Classification to the Pagination and Scrolling in Mobile Web Search and Bidirectional Neural Network

Ruidong Hua
Research School of Computer Science
Australian National University
Acton ACT 2601 Australia
U6688534@anu.edu.au

Abstract. Based on the data from the study done by Jaewon et al, I try to use the neural network technology to determine which type of control the user uses by using the measured time consumption data of the actual experiment. This work should be done in conjunction with statistical analysis of the data, which can help to verify that the control mode is strongly related to the user's performance and improve the use and explanation of the result statistical analysis. Four methods are explained and implemented to achieve the classification task, unidirectional, bidirectional learning, LSTM and evolving neural network to training my net. The result is beyond my expectation because the network's ability to categorize is not as good as expected, which cannot reflect that the measurement data can effectively identify the control type. Thus, I further analyzed the possible reasons.

Keywords: Pagination and Scrolling, Bidirectional Neural Network, Back-Propagation, LSTM, NEAT

1. Introduction

According to de description of Jaewon et al (2016), choosing a different page-turning method on a touch device to search for a target page may have different effects on search performance[1]. They measured the search time from 24 subjects who using two types of control and research data by computing mean and applying analysis of variance. Through statistical analysis of the test data, the authors found that the search performance of horizontal page turning was better than that of vertical page turning. However, the author is also careful to realize that due to the uncontrollable differences of experimental data, it is not appropriate to assert which control method is better simply by statistical data. In fact, this consideration is necessary. Because the measurement data of the experiment are all from the measurement of a small number of participants, the reliability and stability of data cannot be guaranteed. Different subjects may perform differently on different tasks for personal reasons. For example, some subjects may be able to get into a good search state faster while others more slowly. On top of this, the authors also point out that the time spent on the scroll action itself is the key effect on the difference for the search speed between two control type. Whether this difference is determined by the type of control itself requires more careful consideration. Thus, my task will be to explore whether the characteristics reflected in the measured data are necessarily related to different types of control.

Because the relationship between the measurement data and the classification of control types is not obvious, I plan to train a Neural network to use the measurements to determine whether the tester is using a horizontal page flip or a vertical page flip. If there is a performance difference between the two control types in use, the classification model should show good classification accuracy, which means that different control methods can result in different search performance.

In the process of training the neural network, classic back propagation neural network is the basic method I choose. However, in addition to unidirectional propagation, I also try to implement the bidirectional propagation presented by Nejad and Gedeon (1995)[2]. Because the objective of my classification model is to determine if there is a possible strong association between the test data and the control type, so a two-way network helps to more fully explore such a connection that may not be obvious. Besides, Pontes-Filho and Liwicki state that no-direction network is benefit to the robustness to the random noise and adversarial examples through being a classifier and generator[3]. So, I try to take advantage of the no-direction network during my training to implement bidirectional learning to improve the recognition ability of classifier. On top of this, long short-term memory recurrent neural network (LSTM-RNN) by pytorch. Finally, an evolving neural network is implement based on the study about topology and weight evolving artificial neural network [4], which is a model with network optimization using genetic algorithm.

In the following section, I simply preprocessed the raw data and introduce the realization methods and processes of four networks respectively. By discussing the results of classification model, I present the results of correlation identification. Finally, I summarize the full text and suggest future research possibilities.

2. Method

2.1. Data description

The original data comes from the measurement to 24 subjects about time to make different decisions in web search.
The dataset has 15 columns and 288 rows without missing value. An important point is that the number of the two class of the dataset is the same.

2.2. Data preprocessing

Through the simple observation of the original data, I found that there was a certain degree of redundancy in the original data. There are two sets of data that I judged to be duplicates. Time to first click, Time to right click, and Task completion duration belong to set 1, and Task start, 1st click, right click and Task end belong to set 2. The data from set 1 can be computed by subtracting the data of dataset two. In other words, set 2 simply records the contents of set 1 at an exact point in time, so I deleted the feature columns in set 2. Besides, due to the definition of accuracy in the original experiment, I think there is also a duplication of information between the 'accuracy' column and the 'Time with wrong WebDoc' column. Obviously, the column 'Time with wrong WebDoc' is not zero only if column 'Accuracy' is not 0, so I delete the column 'Accuracy'. As for column 'Satisfaction', it is not a completely objective measurement, which is a measurement that is based on the subjective evaluation of the subject. Its reliability is questionable, which is the reason why I delete it. The final column I delete is 'Scroll', the content of this column is not clear and I am not sure what does it mean.
After the delete operation is done, I also try to add some other feature into the dataset. I calculated two ratios as the new feature column. The first one is 'Time to first click'/ 'Task completion duration', and the second one is 'Total time on SERPs' / 'Task completion duration'. In my opinion, the content of 'Time to first click' and ''Total time on SERPs' which is related to the time spend on SERPs may be affected by the way the page is turned, while reading the web itself is not affected by the control type.
Finally, I get 9 columns as feature and 'Type' column as label. Then, I use the PCA function provided by sklearn package to get two low dimensional feature data with four and two dimensions respectively.

2.3. Classic Back Propagation (BP) Neural Network

I used four - dimensional feature data to train and test the BP neural network.
I tried to train two neural networks (NN) by pytorch which have one hidden layer and two hidden layers respectively to compare their performance. The loss function I choose is the cross-entropy function which is widely used in classification task and use adam optimizer to achieve the parameter update [4].
As for the NN with two hidden layers, I thought ahead of time about the risk of overfitting, so I add dropout operation to the net. According to the study done by Sricastava and Hinton et al. (2014), I set the parameter of dropout layer to 0.5 because the number of randomly generated network structures is the most when the dropout rate is 0.5 [5]. However, the result of comparison was disappointing. Although the difference in model accuracy between the training set and the test set is reduced, the accuracy on the verification set does not increase significantly while the accuracy on the training set reduce.
As for activation function, I compare the sigmoid and the RELU function. Under the detection of k-fold cross validation, the model with sigmoid is slightly more accurate on the test set than the model with RELU (Accuracy (sigmoid) – Accuracy (RELU)=0.05). However, RELU converges much faster than sigmoid.

2.4. Bidirectional Neural Network (BDNN)

As for an artificial neural network, if the shape of input and output is the same, this network can be trained in both reverse and forward directions. According to the study about the Slender-Tufted L5A Pyramidal Neurons done by Grewe, Bonnan and Frick(2010), real neurons are able to modulate the plasticity of the system through back-propagation action potentials[6], which suggests that artificial neural networks can also improve system performance by attempting bidirectional propagation.
Based on the Hebbian theory states, the propagation in reverse direction can be regarded as an inverse operation of forward propagation. We can think of the process of forward propagation as a composite function:

$$Y = f(w \times x)$$

Y is the output, x is input, w is weight matrices and f is the activation function. It is easy to get the composite function of propagation in reverse direction:

$$X = f(w^T \times Y)$$

This equation expresses that the weight matrix can be shared during bidirectional propagation in the same network. Thus, with a simple transpose adjustment to the weight matrix, the parameters of the whole network can be updated continuously through implementing the error back propagation algorithm in two directions. Figure 4 shows the topology of a bidirectional neural network.

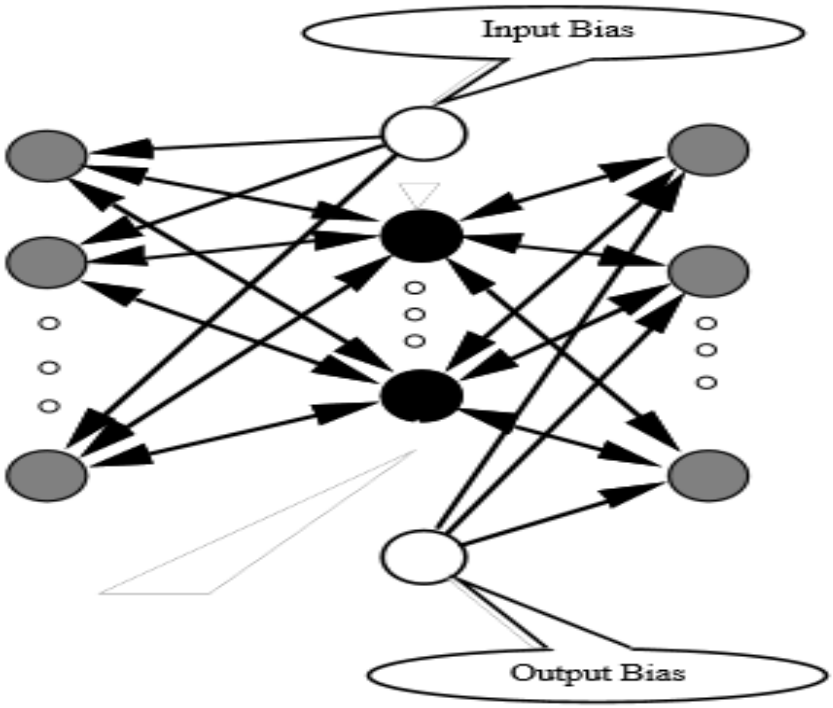

Fig. 1. topology of a bidirectional neural network (From Bidirectional neural networks and class prototypes, A.F.Nejad and T.D.Gedeon.(2016))

In the concrete implement, there is a little trouble for the neural network with two hidden. Due the number of neurons in the two hidden layers is not the same, the forward and reverse routes of propagation are not identical. Besides, because the essence of backward propagation is the generator that restores the input data rather than a simple classifier, so the choice of loss function is also different from the choice of the forward propagation.
In fact, I build two nets by pytorch which are net_forward and net_reverse. The structure of the net_forward is similar to that of the back-propagation network in 2.2, but the number of neurons in the input layer is 2. The structure of the net_reverse is the mirror image of the net_forward but use L1 loss function instead of cross-entropy function.
During the training, I train the net_forward in the first 50 epochs, and then, read the network parameters at this time and assign the weight matrix with the transpose adjustment to the corresponding layer of the net_reverse. For every 50 epochs trained, switch direction in the same way until the end of the training.
Again, I used k-fold cross validation to test and calculate the average accuracy.

2.5. Long short-Term Memory Recurrent Neural Network (LSTM-RNN)

Because the time measure attributes in the data set have inherent precedence relationship, a simple LSTM-RNN model may be useful to achieve this classification task. In order to implement the LSTM-RNN model for this dataset, I am going to make two assumptions: 1). each input feature may affect the value of the next input feature. 2)LSTM's forgetting gate can effectively forget useless information to help the model better explore possible correlations between inputs.
Using pytorch's package can quickly implement the construction of LSTM-RNN. After adjusting the parameters, I set the number of hidden nodes in the network to 19 and the number of hidden layers to 1. During the training, I used the method of batch training. After several parameter attempts, I set batch_size with 16.

2.6. Evolving Neural Network Based on NEAT

The traditional BP neural network use the back-propagation error and gradient descent method to update the network parameters to optimize the network. Although this method of gradient descent has a fast convergence rate and good performance in supervised learning tasks, it is easy to fall into the local optimum and cannot make the model reach the optimum. Unlike this artificial neural network, neural evolution has a structure closer to biological neural network which

uses genetic algorithm to evolve the network to reach the optimum [7]. Because gradients are not mentioned in the evolutionary process, neural evolution has a good performance in avoiding local optimal problems.
The Topology and Weight Evolving Artificial Neural Network designed by Sranley, K.O. and Miikkulainen, R. (2002) propose a method to intersect different structures to change the topology and parameters of the network during evolution. For encode, in addition to the use of traditional graphic coding techniques, NEAT also add a new element named innovation number for connection gene, which is used to encode links directly to facilitate subsequent crossover and mutation operations. In order to avoid a large number of structural crossing leading to the formation of homogeneous networks, NEAT divides similar structures into the same population and carries out cross evolution among the populations to ensure the innovation of network structure. At the same time, in order to prevent a network structure from generating a network with the same structure but innovation number after multiple mutations, NEAT maintains a list during the evolution process to record the update of the current generation. For the crossover process, NEAT compares the innovation number of both parents, innovation number owned by both parents will be randomly selected to be passed on to the next generation, while innovation number owned by only one parent will be directly passed on to the next generation[8].
In my implementation, I set fitness function to:

$$fittness = 1.0 - \text{CrossEntropyLoss}(\text{Pred}_Y, Y)$$

Pred_Y is the predicted result from networks and Y is the actual class. Obviously, the higher the value of fitness, the better the quality of the network model. Besides, the max fitness threshold is set with 0.99 and genetic 300 generations. The mutation probability of both nodes and connections is set to 0.5.

3. Result and Discussion

Except the evolving neural network model, the rest of models use a 12-fold cross-validation test accuracy. Because the size of dataset is 288, so the size of test set is 24.

3.1. Performance of BP Network

In this section, I will show the performance of the above different models on the measurement accuracy.
Table 1 present the accuracy of the BP networks on traning set and test set

	1	2	3	4	5	6	7	8	9	10	11	12	Mean
Train set	0.6629	0.6667	0.6629	0.6515	0.6553	0.6853	0.6553	0.6553	0.4515	0.6629	0.6705	0.6402	0.6600
Test set	0.7917	0.3750	0.5000	0.4583	0.6250	0.5000	0.3333	0.3750	0.5833	0.5417	0.5417	0.7083	0.5278

Table 1. The accuracy for net with two hidden layers with dropout in unidirectional propagation

During the ten-fold cross validation process, I output the confusion matrix of test set results in each test. Table 2 and Table 3 present the confusion matrixs of validation 1 and validation 3.

	Predicted postive	Predicted negative
Actual postive	9	2
Actual negative	3	10

Table 2. The confusion matrix of validation 1

	Predicted postive	Predicted negative
Actual postive	6	6
Actual negative	6	6

Table 3. The confusion matrix of validation 6

It can be seen from the two confusion matrices that the model makes corresponding judgments for each class, and there is no case where only one category is output. According to table 1, the classification ability of the models trained under

different test sets is quite different. Obviously, the model has a certain degree of overfitting the average performance of the model is quite different from that of the training set.

3.2. Performance of BDNN

	1	2	3	4	5	6	7	8	9	10	11	12	Mean
Train set	0.4886	0.5114	0.5000	0.5492	0.5076	0.5189	0.5189	0.5000	0.5000	0.5152	0.5152	0.4924	0.5098
Test set	0.4167	0.5417	0.5000	0.4583	0.4583	0.4167	0.4167	0.4583	0.7917	0.4167	0.5833	0.5000	0.4965

Table 4. The accuracy for net with two hidden layers with dropout in bidirectional propagation

BDNN performed slightly worse than BP Network, while there is no overfitting of BDNN. However, after analyzing the obfuscation matrix, I found that there was only one type of model output in the ten cross validation,which may be caused by the error caused by the bidirectional update of the weight matrix. Obviously, the BDNN model itself is not very good at fitting this dataset.

3.3. Performance of LSTM-RNN

Table 5 present the accuracy of the LSTM-RNN on traning set and test set.

	1	2	3	4	5	6	7	8	9	10	11	12	Mean
Train set	0.7348	0.7348	0.7841	0.8068	0.7462	0.7841	0.7424	0.7462	0.7917	0.7917	0.8409	0.7197	0.7686
Test set	0.4167	0.5833	0.3750	0.7083	0.6250	0.4583	0.5417	0.3333	0.5000	0.5000	0.5000	0.3750	0.4931

Table 5. The accuracy of LSTM-RNN in train set and test set

During the 12-fold validation process, I also output the confusion matrix for each test. Although the average accuracy of the model is around 0.5, according to the confusion matrix, the model does not have a single output. The performance of this model is not much different from that of BP neural network, which indicates that there is no decisive correlation between the input data.

3.4. Performance of Evolving Neural Network

Due to the limitations of the equipment, the running speed of the model was slow, so I only conducted three separate tests.

	1	2	3	Mean
Accuracy on test set	0.4828	0.5172	0.4482	0.4827

Table 6. The accuracy of evolving neural network in testset

According to table 6, the accuracy of evolving neural network is also around 0.5. Its confusion matrixs show that there is no single label output in this model. Because of the limitations of the device, the genetic generation cannot be expanded indefinitely.

3.5. Comprehensive analysis

Surprisingly, regardless of the model, the classification accuracy on the test set was almost distributed around 0.5. For a dichotomous task, an accuracy of approximately 0.5 indicates that the feature is almost not directly related to the label. However, according to their statistics[1], the search speed of using pagination is faster than that of using scrolling. I made the following assumptions about the difference:

1.1 The size of dataset uesd to traing the neural network is too small. The effect of outliers on model training is difficult to remove with a small amount of data. Besides, it is too hard for network to learn about the implicit connection between features and labels based on such a small dataset.

1.2 The model itself is not perfectly designed. I cannot guarantee that the classifier I design is optimal. Too simple models or inappropriate parameter Settings can lead to low accuracy

1.3 There is a real lack of reliable dependencies between features and labels.

Among the four models, the performance of BDNN is the most disappointing, because it does not fit the training set well and the single-label output occurs, there are two possible reasons for this poor performance:

1.1 The bidirectional training method of BDNN is not perfect, and the optimal solution may be missed in the process of bidirectional learning.

1.2 There is no bidirectional correlation between the data set input and the label itself, and the reverse mapping from the label to the input may lead to greater errors in the network.

Suppose 300 generations of inheritance is enough, the performance of evolving neural network indicates that the local optimal problem does not affect the performance of model using gradient training because they all have an accuracy of around 0.5.

Without the occurrence of single label output, the binary classification accuracy of both simple neural network model and complex deep learning model is about 0.5, which is almost no different from the performance of the baseline model of the binary classification. Therefore, I have come to the preliminary conclusion that the input characteristics do not play a decisive role in determining the classification label. Although there are differences in search speed based on statistics using different page-turning methods, due to individual differences and limited experimental conditions, there is no definite conclusion on the merits of horizontal control and vertical rolling[1].

4. Conclusion

In conclusion, I used pytorch to build several neural networks to achieve a classification task to determine whether the user is using pagination or scrolling for web searches when using the touch devices. Through my research, I found that it is too hard to class control type of page turning through the actual test data of the subjects. Thus, we cannot state that type of page-turning will necessarily affect the efficiency of web search. At the same time, I also tried to explore the implementation and effects of BDDN. I designed a simple training method to achieve bidirectional learning based on the study of Nejad and Gedeon[2], and analyzed its influence to the network training. What is certain is that BDDN is not beneficial to all networks in accuracy and its impact on the stability of the model also requires analysis using much larger amounts of data. On top of these, LSTM-RNN and Neuro evolution were also implemented in this study to explore the deep learning model and neural network with genetic algorithm optimization. In the future, it is very valuable to continue to explore the appropriate BDNN network structure and bidirectional learning process. Besides, it is also meaningful to explore more use of NEAT with suitable equipment, such as speeding up reinforcement learning model.

References

1. Kim, J., Thomas, P., Sankaranarayana, R., Gedeon, T. & Yoon, H. J.: Pagination versus scrolling in mobile web search. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management , pp. 751-760(2016)
2. Nejad, A.F. & Gedeon, T.D.: Bidirectional neural networks and class prototypes. Proceedings of ICNN'95-International Conference on Nerural Networks(1995)
3. Pontes-Filho,S. & Liwicki, M.: Bidirectional learning for robust neural networks. 2019 International Joint Conference on Neural Networks(IJCNN) （2019）
4. Kingma, D.P. & Ba, J.： Adam: A method for stochastic optimization. In the 3rd International Conference for Learning Representation, San Ciego (2015)
5. Sirvstava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R.： Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(56), pp. 1929-1958 (2016)
6. Grewe, B.F., Bonna, A. & Frick, A.: Back-propagation of pyhsiological action potential output in dendrites of slender-tufted L5A pyramidal neurons. Frontiers in Cellular Neuroscience, 4 (2010)
7. Angeline, P.J., Saunders, G.M. & Pollack J.B.: An evolutionary algorithm that constructs recurrent neural networks. IEEE transactions on neural networks, 5(1) (1994)
8. Stanley, K.O. and Mikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary computation, 10(2), pp. 99-127(2002)