The Impact of Properly Encoded Data

In Shallow and Deep Neural Networks

Bin Chen

Research School of Computer Science
Australian National University
Email: u6073011@anu.edu.au

Abstract. Data encoding is crucial in neural networks. It can be used to enhance the performance of the network and make the network learning much easier. As well as, it can improve the accuracy of the network. These are based on extracted or enhanced critical features when feeding them to the network. The aim of this report is to show what techniques can be utilised and what are the effects of these techniques when applying them to shallow and deep neural networks. Encoding techniques like handling missing values, removing unnecessary features and data normalisation will be used to achieve the aim. The results were compared with another paper which has used the same data set. The results of the shallow network are lower than the results from the other paper, but the deep neural network had better results.

1 Introduction

1.1 The motivation of the choice of the data set

Choosing a dataset is considered in the first place for experiment conduction. Since the aim is to show the effects of the data encoding on different neural networks. Designing a simple classification problem is sufficient to demonstrate the aim of this experiment. So, I decided to use the mushroom classification dataset.

1.2 Problem Background

This dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family (Dua & Efi, 2017). The problem is to devise a neural network to find suitable pattern representation which can be used to classify mushrooms correctly into two classes- a definitely edible and a definitely poisonous- which the latter class also includes unknown edibility and not recommended class (Bustos & Gedeon, 1995).

1.3 The aim of the investigation

The aim is to try out different techniques of data encoding on shallow and deep neural networks. Then find out the effects of data encoding, whether the neural networks will learn better and produce more accurate results between properly encoded data and raw non-encoded data when feeding them to the neural networks.

2 Method

2.1 Data analysis and encoding

2.1.1 Converting to numeric numbers

The dataset of the mushroom records were drawn from The Audubon Society Field Guide to North American Mushrooms (Dua & Efi, 2017), New York: Alfred A. Knopf. It consists of 8124 instances, 22 features and there are some missing values in the dataset. (As shown in table 1: original raw dataset).

Table 1: original raw data set

1	p	x	s	n	t	p	f	c	n	k	e	e	s	s	w	w	p	w	o	p	k	s	u
2	e	x	s	y	t	a	f	c	b	k	e	c	s	s	w	w	p	w	o	p	n	n	g
…																							
8123	p	k	y	n	f	y	f	c	n	b	t	?	s	k	w	w	p	w	o	e	w	v	l
8124	e	x	s	n	f	n	a	c	b	y	e	?	s	s	o	o	p	o	o	p	o	c	l

The columns are the features of the dataset as shown below (Table 2: the feature information table).

Table 2: the feature information table

1. identified: p= definitely poisonous, e= definitely edible
2. cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
3. cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
4. cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r, pink=p,purple=u,red=e,white=w,yellow=y
5. bruises: bruises=t,no=f
6. odor: almond=a,anise=l,creosote=c,fishy=y,foul=f, musty=m,none=n,pungent=p,spicy=s
7. gill-attachment: attached=a,descending=d,free=f,notched=n
8. gill-spacing: close=c,crowded=w,distant=d
9. gill-size: broad=b,narrow=n
10. gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g, green=r,orange=o,pink=p,purple=u,red=e, white=w,yellow=y
11. stalk-shape: enlarging=e,tapering=t
12. stalk-root: bulbous=b,club=c,cup=u,equal=e, rhizomorphs=z,rooted=r,missing=?
13. stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s
14. stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s
15. stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y
16. stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y
17. veil-type: partial=p,universal=u
18. veil-color: brown=n,orange=o,white=w,yellow=y
19. ring-number: none=n,one=o,two=t
20. ring-type: cobwebby=c,evanescent=e,flaring=f,large=l, none=n,pendant=p,sheathing=s,zone=z
21. spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r, orange=o,purple=u,white=w,yellow=y
22. population: abundant=a,clustered=c,numerous=n, scattered=s,several=v,solitary=y
23. habitat: grasses=g,leaves=l,meadows=m,paths=p, urban=u,waste=w,woods=d

There are a number of features in the dataset, which means we need to create many neurons in the layer to process the features. That's not a big problem here with today's computer storage and processing power. But the features are encoded with English letters, each letter is an abbreviation which is used to describe the feature, e.g. r stands for red, b stands for black etc. But our neural network doesn't understand English letters, so the letters have to be converted into numeric numbers for the network to make sense out of the data and process them.

Here is a table of the proposed encoding for the features in the dataset.

Table 3: proposed encoding data

1	3	4	1	1	8	3	1	2	1	1	4	4	4	8	8	1	3	2	6	1	4	5	1
2	3	4	10	1	1	3	1	1	1	1	2	4	4	8	8	1	3	2	6	2	3	1	0
…																							
8123	5	3	1	2	4	3	1	2	3	2	7	4	3	8	8	1	3	2	2	8	5	2	1
8124	3	4	1	2	7	1	1	1	12	1	7	4	4	5	5	1	2	2	6	6	2	2	0

In this proposed encoding, the English letters were replaced with numeric numbers. For example, if a feature has 3 categories red, blue and black, then numeric number 1 will be representing the first category red, 2 representing the second category blue and 3 for the third category black so on and so forth. Furthermore, the column of identified feature has been moved to the last column.

2.1.2 Handling missing values

The other problem with the data set is that there are some missing values found in the stalk-root feature. Depending on the situation there are a number of options can be taken to handle this problem. The first solution is to include the missing values in the data by filling a value as categorizing the missing values to a category, only if there aren't many missing values. The result of this will introduce some noise to the dataset which might affect the performance. The second solution is to exclude the missing values, as a result, we will be losing valuable data if there aren't many data available to the experience already, then this is not a suitable option. For this experiment, the first solution is taken since only 1 feature has missing values.

2.1.3 Removing unnecessary features

When analysing closely at the veil-type feature, it appears that all the instances have the same value "partial" which are represented as 1 in the proposed encoding data. In this case, removing the feature would be beneficial to the neural network as it is not much use to the network as a feature to identify the difference between different mushroom classes. As a result, the network has fewer features to learn which will speed up the learning process.

2.1.4 Normalization

Normalization is vital in encoding data. It can increase the training speed, as the network will need less training to get to the optimal result. It can also reduce the chance of stuck in the local optima. Therefore, the normalisation to the data can save the training time and get a higher chance of giving optimal results.

2.2 Modelling the problem

2.2.1 Shallow neural network

For the shallow neural network, a standard 2-layer neural network with sigmoid function and Rprop optimization algorithm is designed to handle this mushroom classifying problem. The first layer of the network is the input layer. The size of the input is normally depending on the size of the features in the dataset. To decide how many features will be used in the network, a close look at the meaningfulness of the features was carried out. The conclusion is that all the features are equally important to each other except one feature- veil type. So, a final 21 neurons will be used to represent the 21 features. 1 output layer of two neurons represents the 2 prediction classes and 1 hidden layer. For the hidden layer, a final number of 65 neurons was determined by trying out different number of neurons to get the best results (Shown in table 4). For the training and testing data I have randomly split 80% of the dataset into training data, and 20% for testing data.

Table 4: testing different numbers of hidden neurons

Number of hidden neuron	40	45	50	55	60	65	70	75	80
accuracy	93.71 %	94.87 %	94.49 %	95.89 %	93.97 %	96.88 %	95.05 %	94.88 %	94.47 %

2.2.2 Deep neural network

For the deep neural network, it is similar to the shallow neural network. It also had 21 neurons for the input layer, but instead of one hidden layer like the shallow neural network, it had three hidden layers. Each hidden layer had 65 neurons. The last layer is the same as the shallow network it consists of two neurons for predicting the two class. The activation function used in the deep neural network is ReLU. ReLU is a popular activation function, the advantage to use ReLU instead of Sigmoid is to prevent vanishing gradient, and it will learn much faster than Sigmoid.

2.2.3 Neural network training

When training the networks, I need to determine the epoch value. First I set the epoch value to a very large number e.g. 5000 epochs. Then I started training the network on the 80% of the dataset. The network will print out the loss value for every 10 epoch, then I closely examining the loss value until it can't get any smaller, then I will stop the training and record the epoch value. I will use the recorded epoch value as the final determined epoch value for the network. By doing this I can prevent the network getting overfitted, this means the network will try to memorize the training data set instead of learning the general pattern from the training data, and it will do really well on the training data, but during the testing stage, the network will not do so well.

2.3 The methods used to perform the analysis

During the training, a printout message of accuracy for each epoch will be displayed to analysis the performance over time. A confusion matrix was used to determine the accuracy of the network. The columns in the confusion matrix are the predicted classes of the mushroom. The rows are the actual classes of the mushroom. From the confusion matrix, the number of the corrected and incorrect prediction can be analysed. Applying the confusion matrix in the training and test. The matrix can be used to analyse how well the neural network has learned during the training process. In the testing process, the matrix can be used to show how much the neural network has actually learned not just memorizing the training test set. To get a more statistical view of these numbers, a formula- the total number of correct predicted dividing the total number of the tests- can be used to calculate the accuracy in percentage.

3 Results and Discussion

3.1 Shallow and deep network results

The shallow network has achieved testing accuracy: 95.33%. It has identified 796 poison mushrooms and 46 false positive edible mushrooms as poison mushrooms. Also, it has identified 716 edible mushrooms, and 28 false negative poison mushrooms as edible mushrooms.

Table 5: Shallow network confusion matrix for testing:

Classes	Poison (predicted)	Definitely edible (predicted)
Poison	796	28
Definitely edible	46	716

For the deep neural network, the testing accuracy is 99.88%. It has identified all the 840 poison mushrooms. Also, it has identified 806 edible mushrooms, and 2 false negative poison mushrooms as edible mushrooms.

Table 6: deep network confusion matrix for testing:

Classes	Poison (predicted)	Definitely edible (predicted)
Poison	840	2
Definitely edible	0	806

3.2 Results of different data encoding techniques

From the results, we can see that the techniques have huge effects on the deep network but have fewer effects on the shallow network. This is very surprising to me, so with properly encoded data, the neural network can learn much effectively from the data set.

Table 7: the impact of techniques

Techniques	Shallow network testing accuracy	Change in accuracy	Deep network testing accuracy	Change in accuracy
Handling missing values	89.93%	0%	52.41%	0%
Removing features	91.22%	1.29%	88.4%	35.99%
normalization	95.89%	4.67%	99.88%	11.48%

Note: the accuracy is different for every training.

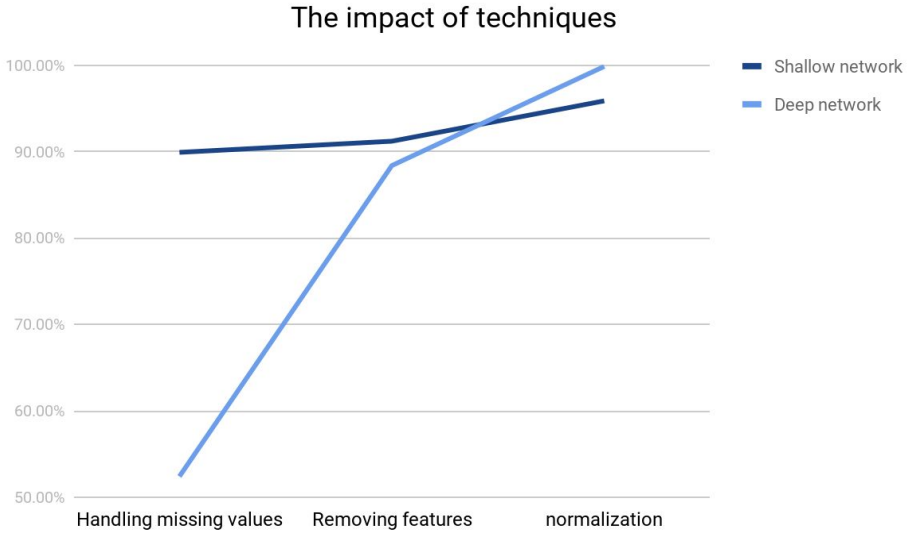

Figure 1 | The impact of techniques. The accuracy of the shallow network has increased by 6%, but the deep network has increased by 47%.

3.3 Results compared with another paper

To gain a better understanding of how well neural networks has learnt. I'm going to compare our results with other results from another paper.

Table 8: Comparison between the other paper

	Our shallow network result	Our shallow network result	Other Results
Accuracy	95.89%	99.88%	99.41%
Technique	Data encoding	Data encoding	MLP2LN, SLF method

3.4 Discussion

From the results, we can see that the techniques used in the data encoding helped to improve the accuracy of the neural network. Some helped a little and some helped a lot. By applying the removing feature technique, we are able to increase the accuracy by a 1.29% on the shallow network. On the deep neural network, we are able to gain a huge 35.99% on the accuracy. The technique is not hard, just by removing a column from the dataset. As a result, we have gained accuracy and saved one input neural which saved computing time. By normalising the data, we have improved the accuracy by 5.59% on the shallow neural network and increased 11.48% on the deep neural network. So, in total by applying all the techniques we are able to gain about 6% accuracy on the shallow network and around 47% for the deep neural network. In addition, the accuracy changes every time when retraining the network. This is because the initial weights for the neurons in the network are initialized randomly at the beginning of each training.

By comparing our accuracy with the results from the other paper. We can see that our neural networks have done a pretty well. In the other paper, with MLP2LN as well as SLF method, giving 48 errors, or 99.41% accuracy on the whole dataset (Duch, Adamczak, & Grabczewski, 1997). On our shallow neural network, it has achieved 96.81% accuracy rate on identifying the corrected mushroom class, which is a bit less since the results from the other paper. But, our deep neural network has achieved 99.88% accuracy which is even higher than the results from the other paper.

4 Conclusion and Future Work

In conclusion, we have seen the importance of the properly encoded data. By simply applying the data encoding techniques like handling missing values, removing unnecessary features and data normalisation, we could improve the accuracy up to 6% on the shallow network and 47% on the deep network, so properly encoded data is inevitable in building a neural network to get the best results. But to get the best results, only focusing on the data encoding is never enough, but it is an essential beginning.

Even though we have got pretty good results. There is still a lot of work left for us to do. Other work that we can still apply to improve the accuracy of the neural networks. For example, applying evolutionary algorithms on the networks, trying out different activation functions, experimenting different thresholds or weights of the hidden neurons. In the future, we will conduct more experiments with aforementioned factors.

References

Bustos, R. A., & Gedeon, T. D. (1995). *Decrypting Neural Network Data: A Gis Case Study*, Vienna.

Dua, D., & Efi, K. T. (2017). *UCI Machine Learning Repository*. Retrieved from: http://archive.ics.uci.edu/ml

Duch, W., Adamczak, R., & Grabczewski, K. (1997). Extraction of crisp logical rules using constrained backpropagation networks.