Comparison between simple neural network, rules generation and deep neural network

Zhizhen Huang

U6728618@anu.edu.au

Abstract. I have trained a simple neural network based on the GIS data [1][2]. I tried to use the method introduced by explaining grades [3] but failed to generate some useful rules to determine the category of the GIS data. However, I did find the key attribute of it. Then I trained a deep neural network to do the classification. To find out which method is the best, I compared these three and figured out the xxx is the one. In the end, I analyzed the reason why it is the best method.

Keywords: Explain predicted result, input gradient, dropout, BatchNorm

1 Introduction

Nowadays, the need of digital map generation increases hugely because everyone is used to use the map on his digital device. However, map generation needs huge amount of human effort in the traditional way. GIS data can help a lot in this field. If we can use a small amount of these data to train a neural network that could do the land category classification, it will be a great progress in this area. To achieve this, I tried these three ways, simple neural network, rules generation and deep neural network.

2 Data pre-process

The GIS data that was used to train the network contains 190 patterns. Each pattern contains 22 numbers. These numbers represent: id, aspect, altitude, topographic position, slope, geology descriptor, rainfall, temperature, landset bands 1 to 7 and 5 category.

For the aspect, it has 3 values, aspect, sin and cos aspect. For my understanding, the aspect is a circular value so I cannot use simple linear squashing function to deal with it. (0 degree is the same as the 360 degrees) The values of this attribute are range from 0 to 80. The sin and cos aspect are redundant. So, I will drop that 2 attributes when pre-process the data. For the aspect, I used 4 digits to represent it. Each digit is one of 0, 0.5 and 1. In this way the distance of two vectors is constant if they are 45 degrees away.

Table 1. Aspect along with direction

AS	0	10	20	30	40	50	60	70	80
Dir.	Flat	N	NE	E	SE	S	SW	W	NW

For the altitude, slope, rainfall, temperature and landset band 1 to 7, I found nothing special in it. So, I will just use the simple squashing function to change the values into 0 to 1.

For the topographic position, there are 6 different values which represent 6 different positions from gully to ridge. The bigger number represents the higher slope. It is categorical. So, I decided to use 0, 0.2, 0.4, 0.6, 0.8, 1 to replace them.

For the geology descriptor, it is not distributed normally. The values are range from 10 to 90 while most of them are 50, 70 and 90. For this attribute I used the method that professor TD Gedeon provided in the GIS paper [1]. The detail solution is use 4 digits to represent this attribute. if the value is not in 50,70,90, its representation would be [0.9, 0.1, 0.1, 0.1]. The second digit represent whether the value is 50. The third is for 70 and the fourth is for 90. In this way, the distribution will keep mostly the same as the original data.

For the output, there are 5 attributes. I combined them as one. I used 0 to 4 to represent these 5 classes: Scrub, Dry scler., Wet-dry scler., Wet scler., Rain Forest.

After all the steps I mentioned above, I got a dataset with 21 attributes. Within these attributes, 20 of them are input, the last one is the target.

3 Build and train simple neural network

After pre-processing the data, I started to build a neural network to learn about these data. Inside these data, I divided them into 2 parts with the ratio, 8:2. 80% of the data will be the training set.

The neural network has 3 layers, input, hidden and output. For input, as I mentioned above, it will have 20 inputs. So, the input layer will have 20 neutrons. For the hidden layer, I tried several numbers from 5 to 30, and decided to use 10 as my final choice. It because this one has a better accuracy. For the output layer, I set 5 neutrons because the network's target is to classify these data into 5 classes. Inside the network, I used ReLU as the activate function for the hidden layer. I used sigmoid function at first. However, due to its 0.25 restriction, the network will end up putting all the patterns into the same class, Dry scler.

For the learning part, I used CrossEntropyLoss as the loss function. Because it combines softmax and NLLLoss, so I did not add activate function for the output layer. Then I used SGD as the optimiser and set the learning rate 0.1. I started with 0.01, but the loss changes really slow after 500 iterations.

After training the network, it got 68.42% (104 out of 152) accuracy in training set and 63.16% (24 out of 38) accuracy in test set.

4 Input gradients analysis

Inside the explain grade paper, professor TD Gedeon mentioned two things to do with the neural network, pruning and causal index. For my understanding, the most effective way to determine which input is most 'important' is to calculate its gradient. The bigger the gradient of the input is, the more influence it will have on the output.

The pytorch provides a convenient way to calculate the gradient of each input to the output. As long as the input torch sets the "requires_grad" equal to True, we can use torch.grad to get the gradient. However, the result is 156 * 20 which means the gradients are for each 20 attributes of 156 patterns. The means of gradients of each attribute will represent the impact to the output it has.

Table 2. gradients of each input

Input	A1	A2	A3	A4	AL	TP	SL	G1	G2	G3
Grad.	0.0004	-0.0072	-0.0085	0.0053	0.0114	0.0207	-0.0085	0.0098	-0.0117	-0.0094
Input	G4	RA	TE	T1	T2	T3	T4	T5	T6	T7
Grad.	-0.0138	-0.0084	-0.0037	0.0032	-0.0024	0.0084	-0.0019	-0.0053	-0.0022	0.0050

As the table shows, these attributes, topographic position (TP) and geology descriptor (G4) have more impact on the final prediction.

I tried to use these attributes to generate rules to classify the pattern into different class. However, unlike professor TD Gedeon got several rules in the explain grade paper, I made no progress on this stage. In this situation, the TP only has 6 values and the G4 only has 2 values. Each type does not contain unique combination of these two attributes. So, the rules that generated by these attributes cannot do the classification properly.

5 Deep neural network with dropout/BatchNorm

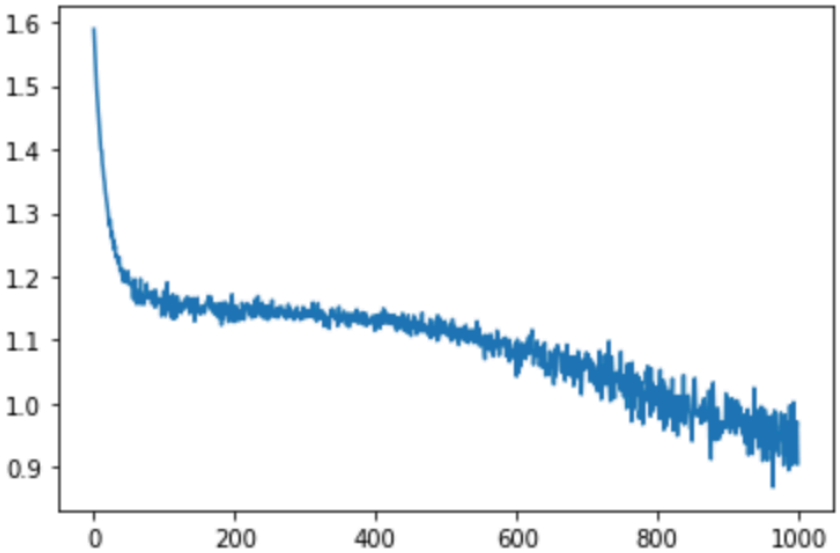
3 hidden layers with dropout or BatchNorm may fit the situation because the data amount is quite small for a deep neural network.
For the DNN with dropout, the neuron numbers of hidden layers are 30, 20, 10 which are optimal after 20 times experience. Then used nn.Dropout to implement the dropout layer for each hidden layer. The dropout rate is 30% which is the optimal from 0 to 50%.
After training, the confusion matrix and loss/epoch graph are showed below:

Graph 1. Confusion matrix of DNN with dropout

```
Confusion matrix for testing:
[[ 0.  1.  0.  1.  0.]
 [ 0. 13.  0.  5.  0.]
 [ 0.  0.  0.  2.  0.]
 [ 0.  1.  0.  9.  0.]
 [ 0.  0.  0.  1.  0.]]
```

```
Confusion matrix for training:
[[ 0. 10.  0.  2.  0.]
 [ 3. 50.  0. 25.  0.]
 [ 0.  2.  0.  5.  0.]
 [ 0. 18.  0. 38.  0.]
 [ 0.  0.  0.  4.  0.]]
```

Graph 2. Loss verse epoch of DNN with dropout



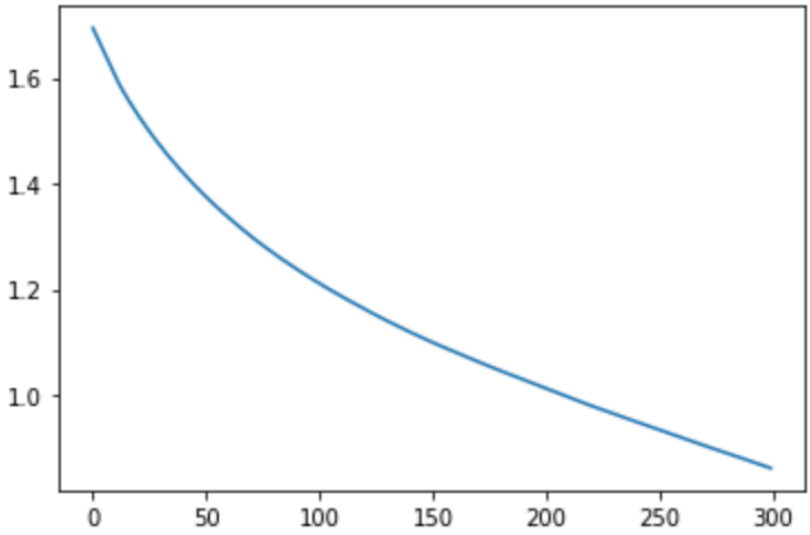
For the DNN with BatchNorm, the neuron numbers of hidden layers are 30, 20, 10 which are optimal after 20 times experiments. In these experiments, the result showed that using BatchNorm will cause overfitting easily. So, the learning rate and the epoch numbers should be smaller. The optimal one is 0.01 for the learning rate and 300 for the epoch numbers.
After training, the confusion matrix and loss/epoch graph are showed below:

Graph 3. Confusion matrix of DNN with BatchNorm

```
Confusion matrix for testing:
[[ 0.  1.  0.  1.  0.]
 [ 0. 13.  0.  5.  0.]
 [ 0.  0.  0.  2.  0.]
 [ 0.  1.  0.  9.  0.]
 [ 0.  0.  0.  1.  0.]]
```

```
Confusion matrix for training:
[[ 0.  9.  0.  3.  0.]
 [ 0. 73.  0.  5.  0.]
 [ 0.  7.  0.  0.  0.]
 [ 0. 18.  0. 38.  0.]
 [ 0.  2.  0.  2.  0.]]
```

Graph 4. Loss verse epoch of DNN with BatchNorm



6 Discussion

In these three methods, the rules generation has the worst performance while the others have similar performance. Simple NN has 68.42% training accuracy and 63.16% testing accuracy. DNN with dropout has 56.05% training accuracy and 66.67% testing accuracy. DNN with BatchNorm has 70.7% training accuracy and 66.67% testing accuracy. Seems like the DNN can improve a little bit performance compared to the simple NN. However, the improvement is too slight while the DNN is way more complicate than the NN. In conclusion, for this GIS data, the simple NN is enough for the classification.

For the failure in rules generation, I think it has several reasons. I will introduce them below one by one.

Firstly, the two datasets are different in the relationship between input and output. In explain grade's situation, the final outcome has strong relationship with the inputs. Because the final grade of a student is calculated by summing up each assignment/lab/exam time its weight. The input and output have a linear like relation. However, the GIS dataset does not have this kind of relation.

Secondly, the GIS dataset has more attributes the grade dataset. It more difficult to generate the rules. For a dataset with 10 inputs, the rule that sets restrictions to 5 inputs may be enough to classify the data. While for the dataset with 20 inputs, restrictions to 5 inputs may be far away from enough.

Thirdly, all the attributes in the grade dataset have clear definitions. In other words, we can figure out each values' meaning in this dataset. However, in the GIS dataset, some attributes like geology descriptor have no clear meaning. Moreover, it is also an 'important' input which has great impact on the final prediction. This problem also stands in my way to create the reliable rules.

The final problem is that my neural network is not well-trained. I can see that through the confusion matrix. Although the network got 60% + accuracy, it can just classify data into 2 classes (actually is 3 classes, but most of the data are in 2 classes). So, part of the accuracy thanks to the distribution of the dataset. Most of the data are in 2 classes, the network put all the data into these 2 classes. The precision of these 2 are, 72% (67 out of 93) and 64% (34 out of 53). If the precision is higher, the rule to classify may be clearer.

7 Conclusion

In this paper, there ways were implemented to classify the GIS data into 5 classes. The result is that rules generation has the worst performance while simple neural network and deep neural network have similar performance. The improvement of the DNN is not obvious enough. This may due to the complexity in adjusting the parameters of the DNN. For the future work, more hidden layers or changing more parameters may help get better performance.

References

[1] Bustos, RA and Gedeon, TD "Decrypting Neural Network Data: A GIS Case Study," *Proceedings International Conf. on Artificial Neural Networks and Genetic Algorithms* (ICANNGA), Alès, 1995.

[2] Milne, LK, Gedeon, TD and Skidmore, AK "Classifying Dry Sclerophyll Forest From Augmented Satellite Data: Comparing Neural Network, Decision Tree & Maximum Likelihood," training. 1995;109(81):0.

[3] Gedeon TD, Turner S "Explaining Student Grades Predicted By A Neural Network," *Proceedings of 1993 International Conference on Neural Networks* (IJCNN-93-Nagoya, Japan) 1993 Oct 25 (Vol. 1, pp. 609-612). IEEE.