Neural Network Techniques on Banking Data

Yuxi Chen

Research School of Computer Science - ANU
108 North Rd, Acton ACT 2601
u6106850@anu.edu.au

Abstract. This paper explores the common techniques of parameter regulation for neural network model based on a bank market data set. We discuss about the selection of inputs for specific goals in classification problem. The paper mainly discusses about how to understand the original raw data, input encoding and to optimize the parameters and hyperparameters. We also discussed when the data set is too biased towards some result, what is the impact on the entire training process and the model and how to choose appropriate performance indicators. Using several comparison to led to some effective conclusions.

Keywords: Hyperparameter Optimization, Business Decision, Input Encoding, Evolutionary Algorithm, Neural Network

1 Introduction

As a mainstream machine learning method, neural network has long been used in various types of classification clustering, regression and other classic issues. Although neural networks can handle many complex models and solving many useful problems, there are so many tricky things to achieve a good model, such as the choice of parameters/hyperparameters, the trade-off between accuracy and model size, as well as the data encoding/decoding.

Data encoding for training back-propagation is one crucial and difficult part of training the neural networks model [1], especially in the area of business intelligence for most data inputs need to be well preprocessed to reach a reasonable and practical model. Furthermore, we also need to work on the data encoding part, to better explain what the result is representing and how it could be to help us for future use.

Optimization hyperparameters is another important and hard part of training the neural networks model [2]. Hyperparameters are referred to parameters whose values are often set before commencing training the model, distinguishing with those parameters like weights which are derived during training. There are various kinds of ways of hyperparameter optimization, such as grid search, random search, Bayesian optimization, etc.[3, 4]. In this paper, we will use method based on the evolutionary algorithm to carry out the selection and optimization of the hyperparameters.

The data used in this paper was collected from a marketing campaign conducting by a Portuguese bank[5, 6]. This is mainly done by real agents through cellular or phone. This dataset has so many attributes we need to deal with. In this case, we are going to first select related and useful inputs and to generate a model which could best suits the goal: prediction of who will make a bank deposit subscription. By doing this we could increase the probability of anyone who are using neural network for analyzing the function of a campaign and target on potential clients.

Through this paper, we are going to present some techniques on data preprocessing, the choice of parameters/ hyperparameters, the trade-off between accuracy and model size, etc.

2 Method, Data Set and Model Design

2.1 Inputs/Output Analysis

The inputs in this dataset is list below, as most of the inputs is categorical, which we need to transfer/encode them into several reasonable forms. It should also be noticed that some input variables are uncontrolled like several market indexes and rates, while others are partially controllable or fully controllable like most of contact information and personal information. Moreover, this dataset is not the full version one comparing those used in the source paper due to some policies, where we can find some important inputs are missing.

There are mainly 3 kinds of inputs to pre-process, first is continuous inputs like ages, which needs to be reflected to the valid and reasonable input area for sigmoid function. Another kind is job, education, dates, etc., which are categorical and need to be encoded in a suitable way. Others like black-or-white(include grey) inputs, which is often true/false or unknown.

Input variables relevant of personal information and known before contacts:

Age: the thing needs to be done is to normalized it to valid input area of the sigmoid function

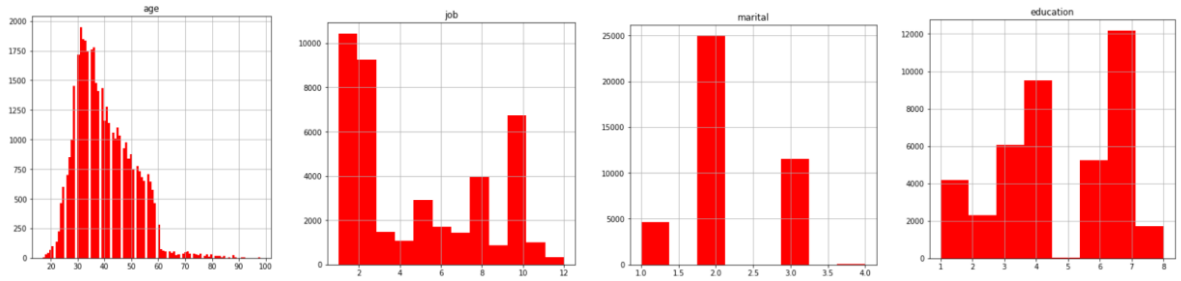


Fig. 1. The distribution of the *Age/Job/Marital/Education,* (the vertical axis means the sample volume)

Job/Marital/Education: For all these inputs are categorical, we need to find meaningful and efficient encoding ways with minimum of the information loss.

Job: Name the jobs by type like this (admin.:1,blue-collar:2,entrepreneur:3,housemaid:4,management:5,retired:6,self-employed:7,services:8,student:9,technician:10,unemployed:11,unknown:12)

For type 1/2/10 is the highest volume meaning the most clients are admins, blue-collars and technicians

Marital: Similarly, we consider marital to be 4 kinds (divorced:1,married:2,single:3,unknown:4)

Noticing that most are married (type 2) and unknown is relative low. Consider to combine divorced and single to one category, for these two types could have some common characteristics to some extent.

Education: We encode the education to 8 types: (basic.4y:1,basic.6y:2,basic.9y:3,high.school:4,illiterate:0, professional.course:5,university.degree:6,unknown:7)

Default/Housing/Loan: these are personal information in terms of financial status(credit in *Default*, *House* loan and personal *Loan*). All has three values yes/no/unknown, for the nature of sigmoid function, we encode it into (yes:1, no: -1, unknown:0).

These seven inputs above is what we could have on this client before any contact, so these inputs are valuable for the study of static before-the-campaign classification of what kinds of characteristics could have on the potential purchase.

Input variables relevant to the contacts:

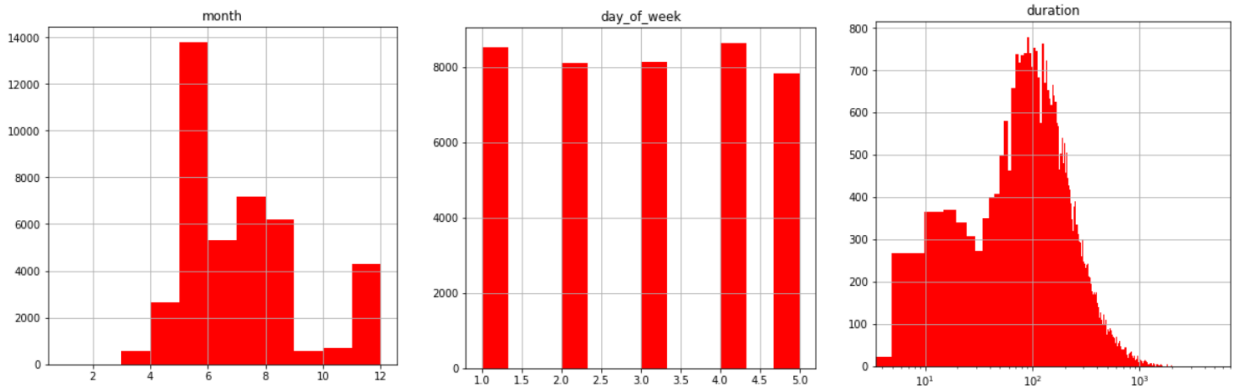


Fig. 2. The distribution of the *Month/Day of the week/Duration,* (the vertical axis means the sample volume)

Contact: This data is the way the communication conduct through cellular or telephone, we could simply use binary coding. Because this is more relevant to the quality of the contact, so we dropped for this static analysis.

Month: As we can say from above figure, there is no data in Jan and Feb and relative low in Mar, Oct, and Nov. And extreme high volume in May. This data may contain so much hidden information for in the different month, there are policies changes and market volatility, etc. It could be better that we use some more explicit data if possible like some market data at below.

Day of the week: Data seemed quite average, needs to encode some way for this may cause a net to generalize to 3 meaning Wednesday. This data is more likely important for the choosing of the contact date like month above.

Duration: This data meaning the last contact duration, which is significantly unusable for before-the-campaign analysis for normally we would have the result at the end of the call. Therefore, this raw data should not be used for before-the-campaign analysis, while we could use for studies like how to keep the clients buying the deposits. As we can see from figure 2, we present the duration with the horizontal axis using log scale and the distribution is normal except for many are below 10 seconds, which we consider the failure of the purchase

These three inputs above is what we would have during/after contacts, so these inputs are currently dropped for the study of static before-the-campaign classification.

Input variables relevant to the campaign:

Campaign: This input means the number of contacts performed during this campaign and for this client including the last contact. For nearly half of the data is 1, meaning the first contact is also the last contact that the agents could do in this campaign. For our first goal is to find out before the contacts, this data could be dropped. However, in further study, we could use this data to analyse other thing like call strategy for comparing what is happened for almost another third is 2.

Pdays: This input means the number of days that passed by after the client was last contacted from a previous campaign, in the raw data they use the value 999 meaning the client was not previously contacted. This number is a big and unusual number which may influence the process in the calculation of gradients in backpropagation. As the most data except the 999 is below 5, we could possible use an approximate binary coding to simply meaning whether the client is contact before. However, if the goal of the model is to find whether the contacting interval time (especially between the first and the second) could influence on the outcome, we should encode in some other way which could significantly distinguish the difference between first, second and so on.

Previous: This data is the number of contacts performed before this campaign for this client, this part is a bit overlapping with what we want to encode the *Pdays*. As we can say from below figure.3, we present the *previous* with log scale at horizontal axis, and showed a better distribution than the *Pdays*, therefore we decide to use this data instead of pdays in this paper.

Poutcome: This data is the outcome of the previous marketing campaign, having three values: failure, success, and non-existent. Therefore we use the -1,1,0 to encode this information.

These four inputs above contains some useable data and some redundant one, therefore we drop some inputs for better suiting the study of static before-the-campaign classification.

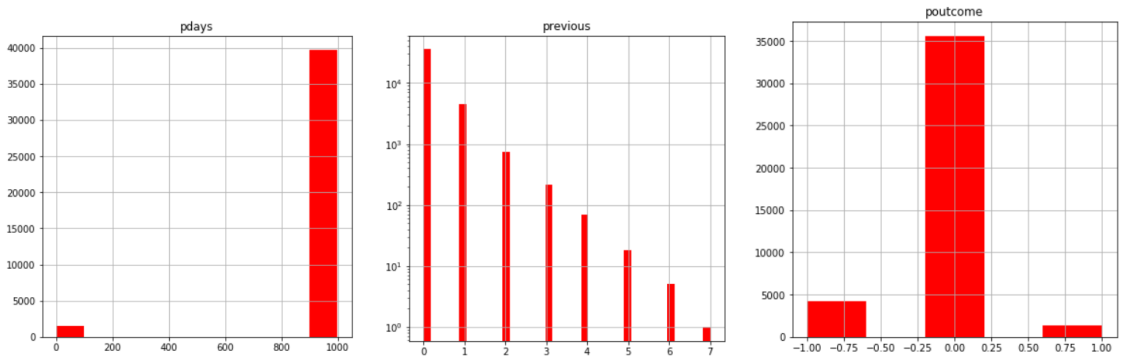


Fig. 3. The distribution of the *Pdays/Previous/Poutcome,* (the vertical axis means the sample volume)

Input variables relevant to social and economic context attributes:

Emp.var.rate/Nr.employed: The percentage and number of persons of who are employed, changing quarterly

Cons.price.idx/Cons.conf.idx: The consumer price index and consumer confidence index, changing monthly

Euribor3m: The data is meaning 3 month rate of the Euro Interbank Offered Rate, changing every market day

These five inputs above are inputs could have hiddenly influence the outcome and not controlled by the bank and/or the agent, therefore we just simply use them for a comparable test.

Output variable:

Y: The data is the output, showing the outcome whether the client has subscribed a term deposit. The raw data is Yes/No, therefore we just encode using binary.

What is we should notice is that only 11% of the outcome is the success subscription, therefore could lead the model prefer to output 0, which will have a useless model. In this situation, we decided to do several tests with resampling from these over 40,000 samples having different ratio on outputs to see whether our model is useful and valid.

2.2 Inputs Encoding examples

After the analysis of the inputs above, the rest of our main nine inputs: *Age/Job/Marital/Education/Default/Housing/Loan/Previous/Poutcome,* as well as the five comparison inputs: *Emp.var.rate/Nr.employed /Cons.price.idx/Cons.conf.idx/Euribor3m.*

We will show some examples of how we encode those inputs by using some suitable methods.

The main encoding haven't been done is the *Job* part, we use the method in [1], to separate in to a four digits encoding, and then normalized each attributes.

Table 1. *Job* encoding to four categories

Job	Admin, Entrepreneur and Management	Blue-collars and Housemaids	Technicians and Services	Others
Job 1	1	0	0	0
Job 2	0	1	0	0
Job 3	0	0	1	0
Job 4	0	0	0	1

For *Education* part, we could use two ways, first is a continues value representing that as the education degree goes up, the numbers goes up, and another is to have similar method for *Job* to cluster the 9y and under to one group, high school and professional course one group, university one group and unknown and illiterate for one group.

2.3 Neural Network Model

For this classification task, we use a three-layer neural network model to train. We use stochastic (minibatch) gradient descent with momentum to do the back-propagation. The hidden layer was initial set as 20 after several tests and could be changed for different study purpose. For the test of the model part, we use the cross-entropy to evaluate our model's performance

As mentioned above, when we train and test with the complete set of data, we find that the function tends to output zero for all kinds of inputs. While this can achieve almost 90% of the correct rate, because the entire data set, the probability of output no is around 90%. Therefore, accuracy may not be a good indicator for this kind of unbalanced dataset. We chose precision and recall to conduct further research. Furthermore, we chose to reduce the data set to, say there are almost 50% to 50%, we also divides the data into 30%, the success and the 70% failure, as a contrast.

We also further explored whether there is any specific difference between the use of one hidden layer and two hidden layers, and if there is no significant difference in performance, we will of course choose the neural network model with only one hidden layer

2.4 Evolutionary hyperparameter optimization

Evolutionary algorithm is an algorithm bio-inspired by the evolution theory commonly used in engineering control, machine learning, function optimization and other fields. As the development of chip technology, especially the remarkable improvement of graphics computing power as well as the invention of the highly integrated deep learning software kit which we can use effectively, the time efficiency for backpropagation is remarkable increased. Therefore, this paper chose to use the internal stochastic gradient descent function with momentum to calculate the weight of the whole neural network, which might be able to reduce some effects of saddle point problems. Besides the structure of the model especially referring to the number of hidden layers, and the number of neurons of each hidden layer, there are some important hyperparameters like learning rate and momentum which will significantly have influence on the entire model.

For optimization of hyperparameters of a traditional neural network, there are two basic tracks: the first is parallel search and the second is sequential optimization[2]. Parallel search refers to parallelly training multiple neural network with different hyperparameters, whose common methods include grid search (LIBSVM) and random search(H2O AutoML) [2]. Sequential optimization uses continuous, training with constantly optimized hyperparameters instead of parallel training processes including methods like, manual adjustment and Bayesian optimization process, which generally provide better solution despite of multiple training runs[2]. Therefore, both tracks have their inherent shortcomings, the former requires computational power to carry out parallel training and the latter that needs multiple runs may not does well in a lengthy model.

Considering to use the benefits and reduce side effects of both tracks, this paper uses a methodology in the [2] called Population Based Training (PBT) derived from the evolution algorithm. PBT requires a single training processes, which combines the method of parallel search and sequential optimization. The main procedure is showed in below table:

Table 2. *PBT algorithm*

Step	**Description**	**Function**
1	Initialize different hyperparameters	*Random*
2	Conduct parallel training	
	For each parallel	
2.1	Do one step of the optimization on the weights (SGD)	*Step*

2.2	Evaluate by the fitness function (Loss function)	*Evaluation*
2.3	For every evolution iteration (user defined)	
2.3.1	Decide if or not carry out the substitution	
2.3.2	Compare the performances of each other parallel	
2.3.2.1	Choose to substitute with parameters from parallel training	*Exploit*
2.3.2.2	Choose to substitute with new hyperparameters generated based on current ones	*Explore*
3	Repeat 2 until the termination condition fulfilled	

PBT algorithm starts similar to parallel search. However, each parallel process has different meanings for the whole algorithm compared to traditional parallel search. Each parallel thread is similar to an individual in evolutionary algorithms. For each parallel process, we perform a *step* first, which is using algorithms like SGD or Adam to do the weight optimization. Then we evaluate through the fitness function to find the performance among each individual. For every evolution iteration defined by users (normally we don't do the evolution every optimization iteration), we have two ways of evolution like the crossover and the mutation in evolution algorithm, called *exploit* and *explore*. Exploit is kind like the idea of crossover, to copy those weights preforming better from other parallel process, while explore is like mutation to have new hyperparameters. These combination optimization method for parameters and hyperparameters allows us to obtain advantages from both.

As we can see from above description, this algorithm would be applicable to a very wide range of models. For different models, we can set different kinds of initializing super parameters function, optimization function, evaluation method(fitness function), as well as the crossover and mutation function for parameters and parameters.

For this paper, we're going to use SGD with momentum as our optimization weight method, therefore the hyperparameters are the learning rate and the momentum. We initialize the learning rate drawing from an exponential random normal distribution between certain boundaries with a normal noise. Similarly, we initialize the momentum with random normal distribution with a normal noise.

2.6 Accuracy, precision, recall and F1

Performance could have various indicators, while accuracy may not be a good indicator especially for such an unbalanced data set. We tried to find out which indicators are better: accuracy, precision, recall and F1 rate[7, 8]. To be convincing, we first of all use the whole result consisting of actual result and forecast result to output into a matrix, and then select the corresponding data at different rows and columns to calculate the corresponding rate to find that which data is better for us to judge the actual performance of this model.

2.6 Others

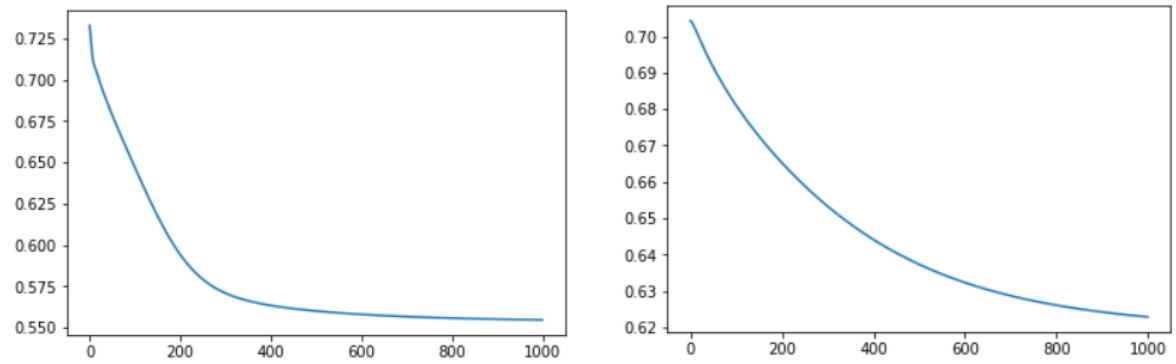
After a series of training, we chose to use more than 20 neurons as hidden layers which is still big for the whole net size and the training speed, and therefore we analyzed the weights of these neurons and used several methods in [9, 10] to deal with these units.

We have adopted a method here which is to compare the weight parameters vector angle after the training. To be specific, after training the entire network, use internal functions of Tensors to calculate the angle between the two neutron weight vector. Then we will find out whether they have too similar (less than 15 degrees) or can complement each other, if so we will delete some[9, 10].

It should be noted here that we did not adopt the strategy that he used in the [9, 10], which was normalized to 0.5, because it was observed that the Pytorch automatically normalized the initial weight matrix. The distribution is good (not all 0 or 1) and does not need to be further normalized.

3 Results and Discussion

3.1 Comparison for Adding the economic data



Whole training time:2863.6314868927ms
Testing Accuracy: 74.25 %

Whole training time:2824.063539505005ms
Testing Accuracy: 66.25 %

Fig. 4. The result (left: with economic data, right: without the data) with loss function 1000 iterations

We use resample number 8000 and encoding jobs to four digits.

As we can see from the figure 4, the training accuracy of the entire model has increased from over 60% to more than 70%, which is not the best case. The best accuracy results among several trainings even reached more than 80. After using economic data, the average accuracy has risen to about 72% among tests.

Therefore, we chose to use these economic data in the below tests because it can significantly improve accuracy while training time is not very slow.

3.2 Comparison with original paper performance for the whole set

Accuracy means the ratio of where the prediction output equals to the real output to the number of output in the whole dataset. For this case, accuracy means that the prediction of customer's choice is consistent with the actual choice of the customer to the overall proportion. This is not very practical for our application of to make a business decision.

Table 2. *Confusion matrix*

	Actual True	**Actual False**
Prediction True	True Positive(TP)	False Positive(FP)
Prediction False	True Negative(TN)	False Negative(FN)

However, we can use the recall rate meaning the ratio of among all the actual true (purchased in real) how many people are predictive true (prediction to purchase). Another one is the precision rate, which means the ratio of among all the prediction true (prediction to purchase) how many people are actual true (purchased in real). Using the table above, the recall rate equals to (TP / (TP+FN)) while the precision rate equals to (TP / (TP + FP)). Furthermore, F1 rate is the Harmonic Mean of the Precision rate and Recall rate.

NN model

(ROC point for D=0.5):

	Predicted	
Target	failure	success
failure	617 (48%)	140 (11%)
success	186 (14%)	350 (27%)

Confusion matrix for training:
tensor([[1066., 48.],
[293., 193.]])
Testing Recall:39.71 %
Testing Precision:80.08 %
Testing F1:53.09 %
Worker 15

Confusion matrix for training:
tensor([[988., 126.],
[203., 283.]])
Testing Recall:58.23 %
Testing Precision:69.19 %
Testing F1:63.24 %
Worker 3

Fig. 5 The confusion matrix result (left: original paper, middle/right: two workers from PBT algorithm)

From above figure (and more details in the code) we can see, our model does better in the precision rate while not doing well in the recall rate. The workers chosen to demonstrate are about the average performance.

This demonstrates that if we use our model to test a new set of data and make predictions on the result, for whom is predicated to subscribe has very high probability (precision rate) to do a real subscription, this is very good for us to make business decisions on this market promotion plan. But what we cannot ignore at the same time is that because the recall rate is very low which means that we can only predicate a small number of our customers among whom have the potential to make a real subscription, which would be a very huge loss of our market.

After several tests, we found that the recall rate was hardly above 75% at the last epochs, so I thought it was possible to do something beyond the model itself and the parameter tuning, It may be reasonable to doubt that our incomplete inputs have some influence on the model.

3.3 Comparison for different parameters of the algorithm

For this part, we're going to focus on some of the more important parameters of their impact on our model, and these parameters include the number of population, the number of training epochs.

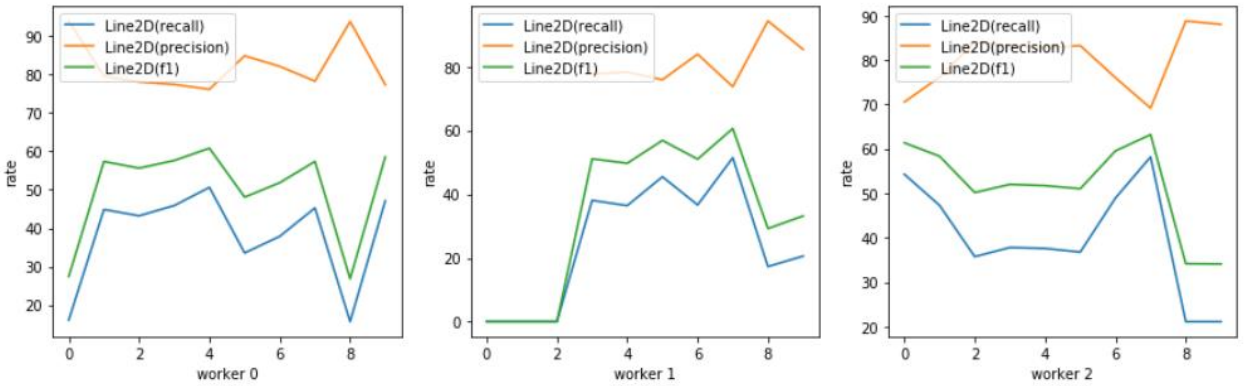


Fig. 6. Hand chosen 3 (out of 16) workers performance indicators with 10 epochs of 16000 samples

First, we use 16 individuals For the last epochs of 10, as we can see, almost every worker experienced some kinds of decline, whether in precision, recall rate or F1 rate. In fact, many workers have reached their best at the fifth and sixth epochs, and then have some kind of a decline, This probably could be that we take the loss function as the evaluation function as well as an incoming parameter of the performance function. That is to say, while the loss reduces, the truly meaningful performance indicators, such as precision, recall rate or F1 rate, fluctuate and even regress.

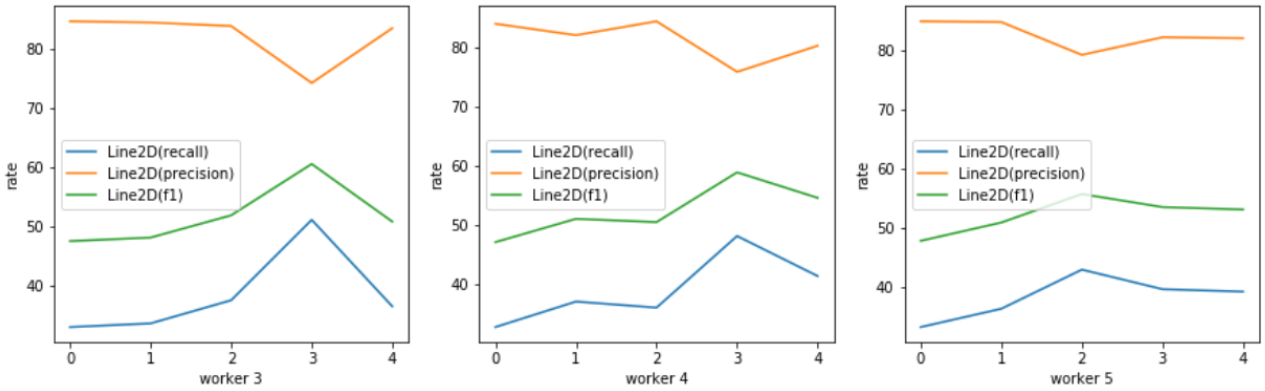


Fig. 7. Hand chosen 3 (out of 8) workers performance indicators with 5 epochs of 16000 samples

Test with 64 individuals of 5 epochs was also tried. From the log we find that only 3 (out of 64) workers has the recall rate to be over 60%, which is only a little bit better than the original paper. Another finding is because of the increase in the number of workers, there are many parallel processes with more than 90% of the precision rate. Besides, we found some workers are dead with all indicators equaling to 0.

There are other parameters in the parameter list that would have effect on the whole model, which here we don't have to show with detail. For example, if the batch size of train is too large, it will produce equivalent effect like training the whole network. If the batch size is too small, it is equivalent to one by one training, which greatly affects the training efficiency of the entire network. While the tester batch size is too small, the test could not a be typical training for the sample size too small to reach out wrong result.

3.4 Comparison for Different Input Coding (Jobs)

This comparison is designed to prove whether the method in the [1] to encode of the *job* as four variables with directly treat it as a continuous variable. Whether or not this method will have a crucial impact on the performance indicators and/or other part of the whole neural network model.
We will use the 16000 resamples.

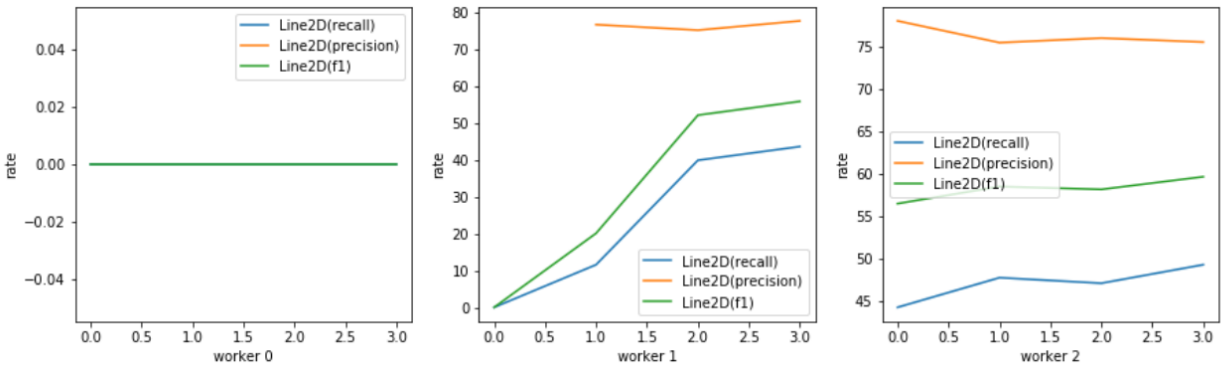


Fig. 8. Hand chosen 3 (out of 8) workers performance indicators with 5 epochs of 16000 samples (Job not encoded)

After many tests, we chose the representative image to show, as can be seen from figure 8, Without using this method [1], which encodes to four digits, the model is not as stable as what is like when we are using encoding of *Jobs.* We can also find that we have a little bit better performance in training time and precision. But it is not particularly prominent. While the recall rate is even better without encoding, which would be a little abnormal. As can be seen from this, we did not get way better in performance and faster time after using this coding, while the model could be more reasonable and stable with the encoding of some inputs.

3.5 Pattern Reduction

In this part, we explore some strategies to optimize the parameters of the whole network by calculating the between distinctiveness of hidden units, which is called pattern reduction [10]. After trying many times, and I've found that when the number of input attribute is very small, for example, two or three. If hidden layers used more than 8-16 neurons, there will be a large number of neurons that need to be trimmed off because the entire weight vector dimension is not so high, It is impossible to have so many vectors with angles between them more than 15 degrees.

But if the input variable's attribute is so many, as is to say the dimension is very high, like this model, when it comes to the 9 - 17 inputs attributes, the entire vector matrix of weights, under the optimization of the pytorch, will not be less than 15 degrees. So when I use my function to find those one needs to be reduced, it was not filtered out any. Moreover, if only I use like so many neurons like 1000 in the hidden layer, it could be overlapping. But the whole system is too slow, and it doesn't need to use so many neurons.

4 Conclusion and future work

4.1 Conclusion

From the above results, we can draw the conclusion that, first of all, economic data may have important impact of marketing activity. The second point is that the use of encoding does not always do well and improve all kinds of performances of this model, but such input encoding is more explanatory and scientifical rather than simply treat it as a continuous variable.

In addition, our neural network model does not necessarily have the ability to do all kinds of strict predication, for many inputs are transformed by us, which doesn't necessarily have a strict and accurate meaning. While the precision is a explicit indicator that our model does well in, we could use this to make business decisions: if the model predicts some customers to be positive, then use the best of the promotion and marketing skills on those customers.

Through this PBT algorithm, we can better find more suitable hyperparameters used for our model training, which is more efficient and scientific than manual search or use complicated method. There is a strong advantage especially for a person with unfamiliar models, it is very ideal and timely to use this algorithm to select and optimize hyperparameters.

Also, this algorithm has strong extensibility, with different functions to be chosen for different parts, we can apply it to many classical and tricky models in the field of depth learning, convoluted neural networks, reinforce learning, etc[11-14].

What needs to be noted is that we have two random sampling process, one of which is taking data from the entire data set to be consistent with output ratio we wanted called resample, another is, and the other is the process of random dividing the extracted part of the whole dataset into a training set and a test set. This may have a very important impact on the results, which we need to control in the future work for more valid and effective conclusions.

4.2 Future Work

Because one of the most important points in this algorithm is to make scientific comparisons of the performances among different workers. In this case, we use the loss function as the evaluation methods, but this is not necessarily very scientific, especially for this kind of business case, where precision, recall rate or f1 rate may be better methods. In the future, we can improve this by using establishing new evaluation criteria and methods.

In addition, we did not carry out tests with a large number of workers like over a hundred or even a thousand workers, which may be very impressive in the improvement of the entire model. Because the computing power and time efficiency considerations, we don't implement it here, and in the future we can deploy code to parallel systems, like cloud terminals, to conduct parallel computing on larger models, to find more useful conclusions of optimizing parameters/hyperparameters.

Another important point is that in our function, the method of evolution or mutation is not proved necessarily very effective to different hyperparameters[15], where we didn't do more tests. For example, for the learning rate, which one is better: using exponential change or using normal linear change. In the future, we can do some more tests with some effective algorithms for this subject.

We can use this model (with some different inputs encoding methods) to find whether the contacting order (the first, the second or the fifth) could influence on the performance of the agent, in which way we should encode in some other way which could significantly distinguish the difference between first, second and so on.

In this dataset, some inputs attributes are not likely usable for static predication model, while we could use to do times series analysis for through several campaigns what methods of campaign (like the frequency of contacts, the suitable and maximum contact times for one client) could lead to the buying with those inputs contains previous contacts information (duration, month/date,etc.).

References

1. Bustos, R. and T. Gedeon. *Decrypting Neural Network Data: A GIS Case Study*. in *Artificial Neural Nets and Genetic Algorithms*. 1995. Springer.
2. Jaderberg, M., et al., *Population Based Training of Neural Networks.* CoRR, 2017. **abs/1711.09846**.
3. Bergstra, J. and Y. Bengio, *Random search for hyper-parameter optimization.* Journal of Machine Learning Research, 2012. **13**(Feb): p. 281-305.
4. Bergstra, J.S., et al. *Algorithms for hyper-parameter optimization*. in *Advances in neural information processing systems*. 2011.
5. Moro, S., R. Laureano, and P. Cortez. *Using data mining for bank direct marketing: An application of the crisp-dm methodology*. in *Proceedings of European Simulation and Modelling Conference-ESM'2011*. 2011. Eurosis.
6. Moro, S., P. Cortez, and P. Rita, *A data-driven approach to predict the success of bank telemarketing.* Decision Support Systems, 2014. **62**: p. 22-31.
7. Davis, J. and M. Goadrich. *The relationship between Precision-Recall and ROC curves*. in *Proceedings of the 23rd international conference on Machine learning*. 2006. ACM.
8. Powers, D.M., *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation.* 2011.
9. Gedeon, T. and D. Harris. *Progressive image compression*. in *Neural Networks, 1992. IJCNN., International Joint Conference on*. 1992. IEEE.
10. Gedeon, T. and D. Harris. *Network Reduction Techniques*. in *Proceedings International Conference on Neural Networks Methodologies and Applications, AMSE, San Diego*. 1991. IEEE.
11. Such, F.P., et al., *Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning.* arXiv preprint arXiv:1712.06567, 2017.
12. Snoek, J., et al. *Scalable bayesian optimization using deep neural networks*. in *International conference on machine learning*. 2015.
13. Swersky, K., J. Snoek, and R.P. Adams. *Multi-task bayesian optimization*. in *Advances in neural information processing systems*. 2013.
14. Domhan, T., J.T. Springenberg, and F. Hutter. *Speeding Up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves*. in *IJCAI*. 2015.

15. Qin, A.K., V.L. Huang, and P.N. Suganthan, *Differential evolution algorithm with strategy adaptation for global numerical optimization.* IEEE transactions on Evolutionary Computation, 2009. **13**(2): p. 398-417.