Predicting the phone screen-size from users web-search behaviour

Ferdinand Wittmann

Australian National University, Canberra ACT 0200, Australia

Abstract. This study describes the design of a neural network to accurately predict a person's phone size from his web-search behavior. Initially, different network structures are fitted to potential feature sets to derive an adequate network structure. Afterward, a genetic algorithm is used to perform feature selection. The results show that our neural network can predict a person's phone size with an accuracy of around 50%. Lastly, a network reduction technique was applied to the neural network in order to derive networks of lower complexity. The reduction of the network led to a high loss in accuracy, with only a small decrease in complexity.

Keywords: Neural Networks · Network Reduction · Screen-size Prediction · Genetic Algorithm · Feature Selection

1 Introduction

In this paper, we describe the design of a classifier for predicting a user's phone size from his web-search behavior. To predict the phone size accurately, a fully connected neural network with two hidden layers was chosen. The model is trained on a dataset from the study "Understanding Eye Movements on Mobile Devices for Better Presentation of Search Results"(UEM) that was conducted in 2015 at the Australian National University [2]. The data consists of 42 features encoding the users' behavior while using a web search engine. In order to reduce the feature size, we used a genetic algorithm for feature selection. This neural network could then be embedded into websites to predict the users' phone size. Since computing power varies greatly between different smartphones, another goal of the paper was to test a technique for post-training network compression. In this matter, the trained network could be reduced in complexity depending on the phone's computing power. For the model compression, we used a network reduction technique introduced by Gedeon & Harris in 1997, that eliminates hidden units that a similar or opposite aligned in the hidden layers vector space [1].

2 Pre-processing of the Dataset

Together with collecting the data, the authors of the paper UEM also did some statistical analysis on the dataset. The results of this analysis can be seen in

fig 1. From this correlation analysis, one can see that 11 from the original 42 features are in statistically significant relation to the screen size. Do do a first analysis of the network we now create 3 different datasets. The first dataset is consist of all features with a significanz level lower than 0.01, the second dataset of all features with a significanz level lower than 0.0 and the third of all features. For this analysis, the authors assumed that some of the features are Poisson

		Mean values			Statistics			
		L	M	S	*p*-value	L	M	S
Search performance								
Search speed	Time to first click [s]	7.70	10.47	9.12	0.195			
	Task completion duration [s]	20.89	24.79	23.08	0.655			
Search accuracy	Correct answer rate [%]	94.44	98.15	94.44	0.589			
Search behavior								
Fixation duration on SERP	Per task [s]	3.97	5.60	5.53	0.087			
	Per link [s]	2.16	1.89	2.49	*	ab	a	b
Clicks	Ranks	1.39	1.52	1.46	0.697			
Scanpath	Minimal scanpath	2.06	2.76	2.26	**	a	b	a
	Compressed sequence	3.33	5.50	4.35	**	a	b	ab
	Compressed minus minimal	1.28	2.74	2.09	*	a	b	ab
Scanning direction	Complete rate [%]	96.30	94.44	100	**	a	a	b
	Linear rate [%]	46.30	31.48	57.41	*	ab	a	b
	Strictly linear rate [%]	11.11	1.85	9.26	0.087			
	Linear/ID rate [%]	81.48	55.56	79.63	**	a	b	a
	Strictly linear/ID rate [%]	46.30	25.93	31.48	*	a	b	a
Skip and regression	Skip [%]	14.81	22.22	7.41	0.087			
	Regression [%]	53.70	74.07	68.52	*	a	b	ab
Scroll	Scrolled rate [%]	3.70	20.37	35.19	***	a	b	b
Trackback	Count	1.07	1.91	1.28	***	a	b	a
Search satisfaction	7-point Likert scale	5.24	4.91	4.20	***	a	a	b

*Significant at 0.05 level. **Significant at 0.01 level. ***Significant at 0.001 level. *Note*. SERP denotes search engine results page, and L, M, and S denotes large, medium, and small, respectively.
Note. Labels a and b indicate the type of result, "a" type is significantly different from "b", but not different to "ab".

Fig. 1. Statistical analysis of the dataset [2]

distributed (E.g., Minimal Scnapath Length) [2]. This underlying information about the input data can, therefore, be used to transform the corresponding features, before training the model.

$$mean(X) = \lambda, p\_transform(x) = \begin{cases} \frac{x!}{\lambda^x e^{-\lambda}} & \text{if } x \geq \lambda \\ \frac{-x!}{\lambda^x e^{-\lambda}} & \text{otherwise} \end{cases} \tag{1}$$

The applied transformation can be seen in equation 1. Equation 1 applies the inverse probability mass function for Poisson distributed variables to each data point. This results in data points being mapped to there inverse probability and will increase the impact of outliers. Since we assign data points to their corresponding probability values, data points smaller and bigger than the mean can be mapped onto the same value, and we save this information by giving values before the mean a negative sign. After transforming the corresponding features, we apply min-max normalization to scale all features between 0 and 1 [6]. Furthermore, categorical data was encoded using one-hot encoding [7].

3 The Network

The network was derived in two steps. In the first step, different network architectures were compared on the three feature sets that were chosen based on the correlation analysis. The best network architecture is then used for a genetic feature selection algorithm (GFS). The goal of the GFS is to increase the neural network's accuracy by excluding features that introduce noise into the data.
The initial network architecture differed in having up to 3 hidden layers and either 5,10 or 20 hidden units per layer. As activation function, either Tanh or Relu were chosen [8]. As mentioned earlier, the network architecture was tested on three different feature sets. The output layer had a size of 3, corresponding to the labels in the dataset. The dataset used three different phone sizes labeled as large (L), medium (M), or small (S). On the one hand, the Tanh network was trained using the adam optimizer with adaptive learning rates. On the other hand, the RelU network was trained with stochastic gradient descent with momentum. Momentum uses "Exponentially weighed averages that can provide [us] a better estimate which is closer to the actual derivate than our noisy calculations" [4]. The adam optimizer combines the popular RMSprop optimizer with stochastic gradient descent with momentum and has shown to successfully train networks [3].
After the network architecture is decided on, we apply the GFS. The GFS tries to derive the best feature set in a number of iterations that are called generations. Each generation has as input a population consisting of a set of binary vectors, the chromosomes. The chromosomes indicate for each feature if it is activated or not. Each chromosome fitness is tested by inputting the activated features into the predefined neural network and evaluating the network's accuracy (fitness). Now a new population for the next generation has to be created. Here we sample the future population from the old population, where the old chromosomes get selected with the probability function in equation 2. In the next step, we apply with some probability 2-point crossover and randomly mutate each bit at a certain likelihood. After letting this algorithm run for several generations, we were hoping to find the best performing feature set.

$$p(c) = \frac{fitness(c) - min}{\sum_{x \in P} fitness(x) - min}, min = \underset{x \in P}{\operatorname{argmin}} fitness(x) \tag{2}$$

4 Network Selection

When training the network, we saw that performance would not increase after adding a third layer, and therefore we decided on a dual-layer network. We then trained the dual-layer network for the different activation techniques and with either 5,10 or 20 hidden units per layer and for each feature set. In order to achieve accurate results, we folded the training set into multiple parts. This technique is known as k-fold cross validation and we used k=10 that is commonly favoured in neural network research [9]. After that, the network is trained ten times, with each time having a different fold as the testing set. Additionally, we

ran this setup 20 times and took the mean of the test accuracy as a performance measure. For both relu and tanh activation a learning rate of 0.01 was used and the network was trained in 600 iterations (epochs).

The results are summarized in Table 2. We can directly see that feature set

Table 1. Performance Evaluation of the Network

Testing Accuracy in %	Tanh S	Tanh M	Tanh L	**Tanh**	RelU S	RelU M	RelU L	**RelU**
Hidden Units: 5 Feature Set: 1	50	35	65	50	40	31	64	45
Hidden Units: 10 Feature Set: 1	57	33	64	52	57	33	64	52
Hidden Units: 20 Feature Set: 1	52	37	70	53	50	42	65	52
Hidden Units: 5 Feature Set: 2	47	42	56	48	44	44	54	47
Hidden Units: 10 Feature Set: 2	47	42	45	45	46	44	48	47
Hidden Units: 20 Feature Set: 2	47	42	45	45	45	42	53	47
Hidden Units: 5 Feature Set: 3	47	42	44	44	-	-	-	-
Hidden Units: 10 Feature Set: 3	51	43	50	48	-	-	-	-
Hidden Units: 20 Feature Set: 3	48	42	42	44	-	-	-	-

1 with 20-hidden units and tanh activation in each layer outperformed all the other sets with an accuracy of 53%. Increasingly worse performance for the bigger feature sets indicates that adding more features to the training set leads to over-fitting the data [5]. That the bigger feature sets led to overfitting was also indicated from the training, where the feature set 1 achieved a training accuracy of up to 62%, feature set 2 up 67%, and feature set 3 up to 100%. One has to note here that our testing accuracy achieved relatively poor results, with only being 20% better than a random classifier. We can justify this bad performance when looking again at the training accuracy. For feature set 1, we could not achieve better training accuracy than 62%, and if we compare our testing accuracy to this threshold, one can see that our network performed relatively well. Lastly, we can see for which screen sizes our prediction works the best. The network classifies 70% of the large screens correctly, 50% of the medium-sized screens, and 30% of the small screens. These results are the same as the authors of the original study concluded. In figure 1, the middle column of the statistics show that for all of the features, the medium-sized screen is significantly different two the other screen sizes individually, but not to them combined. For the large and small screen, significant relations two the other screen sizes combined exist. Our results indicate the same by classifying the larger and smaller screen size more accurately. Lastly training the network with the introduced Poisson transformation increased the network accuracy from 45% to 53%. Concluding this analysis, we decided to use the best performing network for the fitness evaluation of the genetic algorithm. The selected network uses tanh activation with two hidden layers of 20 hidden units and incorporates the poisson transformed features.

5 GFS Evaluation

The GFS was initialized with the full feature set (3) concatenated with the Poisson transformed variables. For evaluation of the GFS, we split our data further into validation and training data (validation: 0.2, training: 0.8). This step is advertised in neural network research to get accurate results and prevent overfitting of the model. In figure 2., we can see the testing accuracy of each chromosome in the population at the first, fifth, and tenth generation. We can see clearly that the testing accuracy increases wit from generation to generation. On the other hand, as can be seen in figure 3., the validation accuracy did not consistently increase, indicating that our model overfits the data. Therefore we introduced to the adam optimizer a weight decay of 10^-3. This penalizes weights of high magnitude [11]. After this alternation of the initial network structure, we were able to achieve better validation accuracy, while our testing accuracy decrease around 10%, showing again that we initially overfitted the data. Lastly, we tested different mutation rates shown in table 3. The best validation accuracy was achieved for a network with population size 10 and 16 Generations. The mutation rate was 0.1 for the first 8 Generations and 0.001 for the last 8 Generations. This leads to a first step where we value exploration of the solution space more and then switch to fine-tuning of the feature selector. On the one hand, we were able to reduce the feature set from 47 to 26 features with the GFS. On the other hand, we had the results have shown to perform 4% worse than the feature set 1. We believe that this loss in accuracy comes from further reducing the dataset by introducing the validation set. This has an exceptionally high impact in this case since our dataset with only 162 entries is very small.

Table 3: GFS results

Mutation Rate	Validation Accuracy in %
0.001	44
First Half: 0.1 - Second Half: 0.001	48
0.1	43

6 Network Compression

In this section, we are going to analyze Gedeon Harris network reduction technique introduced in 1997 [1]. The strength of this technique is that it compresses the network post-training. This fits our network in the sense that it is designed to be deployed on smartphones with different computing capabilities. The compression works as follows [1]:

- Create a vector for each hidden-unit containing the result of the activation unit for each training pattern (training data-point)
- Calculate the angle between all hidden unit vectors
- If the angle for two vectors is less than a $treshold$ or more than $180-treshold$ they are similar

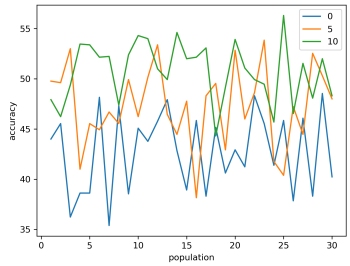

Fig. 2. GFS without weight decay, for 3 different Generations(0,5,10) (Mutation: 0.1-0.001, Crossover: 0.5)

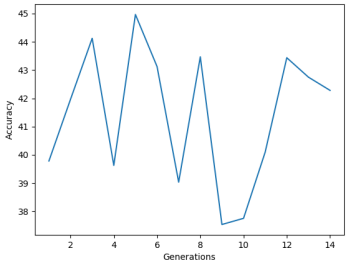

Fig. 3. Validation accuracy for GFS without weight decay (Mutation: 0.1-0.001, Crossover: 0.5)

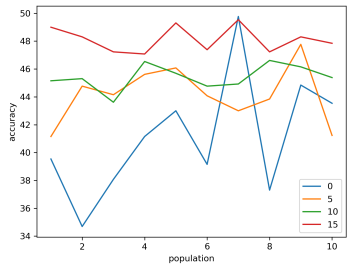

Fig. 4. GFS with weight decay (Mutation: 0.1-0.001, Crossover: 0.5)

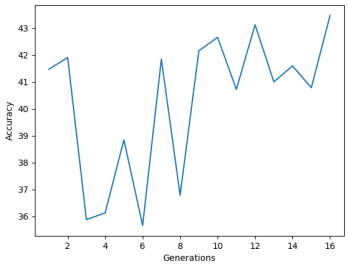

Fig. 5. Validation accuracy of GFS with weight decay (Mutation: 0.1-0.001, Crossover: 0.5)

– For every pair of similar vectors delete one of the corresponding hidden units and copy its weight vector onto the other one

We tested this technique together with the Tanh activation network. The Tanh activation unit maps its output between -1 and 1, and therefore also our angle range is between 180 deg and 0 deg, and we can directly apply the algorithm. The main benefit of this algorithm is that "This produces a network with one fewer unit which requires no further training" [1]. This means we can reduce the network to multiple different sizes for different devices of different computing power with a single trained network.

7 Evaluation Network Compression

We evaluated the network compression similarly as we assessed the initial network. We train a network once for ten different folds, then randomly shuffle the data and repeat the first step 20 times. The result is the achieved average accuracy over all iterations. Additionally, after every training, we compress the network with the described technique and evaluate how many hidden units were deleted and how our accuracy changed. We only applied the reduction to the first layer of the network to simplify the experiment. Since our network is a simple feed-forward network, the number of multiplications can be calculated with equation 3, where n is the number of features, h1 is the number of hidden neurons in layer 1, h2 the number of hidden neurons in layer two and m the number of output neurons.

$$\Theta(nh1 + h1h2 + h2m) \tag{3}$$

By applying equation 3 to our network, we get (for feature set 1 with 7 features and twice 10 hidden neurons) 200 multiplications. For every neuron that we can delete, we reduce the network size by (7 + 10) multiplications and therefore get a speedup of 11.8%. From the results, we can see that this compression technique

Table 2. Performance Evaluation of the Network Compression for a 2 layer Tanh network with 10 hidden units per layer

Threshold angle	Avg number of deleted Units	Avg accuracy loss per unit
10 deg	1.02	10.52%
20 deg	1.63	6.89%
30 deg	2.85	4.61%

does not work for our neural network. If we delete all hidden unit vectors that have an angle of below 10 degrees, we lose, on average 10% of prediction accuracy for each hidden unit. Compared to this loss, we only lose around 2% of accuracy when training the network with only five hidden units per layer. The same high losses in accuracy are also the case for a pruning threshold of 20 degrees or 30 degrees.

8 Future Work Conclusion

Initially, we analyzed a small subset of features for this classification task, which has shown to achieve an accuracy of 53%. By applying the signed inverse mass probability function of the Poisson distribution and therefore applying prior knowledge to the data, we managed to increase the performance of the network significantly. Afterward, we tried to increase the network's accuracy with a genetic feature selector and achieve an accuracy of 48%. The GFS has shown to increase the performance of the network gradually but still underperformed. This is likely to the dataset being so small that the introduction of the validation set led to a high loss in accuracy. Therefore in the future, it should be analyzed if the described model can accurately predict a user's phone size. Additionally, in the future, the GFS can be further analyzed by introducing different crossover and selection techniques. The described compression technique could not be applied to our model, and other techniques have to be explored.

References

1. Gedeon, T.D., Harris, D.: Network Reduction Techniques. In: Int. Conf. on Neural Networks Methodologies and Applications, AMSE, San Diego, (1991)
2. Kim, J., Thomas, P., Sankaranarayana, R., Gedeon, T., Yoon, H.-J. (2015). Understanding eye movements on mobile devices for better presentation of search results. Journal of the Association for Information Science and Technology, 67(11), 2607–2619. doi:10.1002/asi.23628
3. Bushaev, V.: Adam - latest rends in deep learning, Blog, https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c accessed 6 May 2020 (2018)
4. Bushaev, V.: Stochastic Gradient Descent with momentum, Blog, https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d accessed 6 May 2020 (2017)
5. Prechelt L. (1998) Early Stopping - But When?. In: Orr G.B., Müller KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 1524. Springer, Berlin, Heidelberg
6. Gopal, K.P.S , Kishore, K.S.: Normalization A PreprocessingStage
7. Potdar, Kedar, Pardawala, Taher, Pai, Chinmay.:A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers. In: International Journal of Computer Applications (2017)
8. Chigozie, E.N, Winifred, I., Anthony, G., Stephen, M.:Activation Functions: Comparison of Trends inPractice and Research for Deep Learning
9. Fushiki, T. Estimation of prediction error by using K-fold cross-validation. Stat Comput 21, 137–146 (2011).
10. Leardi R, Boggia R., Terrile M.: Genetic algorithms as a strategy for feature selection. In: Journal of Chemometrics (1992).
11. Zhang G., Wang C., Xu B., Grosse R.: Three Mechanisms of Weight Decay Regularization (2019)