to filter out unimportant features. Feature selection is a common strategy in data mining and machine learning problems, and the main reason of it is to gain cleaner and more comprehensible data (Cheng et al., 2017). With a smaller number of features, the complexity of the neural network is reduced, and the training could be accelerated. Also, feature selection improves the model accuracy because the only important features are reserved.

Genetic algorithm (GA) is a popular type of evolutionary algorithm. It simulates the process of natural selection through mutation, crossover, and selection. It starts with a population of individuals and takes generations to complete the selection. In each generation, children are produced while parents are replaced to maintain the population size (Mitchell, 1998). A GA model is constructed using PyTorch.

To implement the GA model, population representation and several functions need to define. Each individual has a binary DNA which represents its attributes. For example, if a candidate can have 3 attributes at most and its DNA is [1, 0, 1], this means that the candidate only has the first and third attributes. In this way, each individual is represented as a DNA of size 28. The first function needs to define is target function. Consider that our goal is to predict the screen size using a neural network, the accuracy on the testing set is what we want to maximise. Therefore, the testing accuracy of a neural network is the return value with a DNA as the input. Next one is the fitness function. Because the accuracy is always non-zero and enough to reflect the fitness of an individual, we will not define the fitness function explicitly. The third one is the translation function which translate a DNA to the input of a neural network. That is, we need to extract a subset of data from the training set based on the DNA. The selection function plays a role in selecting population, and only selected population will reserve in next generation. Population with higher fitness value is more likely to be selected. The crossover function generates new offspring. We randomly select an individual from the population as the first parent, and then make another as parent 2. Then, crossover points are randomly chosen to produce a child based on parents' DNAs. It should be mentioned that crossover happens with a probability *cross_over* which is set 0.8. The mutation function adds diversity to the genetic characteristics of the population. Given a probability, an arbitrary bit in an individual's DNA will be flipped. The probability *mutate_rate* is set 0.004. This enables us a relatively high ability to explore new solutions.

2.3 Neural Network

A feedforward neural network is constructed using PyTorch. The neural network consists of an input layer, a hidden layer, and an output layer. These layers are in sequence. Stochastic Gradient Descent (SGD) optimiser is applied to hold the current state and update parameters based on the computed gradients. Compared to Gradient Descent, SGD is much faster when training a neural network (Bottou & Bousquet, 2008). The speed of training matters because the population size and the generation size are large and hundreds of thousands of neural networks need to be trained. The SoftMax function is chosen as the activation function. In the final layer of the neural network, each component of an input instance is assigned with a probability, and the largest probability will be compared with the threshold to

determine which class the input instance belongs to. Consider that we are building a classifier, cross entropy is preferred.

Before GA, we construct a neural network using all attributes as the input. The number of input neurons should equal to that of input attributes (28). Similarly, the number of output neurons is 2. Here, we follow three rules of thumb from Heaton (2017) to determine the number of hidden layer neurons to 19. The neural network will be trained for hundreds of epochs. To avoid overfitting, we use validation set to see how the network performs on unseen data. A confusion matrix is created for the results of training, validation, and testing, respectively. After we complete features selections, the input neuron number may change, and the hidden layer neuron number changes accordingly.

2.4 Adjust Threshold Values

We assume that the newest generation has the population with the highest quality, so the most fitted DNA after 100 generations is chosen and we extract the training set, validation set, and testing set based on it. After translating the DNA, we construct a neural network with different threshold values. What to do next is to determine the boundaries of the threshold. Apparently, for a binary classification model, one output class with a factor larger than 0.5 will be regarded as the predicted class, because the factor of the other output class will always be less than 0.5. However, Kogan (1991) points out that the probabilities of class membership need to be taken into consideration. The primary technique in Milne, Gedeon, and Skidmore's (2015) research is to determine the threshold values where false positives and false negatives appear (or disappear) respectively. Therefore, we vary the threshold from 0.40 to 0.60 and use 0.01 as a stride. For each threshold value, we record the number of correct labels, false positives, and false negatives, to find out expected threshold values.

3. Results and Discussion

Figure X shows the results of the last 10 generations. It could be seen that the last generation has the highest accuracy rate. Although the accuracy rate of a newer generation is not always higher than that of an older generation, roughly the accuracy rate is on the rise.

```
Generation: 89  Most fitted DNA: [1 1 0 1 1 1 1 0 0 1 0 0 0 1 0 0 1 1 1 1 0 1 1 0 0 1 1 0] Accuracy: 63.89
Generation: 90  Most fitted DNA: [1 0 1 1 1 1 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 1 1 1 0 1 1 0] Accuracy: 67.78
Generation: 91  Most fitted DNA: [1 1 1 1 1 1 1 0 1 0 0 0 0 1 0 0 1 1 1 1 1 1 1 0 0 1 1 0] Accuracy: 64.44
Generation: 92  Most fitted DNA: [1 1 1 1 1 1 1 0 1 1 0 1 0 1 0 0 1 1 1 0 1 1 0 0 0 1 1 0] Accuracy: 66.11
Generation: 93  Most fitted DNA: [1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 0 1 1 1 1 0 1 1 0 0 1 1 0] Accuracy: 65.56
Generation: 94  Most fitted DNA: [1 1 0 1 1 1 1 1 0 1 0 1 0 1 0 0 1 1 1 1 0 1 1 0 0 1 1 0] Accuracy: 66.67
Generation: 95  Most fitted DNA: [1 0 1 1 1 1 1 0 1 1 0 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0] Accuracy: 67.22
Generation: 96  Most fitted DNA: [1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0] Accuracy: 67.78
Generation: 97  Most fitted DNA: [1 0 1 1 1 1 1 0 1 1 1 1 0 1 0 0 1 1 1 1 1 1 0 0 0 1 1 0] Accuracy: 68.33
Generation: 98  Most fitted DNA: [1 1 1 1 1 1 1 0 1 0 0 1 0 1 0 0 1 1 0 1 0 1 1 0 0 1 1 0] Accuracy: 62.22
Generation: 99  Most fitted DNA: [1 0 1 1 1 1 1 1 1 0 0 1 0 1 0 0 1 1 1 0 1 1 1 0 0 1 1 0] Accuracy: 71.67
```

Figure 1. Most fitted DNAs and accuracy rates of the last 10 generations

Firstly, we construct a neural network without GA. When we set the threshold 0.5, the accuracy of training is 60.75% while that of the testing is 62.78%. The results of testing with different threshold values is shown below. If the threshold < 0.4, all entries are categorised as **s**, and if threshold > 0.6, all entries are categorised as **L**. When the threshold becomes 0.43, false positives appear. Similarly, 0.59 is

the largest threshold where false negatives still exist. Also, the accuracy of threshold 0.43 and threshold 0.59 on the testing set are 55.0% and 51.1%, respectively. The threshold of 0.5 maximises the accuracy of the model.

Table 2. Neural net performance on the testing set (before GA)

Threshold	Correct	False +ve	False -ve
0.40	90	0	90
0.42	**90**	**0**	**90**
0.43	99	8	73
0.50	111	36	33
0.59	92	84	4
0.60	**90**	**90**	**0**

Secondly, we construct another neural network with features selected by GA. When the threshold = 0.49, the accuracy is maximised (69.44%) and lager than our initial neural network accuracy. The threshold boundaries are in range [0.44, 0.58], and the accuracy rates of 0.44 and 0.58 are 50.0% and 50.5%, respectively.

Table 3. Neural net performance on the testing set (after GA)

Threshold	Correct	False +ve	False -ve
0.43	**90**	**0**	**90**
0.44	90	2	88
0.49	125	25	30
0.50	113	36	31
0.58	91	83	6
0.59	**90**	**90**	**0**

In this way, we have found boundaries of the threshold. Take threshold = 0.59 as an example, all errors are false negatives. That is, all entries, which are labelled as **s**, belong to small screen size. In the same way, all entries categorised as **L** belong to large screen size under the threshold = 0.43. With these two thresholds, we can categorise some entries accurately.

Our project is classifying the screen size in terms of user search behaviour and search performance, while Kim, Thomas, Sankaranarayana, Gedeon, and Yoon's (1995) research explores how screen size and task type influence user search performance and search behaviour, it is hard to compare the results directly. However, a very important finding is their research is "there is no significant difference between the 2 screens in time spent on search results pages and the accuracy of finding answers". This indicates some features have very limited effect on predicting screen size, and our project has showed that with some unimportant features ignored, the performance of the neural network could be improved to some extent.

4. Conclusion and Future Work

In this project, we construct a triple-layer neural network for classifying screen size based on user search behaviour and search performance. The accuracy on the testing is treated as the fitness function of our GA model. We use GA model to extract some features, then another neural network is built using these features as input. With feature selection, the result has a higher accuracy than that of the neural network without GA. We have found the threshold boundaries (0.43 and 0.59) of no false positives and no false negatives, and we are able to label some entries correctly with these threshold values. The technique increases the accuracy of the neural network by 6.66%, and the threshold which maximises the accuracy is found to be 0.49 instead of 0.5.

Some future works deserve trying. For example, the neural network here is a simple feedforward network, with one single hidden layer only. We could build a neural network with more hidden layers or try some other networks like a bidirectional neural network. Another future work could be choosing different values of crossover and mutation. Crossover and mutation influence the generation of new population significantly, and Hussain and Muhammad (2019) point out that it is important to find a balance between exploitation and exploration through adjusting the operators. Thus, appropriate choose of operators is likely to improve the performance of the GA model. Also, in this project, we simply replace the parent with its child. However, there are many other replacement strategies such as replacing worst and replace oldest. Lozano, Herrera, and Cano (2005) emphases that replacement strategies play an important role in bringing population diversity and improving the quality of the population. This will not only speedup generating individuals with higher fitness scores, but also increase the possibility of seeking out the global optimization.

Reference

Aula, A., Majaranta, P., & Räihä, K. (2005). Eye-tracking reveals the personal styles for search result evaluation. Lecture Notes in Computer Science, 3585, 1058–1061.

Bottou, L., & Bousquet, O. (2008). The tradeoffs of large scale learning. In *Advances in neural information processing systems* (pp. 161-168).

Browniee, J. (2017). Why One-Hot Encode Data in Machine Learning? Retrieved from: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

Hassanat, A., Almohammadi, K., Alkafaween, E., Abunawas, E., Hammouri, A., & Prasath, V. B. (2019). Choosing Mutation and Crossover Ratios for Genetic Algorithms—A Review with a New Dynamic Approach. *Information*, *10*(12), 390.

Heaton, J. (2017). The Number of Hidden Layers. Retrieved from: https://www.heatonresearch.com/2017/06/01/hidden-layers.html

Hussain, A., & Muhammad, Y. S. (2019). Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator. *Complex & Intelligent Systems*, 1-14.

Kim, J., Thomas, P., Sankaranarayana, R., Gedeon, T., & Yoon, H. J. (1995). Eye-tracking analysis of user behavior and performance in web search on large and small screens. *Journal of the Association for Information Science and Technology*, *66*(3), 526-544.

Kogan. (1991). "Neural networks trained for classification can not be used for scoring," IEEE Trans. on Neural Networks.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, *50*(6), 1-45.

Lozano, M., Herrera, F., & Cano, J. R. (2005). Replacement strategies to maintain useful diversity in steady-state genetic algorithms. In *Soft Computing: Methodologies and Applications* (pp. 85-96). Springer, Berlin, Heidelberg.

Milne, L. K., Gedeon, T. D., & Skidmore, A. K. (1995). Classifying Dry Sclerophyll Forest from Augmented Satellite Data: Comparing Neural Network, Decision Tree & Maximum Likelihood. training, 109(81), 0.

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

Okada, S., Ohzeki, M., & Taguchi, S. (2019). Efficient partition of integer optimization problems with one-hot encoding. *Scientific Reports (Nature Publisher Group), 9*, 1-12. doi:http://dx.doi.org.virtual.anu.edu.au/10.1038/s41598-019-49539-6

Qiao, Y., Yang, X., & Wu, E. (2019, May). The research of BP neural network based on one-hot encoding and principle component analysis in determining the therapeutic effect of diabetes mellitus. In *IOP Conference Series: Earth and Environmental Science* (Vol. 267, No. 4, p. 042178). IOP Publishing.