A Binary Classifier of an Eye-tracking Data Set: Implementation an Artificial Neural Network Classifier with Threshold Techniques and Genetic Algorithm

Yukun, Hao
Research School of Computer Science, Australian National University
U6013736@anu.edu.au

Abstract. Neural network is widely used for prediction task in modern computer science area, with more researchers are contributing to it. This paper will mainly focus on an implementation of an artificial neural network using Python/PyTorch on a real world eye-tracking dataset. A binary classification problem is issued and solved with an ANN classifier, and the performance will be evaluated. A threshold technique will then be applied on the classifier, to observe whether it is helpful for improving prediction accuracy. Also, a genetic algorithm for feature selection will be applied, to see if some features are not relevant to the classifier.

Keywords: Artificial Neural Network, Eye-tracking, Pre-processing data, Threshold, Genetic Algorithm, Feature Selection

1. Introduction

This paper describes a neural network, which is designed to solve a classification problem with an eye-tracking dataset (Kim, Thomas, Sankaranarayana, Gedeon & Yoon 2015). To achieve the dataset: http://cs.anu.edu.au/~tom/datasets/eyegaze-search1.zip. The dataset contains 29 columns of data, and details of each column is shown in figure 1-1.

Id	Attribute name	Attribute value	Id	Attribute name	Attribute value
1	Subject	Integer	16	Strictly Linear	0 / 1
2	Size	“L” / ”s”	17	Strictly Linear W/ID	0 / 1
3	Task	“info” / ”nav”	18	Regression rate	0 / 1
4	Time to first click	Decimal	19	Regression distance	Integer
5	Task completion duration	Decimal	20	Regression to 1 or 2	0 / 1
6	Accuracy	0 / 1	21	Skip	0 / 1
7	Wrong answer	0 / 1	22	Skip distance:	Integer
8	Mean fixation duration	Decimal	23	Skip to 1 or 2	0 / 1
9	Mean fixation duration for on link	Decimal	24	ScanDown	Integer

10	Minimal scanpath value	Integer	25	ScanUp	Integer
11	Compressed scanpath value	Integer	26	Maximum gaze position	Integer
12	Compressed/Minamal:	Decimal	27	Strategy:	“DF” / “BF” / “MX”
13	Complete	0 / 1	28	Trackback	Integer
14	Linear	0 / 1	29	Page visit	Integer
15	Linear W/ID	0 / 1			

Figure 1-1: Types of attributes in dataset

I have chosen this dataset since it has enough data with varieties of attributes, which can provide enough information to the neural network model during training. Besides, there is no missing value in the dataset needed to be processed, and data are in high accuracy. Also, the eye-tracking problem is an interesting topic for me.

The classification problem defined for the dataset is to predict if a data belongs to size “L” or “s”, so the attribute “Size” is used as label (output in the NN). There are 25 input attributes for the classifier, which is shown in figure 1-1, from attribute No.4 to attribute No.29, exclude attribute No.27.

A neural network has been trained to solve the binary classification problem, with SGD optimiser, to observe the accuracy and confusion matrix of the NN. After that, a technique about changing threshold during making prediction is implemented on the NN after the observation, with new accuracy and confusion matrix comparing to the original NN to see if the threshold technique has improved the NN or not.

2. Method

2.1 Pre-processing of the Dataset

The details of the eye-tracking dataset have been described in figure 1-1, with some attributes are non-numerical, and some attributes are not considered as a part of the problem. Attributes “Task”, “Size” and “Strategy” can be regarded as label, and “Size” is selected as output, while “Task” and “Strategy” are deleted.

There are 27 attributes remaining in the modified dataset, as described above, from attribute No.4 to attribute No.29, exclude attribute No.27 but with attribute No.1 (“Subject”). In addition, values in the label attribute “Size” are modified to make the dataset fit in with the NN technique, where label “L” is regarded as integer 1 and label “s” as 0. The modified dataset file is called “eyegaze_after_preprocess.csv”.

The next step is using python code to do further pre-processing. Dataset are loaded and the attributes “Subject” is dropped, since it only describes which data belongs to which subject, acting as orders or data IDs and can give no meaningful information to the classifier. Attributes’ names are dropped, so only

useful data are remaining for further training and testing process. Data are re-ordered randomly and splitted, with 80% of data are regarded as training dataset and the rest are for testing. The first column is used as output and the rest are input data. All input data being used for training process and testing process are normalized, from 0 to 1, to guarantee that all input features can contribute to the neural network model eaqually.

2.2 Implementation of a Binary Artificial Neural Network Classifier:

For the specific classification problem described in this paper, a neural network is trained using Python and PyTorch programing, with 3 layers includes an input layer, a hidden layer and an output layer with 2 neurons. Overfitting is a widely considered problem when using neural network model, and models can be inaccurate if the parameters are not set well for a specific problem (Hawkins, 2004). Thus, to improve the performance of the neural network, numbers of hidden neurons and training epochs are important. Different values of hidden neurons and epochs are tested for 20 times, and the average accuracies for each pair of values are shown in figure 2-1, where column stands for epochs number and rows are hidden neurons number.

(Hidden, Epoch)	400	500	600	700
15	64.63%	65.87%	65.39%	64.44%
20	66.25%	67.12%	66.11%	64.89%
25	63.13%	64.97%	66.49%	65.26%
30	63.07%	65.10%	66.17%	64.89%

Figure 2-1: Average accuracy of different setting for hidden neurons and epochs

From figure 2-1, it is figured out that overfitting can happens when the model is trained for too many epochs since the accuracy is decreasing. Also, as hidden neurons number increases, the training accuracy could be affected. So, to the specific problem defined in this paper, I have chosen the highest accuracy hidden neurons and epoch pair demonstrated above, which is: 20 hidden neurons and 500 epochs.

To summarize, the parameters for the neural network is:

```
input_neurons = n_features; (n_features = 25 here)
hidden_neurons = 20;          output_neurons = 2
learning_rate = 0.005;       num_epochs = 500
```

Figure 2-2: Parameters setting for the neural network

2.3 Performance Evaluation of the Neural Network Classifier:

There are mainly two methods for evaluating the performance of the neural network described above. One method is the accuracy of the predicting results. However, the accuracy is calculated with testing dataset, but not training dataset, to prevent problems of overfitting. 20 percent of data are using for testing the accuracy of the NN model, to judge whether it is overfitting, or its generalization ability is good

enough. The accuracy is calculated as figure 2-3 shows.

$$accuracy = \frac{\text{true positive} + \text{true negative}}{\text{number of all data}} = \frac{\text{number of data being correctly predicted}}{\text{number of all data}}$$

Figure 2-3: Formula for accuracy calculation

The other method is confusion matrix, which is widely used to observe prediction results for classification problems (Lewis & Brown 2001). In this case, the confusion matrix is a 2 by 2 size matrix, with column means the predicting results and the rows for real labels. Figure 2-4 is an example of how a confusion matrix look like, with top-left and bottom-right numbers means correct prediction results.

(Real class/Prediction class)	Class 0	Class 1
Class 0	40	15
Class 1	21	41

Figure 2-4: A confusion matrix example

2.4 Using a Threshold Technique to Improve the Classifier

In the neural network described above, when using the output neurons to make a prediction of which class does a specific data belongs to, a "torch.max()" function is used. The function can receive the output values from each output neuron and choose the neuron with the largest value as the prediction output. For instance, figure 2-5 shows an example of how a pair of output neuron values can be used to make a prediction.

```
Y_pred is considered as holding the out value of the output
layer, which can be represented as a pair of numbers:

[7.2871e-01, 9.7955e-02], output from testing data row 1;
[3.9019e-01, 3.9635e-01], output from testing data row 2;
[7.8267e-01, 2.3727e-01], output from testing data row 3;

for each pair of Y_pred data, the predicting process is:
7.2871e-01 > 9.7955e-02, so predicted class is: Class 0;
3.9019e-01 < 3.9635e-01, so predicted class is: Class 1;
7.8267e-01 > 2.3727e-01, so predicted class is: Class 0;

so the predicted result is:
Predicted = [0, 1, 0]
```

Figure 2-5: The prediction process using "torch.max()"

However, Milne, Gedeon and Skidmore (1995) argue that choosing the largest one from the output neurons as the class of a data is not essential. For some binary dataset, setting the threshold as 0.5 (if the output values add up to 1) might give worse performance (Kogan 1991), comparing to have threshold larger or smaller than 0.5. Thus, to test with different threshold when predicting classes from output neurons, a "torch.softmax()" function is used instead of "torch.max()", which can apply a softmax

function to output values. As a result, the output values now add up to 1, which make setting a threshold from 0 to 1 possible, as shown in figure 2-6.

```
Set threshold = 0.45
rather than using torch.max(), here we use softmax()

the softmax function will give output like:
[0.5181, 0.4819];
[0.4440, 0.5560];
[0.3248, 0.6752];

#each pair of output values now add up to 1, and will be used for
predicting based on the threshold:
0.5181 > threshold, so the prediction class is: Class 0;
0.5560 > threshold, so the prediction class is: Class 1;
0.6752 > threshold, so the prediction class is: Class 1;
```

Figure 2-6: Set a special threshold for predicting

2.5 Using a Genetic Algorithm for Feature Selection

Genetic algorithm is a bio-inspired technique learning from nature evolution to deal with a wide range of problems (Reeves, Rowe & SpringerLink, 2003). It can be used to find optimal or acceptable solutions when doing feature selection, according to Yang & Honavar (1998), since genetic algorithm can achieve multicriteria optimization considering select subsets and will usually perform well if well designed. Thus, in order to improve the performance of the classifier, I have applied a genetic algorithm to select features from the original dataset.

In the genetic algorithm, DNA on each chromosome is a binary value, stands for if a feature should be used or not, where 1 stands for using a feature and 0 for not using it. Thus, the DNA size is 25 since there are 25 features in total. The chromosome with DNA is shown as figure 2-7.

```
Chromosome = [1 0 1 0 0 0 0 1 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 1 0]
So, this chromosome means: The first feature is used;
                           The second feature is not used;
                           …
```

Figure 2-7: A chromosome example

The fitness function of the genetic algorithm is based on the testing accuracy given by the NN described section 2.2, where the NN should be trained and tested 10 times with some features given by a chromosome, and the average accuracy is calculated as the fitness value.

The reproduction process is based on fitness value given by fitness function. Chromosome with higher fitness value will have more chances to be selected as parents during the reproduction. If a chromosome is selected, then it will be recombined with a random another chromosome to produce children. The ranked selection of parent will make those chromosomes with higher fitness value have higher chances to reproduce, while the random selection of the other parent can provide more possibility solution space.

The mutation function uses random mutation as its rule, so there might be more possibilities space can be discovered. After reproduction process, all parent chromosomes will be abandoned and those new chromosomes are set as the next generation.

The overall settings of the GA are demonstrated in figure 2-8:

```
DNA_size = 25, since there are 25 features in total;
population_size = 40, so each generation will have 40 chromosomes;
mutation_rate = 0.02, so each DNA on a chromosome will have 2% of
                      chance to mutate to a different value.
generations = 30, so there are 30 generations tested for the GA
```

Figure 2-8: GA settings

3. Results and Discussion

3.1 Results of Threshold Technique:

A threshold technique is described in section 2.4, and to test with it, the original NN demonstrated in section 2.2 is used, with 500 epochs, 20 hidden neurons and 0.005 learning rate. Figure 3-1 shows how the average accuracy (testing 20 times) change when using different threshold (represented as θ) from 0.25 to 0.75 (θ= 0.5 is the baseline).

θ = 0.25	θ = 0.30	θ =0.35	θ = 0.40	θ = 0.45	θ = 0.50
64.59%	63.40%	66.81%	69.17%	69.28%	67.13%
θ = 0.55	θ = 0.60	θ = 0.65	θ = 0.70	θ = 0.75	
66.77%	67.17%	65.93%	65.33%	64.25%	

Figure 3-1: Average accuracy for different threshold θ

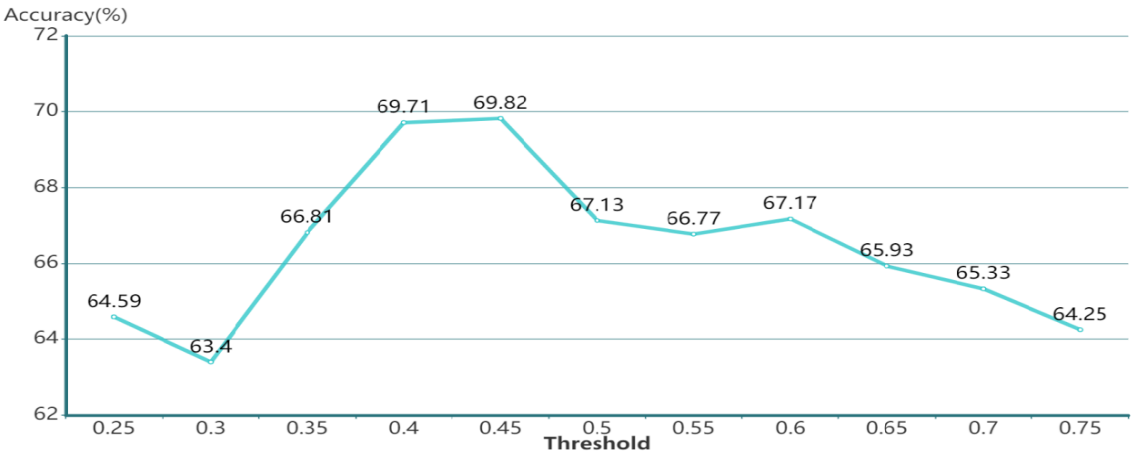

Figure 3-2: Line chart for threshold and accuracy

To make it easier to observe the relationship between accuracy and threshold, and to see the trend of it, figure 3-1 has been visualized to a line chart, as figure 3-2 shows.

As it's shown in figure 3-1 and 3-2, the best accuracy appears when θ equals 0.45, while accuracy equals 69.82%. Figure 3-2 shows that, when the threshold increases from a low-level value (i.e. 0.25), the accuracy will increase, which means the performance of the binary classifier is improved. However, if

the threshold keeps increasing, the accuracy will be reduced, hence the performance of the classifier is reduced. This is a very significant observation, since it shows us that setting threshold as 0.5, or simply choosing the largest value from the output neurons as the prediction result, could possibly not be the best solution for a dataset, and might reduce the performance of a corresponding binary classifier.

A possible reason for setting threshold as 0.45 not 0.5 might be observed from the prediction confusion matrix. Figure 3-3 shows the difference of confusion matrix of a classifier when using threshold 0.45 and 0.5.

```
When threshold = 0.50:
Confusion matrix: [50., 16.]
                  [24., 35.]
When threshold = 0.45:
Confusion matrix: [47., 17.]
                  [18., 43.]
```

Figure 3-3: Confusion matrix for threshold 0.45 and 0.50

From figure 3-3 we can observe that, when the threshold is set as 0.45, more testing data could be regarded as class 1 rather than class 0. The correct prediction of class 1 increases from 35 cases to 43 cases, while the correct prediction of class 0 decreases 3 cases. Also, the false prediction will be affected as well. The class-0 data are more likely to be predicted as class-1 data, as the wrong prediction increases from 16 cases to 17 cases. On the contrary, the class-1 data are harder to be predicted wrongly, since there are 6 less cases of wrong class-1 prediction (from 24 to 18).

The confusion matrix indicates that, if the testing dataset can be predicted more accurately by setting some threshold, which might affect the prediction process and could especially work for those dataset have imbalanced label.

Comparing to the paper published by Kim, Thomas, Sankaranarayana, Gedeon & Yoon (2015) which provided the dataset, my research direction is different. The paper is mainly focused on the dataset itself, and trying to figure out some relationship between some features inside the dataset, while my research is a binary classifier using neural network techniques. But, still there are something I can find from the paper. A very interesting thing is that, researchers of the paper will not use plenty of features as inputs for an analysis. For instance, the paper has described the relationship between search speed and fixation duration, and only focused on these two features, while I have used 25 features as inputs. This inspires me to think of if I have used too many features as inputs, including some features that has no relationship with the output attributes “size”. And this might be a reason why the binary classifier described above performs not really well (i.e. 70% of accuracy).

Another thing I have found from the paper is about the task problem I have defined for this dataset. I have defined a binary classification problem for it, however, the dataset might be more suitable to other types of topics, like regression. The paper has used some statistical techniques on the dataset to make analysis on it, since most of the features in the dataset are numerical features, and this could make the dataset more suitable for topics using numeric relative models like linear regression.

3.2 Results of the Genetic Algorithm

When applying a genetic algorithm described in section 2.5, comparing to baseline NN, the accuracy of the binary classifier is improved. The best chromosome is found in the 25^{th} generation, where the chromosome is represented as [1 1 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 1 1 0 1 1 0] with the fitness value

(average testing accuracy) 69.89%, which is higher than the baseline NN which can be described as chromosome [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1] with accuracy 67.12%, while in the 7th generation a chromosome [1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 0 1 1 1 0 1 0] has 59.78% fitness value which is much lower than the baseline NN. This performance indicates that some features in the dataset can contribute more than others.

Comparing to the first generation where all chromosomes are randomly generated, in the last generation, some DNA on chromosomes are more commonly appears as "0" or "1". For instance, in the last generation, there are 26 chromosomes with value "1" on the second DNA of the chromosome, which is 90 percent and much higher than around 50 percent as baseline if the DNA value is randomly arranged as the first generation. On the contrary, there are only 8 chromosomes with value "1" and 22 with "0" on the 24th DNA, which shows that this feature is more likely to be irrelevant to the binary classification task. The chromosomes of the last generation are shown in figure 3-4.

Chromosomes	
[1 1 0 1 0 1 0 0 0 0 0 1 1 0 0 1 0 1 0 0 0 0 1 0 0]	[0 1 0 1 0 1 1 0 0 0 0 1 1 0 0 1 1 1 1 1 0 1 1 0 0]
[0 0 1 1 0 0 1 0 0 0 1 0 0 0 0 1 1 0 0 1 1 0 0 1 0]	[1 1 1 1 1 0 1 1 1 0 1 1 0 1 0 0 1 1 0 1 0 0 0 1 1]
[0 1 0 1 0 0 0 0 1 0 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1]	[0 1 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 1 0 0 0]
[0 1 0 1 0 1 1 0 0 0 1 1 1 1 0 1 1 1 0 1 0 0 0 0 0]	[1 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 1 0 0 1 1 0 0 0 0]
[1 1 0 1 1 0 1 1 1 0 1 1 0 0 1 0 1 1 0 1 0 0 0 0 1]	[1 1 0 1 0 0 0 1 0 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 0]
[0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0]	[0 1 0 1 0 0 0 0 0 1 1 0 0 0 1 1 1 1 0 1 1 1 1 0 0]
[0 1 1 1 1 0 1 1 1 0 1 1 0 1 0 0 1 0 0 1 1 0 0 0 1]	[1 1 0 1 0 0 1 1 1 0 1 1 1 0 0 0 1 1 0 1 1 0 1 1 1]
[1 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0]	[0 0 1 1 1 1 1 1 1 0 0 0 1 1 1 0 1 1 0 0 1 0 0 0 1 1]
[0 1 0 1 0 1 0 0 1 1 1 0 1 0 0 0 1 0 0 1 1 0 0 0 1]	[0 1 0 1 1 1 1 1 1 0 1 0 0 0 0 1 1 0 0 1 1 1 0 0 1]
[1 1 0 1 1 0 1 1 1 0 1 0 0 1 0 0 1 1 0 1 0 0 0 0 1]	[1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 1 1 1 0 1 0 1 1 0 0]
[0 1 0 1 0 1 0 0 0 0 1 1 0 0 0 1 1 1 0 1 1 0 1 0 0]	[1 1 0 1 1 0 0 1 1 0 1 1 0 1 0 0 1 1 0 1 1 0 0 0 1]
[1 1 1 1 0 1 1 0 1 0 0 1 1 1 1 0 1 1 0 1 1 1 1 0 0]	[0 1 0 1 0 0 0 1 0 1 1 0 0 1 0 1 1 1 1 0 0 1 0 1 1]
[1 1 0 1 0 1 1 1 0 1 1 1 1 0 0 0 1 1 0 1 1 1 0 0 0]	[1 1 0 1 0 1 0 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 0 0]
[1 1 0 1 0 1 0 0 0 1 0 1 1 0 0 1 1 1 0 1 0 0 0 0 0]	[0 1 0 1 1 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0 1]
[0 1 1 1 1 0 1 1 1 0 1 1 0 0 0 0 1 1 0 1 1 0 0 1 1]	[0 0 1 1 0 0 1 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 0 0 1]
[1 1 0 1 0 0 1 1 1 1 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1]	[1 1 0 1 0 0 1 1 0 0 0 0 0 1 0 1 1 1 1 1 0 0 0 1 0]
[1 1 0 1 0 1 0 0 0 1 0 1 1 0 0 1 1 1 1 1 1 0 0 0 0]	[1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 0 0 1 1 0 1 0 0 0 0]
[1 0 0 1 1 0 1 1 0 0 1 0 0 0 0 0 1 1 0 1 1 0 0 0 0]	[0 1 1 1 0 1 1 1 1 0 1 0 0 0 0 1 1 1 0 1 1 0 0 0 1]
[1 1 0 1 0 0 0 0 0 1 1 0 0 0 0 1 1 1 0 1 0 1 1 1 0]	[1 1 0 1 0 0 1 1 1 0 1 0 0 1 0 0 1 1 0 1 0 0 0 0 0]
[1 1 0 1 0 1 0 0 0 1 0 1 1 0 0 1 1 1 1 1 1 0 0 0 0]	[1 1 0 1 0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 1 0 0 1 0 0]

Figure 3-4: the chromosomes of the last generation

It can be observed that: the 2nd, 4th, 17th and 18th DNAs are more likely to be value "1", which indicates that corresponding features are more likely to contribute to the classifier, while the 15th, 22nd, 23rd and 24th DNAs are more likely to be value "0" and irrelevant to the classification task. Thus, features should be selected so as to build a better classifier.

4. Conclusion and Future Work

In this paper, I have solved a binary classification problem using neural network and threshold adjustment

techniques. The implementation is based on Python/Pytorch coding, and is described above. However, the neural network is still not good enough for the given classification problem, since the accuracy is not good enough for a binary classification task. There are some limitations of the baseline network, and the most significant one is about I have used too many features as inputs, which has been discussed in section 3. To figure out which feature is relative and which is not, a genetic algorithm-based feature selection technique is used, but different to what Li et al. (2018) and Ang et al. (2016) has demonstrated on their paper. The genetic algorithm I have implemented is still not perfect for the feature selection task, since the classifier accuracy only increase about 2 percent. This is probably because of my genetic settings, and having larger population with more generations could provide a better convergent feature selection result giving more accurate predicting result. Also, as described above, I have used ranked-based selection method to choose one parent and randomly chose another one, which might be improved with other selection methods like Elitism selection or Hall Of Fame selection.

Another problem is also discussed in section 3, about if the dataset is suitable for a classification task using neural network techniques. Some future work can be done to determine if this dataset is more suitable for a regression task. In addition, the inner implementation of the NN might be not good enough for this task. The neural network is trained with Stochastic Gradient Descent (SGD) optimizer, and use the cross entropy as the loss function, but there is less evidence can indicate that they are the best choice for this task. Thus, to improve the performance of the NN, some other optimizer can be used, while the "BCEloss()" loss function can be tried on the NN, since it is specialized designed for binary classification task (Torch Contributors 2019).

There are some more techniques can be applied and used on the implementation described in this paper, and more researches should be done on the classifier to improve its performance in the future.

5. References

Ang, J.C., Mirzal, A., Haron, H. & Hamed, H.N.A. 2016, "Supervised, Unsupervised, and Semi-Supervised Feature Selection: A Review on Gene Selection", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 13, no. 5, pp. 971-989.

Hawkins, D.M. 2004, "The Problem of Overfitting", *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 1, pp. 1-12.

Kim, J., Thomas, P., Sankaranarayana, R., Gedeon, T. & Yoon, H. 2015, "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*, vol. 66, no. 3, pp. 526-544.

Kogan, I. 1991, "Speculative experiment with neural networks on separation and scoring in financial applications," *IEEE International Joint Conference on Neural Networks*, vol. 1, pp. 775-776.

Lewis, H.G. & Brown, M. 2001, "A generalized confusion matrix for assessing area estimates from

remotely sensed data", *International Journal of Remote Sensing*, vol. 22, no. 16, pp. 3223-3235.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R., Tang, J. & Liu, H. 2018, "Feature Selection: A Data Perspective", *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1-45.

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

Reeves, C. R., Rowe, J. E., & SpringerLink (Online service). (2003). Genetic algorithms: Principles and perspectives : *A guide to GA theory*. Boston, MA: Kluwer Academic Publishers. doi:10.1007/b101880

Torch Contributors 2019, ‘PyTorch docs’, viewed 1 May 2020, https://pytorch.org/docs/stable/nn.html?highlight=bceloss#torch.nn.BCELoss

Yang, J., & Honavar, V. (1998). Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems and their Applications, 13*(2), 44-49. doi:10.1109/5254.671091