1. CLASSIFYING DRY SCLEROPHYLL FOREST: A GIS CASE STUDY

Jinwen Zhu,

Research School of Computer Science,
Australian National University
u6108702@anu.edu.au

Abstract. We obtained some data through satellite image analysis and ground observation, and now we want to predict the type of forest-based on the available information. To obtain accuracy and comprehensive expression, a neural network is used to build a basic model. In this process, we try to prune the unnecessary and less important hidden neurons by similarity measurement. We calculate the angle between the output vectors to determine whether this neuron should be pruned. We acquired a prediction accuracy of about 70% on the test set. And the accuracy on the test set after pruning usually decreases a little but sometimes it increases since overfitting before.

Keywords: GIS, Neural network, Hidden neurons pruning, Distinctiveness measure, Vector Angle Analysis

1 Introduction

Satellite data can be very good at distinguishing some huge terrain differences and establishing a geographic information system, such as distinguishing mountains and rivers, but it cannot make the distinction of smaller geomorphology. Through satellite data, we can only know that this is a forest and It is impossible to distinguish what type of forest it is. So, we need some detailed ground observation data provided by geographers to build this model. Because the application of this model must involve a lot of data, which may make the neural network require a lot of calculation and time to run, so we want to prune some neurons to compress the neural network and under the condition of ensuring acceptable accuracy, we want to improve efficiency as much as possible under conditions.

1.1 The Dataset

I am using geographic data from the Nullica National Forest area on the south coast of New South Wales [1]. The area is about 20 x 10 km, divided into a grid of 179831 pixels, and the size is 30 x 30 m. The available information comes from a rectangular grid of 244,494 points and is a vector of 16 values, of which 7 are from satellite images, and 9 values are from soil models and aerial photography derived terrain models. The last five are forest supra-types. The data provided by geographers have been encoded in some way. Our model aims to predict a forest is dry sclerophyll or not.

1.2 Data analysis

For a lot of data, as humans, we can understand very well, but when the computer runs them, there will be some ambiguity, so we use some methods to pre-process these large amounts of data. We first indicated these data in some proper way [2] to avoid misunderstanding. and then since the distributions of data are different, it will inevitably lead to different gradient descent of each dimension. It is difficult to reach the lowest point of the cost function using the same learning rate. After normalization processing, the cost function becomes "rounder", and it is easy to perform gradient descent. By doing this, we expect the dataset can produce meaningful predictions.

- ✧ Aspect: Since aspect value is circular and the raw encoding cannot show this feature. We use 4 units to represent north, east, south, and west, then all the eight aspects can be encoded. For example, the north can be coded as 1, 0.5, 0, 0.5, and north-east can be represented as 1, 1, 0, 0, and so on. Then the original aspect can be deleted as well as aspect-sin and aspect-cos values.
- ✧ Altitude: The values of altitude are distributed normally, so we just need to use a linear squashing function to squash the range from 7 - 71 to 0 - 1.
- ✧ Topographic position: This variable is categorical, and the indicator seems to be relative. So, we encode topographic position in this way: gully as 0.0, lower slope as 0.25, mid-slope as 0.5, upper slope as 0.75, and ridge as 1.0.
- ✧ Slope degree: The degree increases exponentially with the value. We guess that this is not coincident and it may have an important effect on our precision. So, we try to linear squash it to range 0 – 1 but keep the exponential relation.

- ✧ Geology descriptor: It is hard to encode since we don't exactly understand the relationship it wants to indicate between the numbers and the information. So, we just use the frequency divide the number of all pixels to encode it.
- ✧ Rainfall, Temperature, and Landsat bands: Use a single continuous input. Simple linear squash to range 0 - 1.

2 Main Method

2.1 Network architecture

Since the data is in one dimension, we selected a NN as the basic model.
The standard NN reference model used in this study has a multi-layer perception of 17 neuron input layers, 14 neuron hidden layers, and 2 neuron output layers. Regarding reference training, 1000 epochs will be performed using the training data, and the results will be evaluated based on the test data.
I designed a network with one hidden layer and the general expression of the prediction model is:

$$y(x) = Sigmoid((f(x,w1),w2)$$

where $w1$ and $w2$ represent the weight and bias of the input layer and hidden layer respectively.
Both Reference and Comparison NN utilizes Mean Squared Error (MSE) for loss calculation, Adam algorithm for optimization with a learn-rate of 0.01. Since we only have about 30 samples in the test set, it is better to use 10-fold cross-validation.

2.1 10-fold cross-validation

In the process of machine learning modeling, the common practice is to divide the data into a training set and a test set. The test set is data independent of training, and does not participate in training at all, and is used for evaluation of the final model. In the training process, there is often a problem of overfitting, that is, the model can match the training data very well, but cannot well predict the data outside the training set. If the test data is used to adjust the model parameters at this time, it is equivalent to knowing part of the test data information during training, which will affect the accuracy of the final evaluation result. The common practice is to divide a part of the training data as validation data to evaluate the training effect of the model.

The validation data is taken from the training data but does not participate in the training, this allows a relatively objective assessment of how well the model matches the data outside the training set. It divides the original data into 10 groups. Each subset of data is used as a validation set, and the remaining 9 groups of subset data are used as training sets, which will get 10 models. The 10 models are evaluated in the validation set, and the final error is averaged to obtain the cross-validation error.

2.2 Hidden-neuron pruning

In a neural network, each neuron in the lower layer is connected to the upper layer, but this means that we must perform many floating-point multiplication operations. In a perfect situation, we only need to connect each neuron with several other neurons, without performing other floating-point multiplication operations. This kind of network is easier to compress, and we can skip zero during inference, thereby improving latency. Faster and smaller networks are very important for running them on mobile devices.
In the beginning, choosing the appropriate model structure and the number of hidden neurons often depend on the designer's experience, and there is no fixed and accurate method for reference. Therefore, people usually design the neural network to be slightly larger to ensure that there are enough parameters to extract the necessary features, even if this method has the risk of overfitting.
If we can sort the neurons in the network based on their "contribution", then we can remove the neurons with a lower rank to get a smaller and faster network.

2.3 Vector angle analysis

There are many methods to define attributes to select neurons that should be eliminated. The seminal work of pruning the training network uses the output of neurons in the two-stage pruning process, which is performed by inspection. Properties such as relevance, contribution, sensitivity, badness, and distinctiveness have been described in detail elsewhere [3]. Here, we choose the distinctiveness.

Vector angle analysis is one of the methods to judge the distinctiveness of hidden neurons [4]. The same hidden neuron will extract the same feature in different samples, and different hidden units should extract different features for the same input sample. Vector angle analysis is to analyze the similarity of the output results of different hidden neurons to the same input sample. The similar two columns of data indicate that the corresponding two hidden neurons extract similar features, and their functions are similar, so they can be merged while the opposite two columns of data indicate that the results of the corresponding two hidden neurons cancel each other out, so they can be deleted at the same time.

The output results of multiple input samples in the hidden layer can form a matrix. The neural network designed in our model uses Sigmoid as the activation function in the hidden layer. The output value of the activation function is in the range from 0 to 1. Therefore, if we use the cosine similarity [5]:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

the vector angle between two vectors must be between 0 and 90 degrees. To obtain a result between 90 and 180 degrees, we add an offset of -0.5 to the result of the activation function Sigmoid. Negative values will appear, and the calculated vector angle will cover the range of 90 to 180 degrees.

After calculating the vector angle, the general processing rule is that two vectors with a vector angle less than 15 degrees, we think their effects are similar, so you can delete one of the hidden units and put the parameters (weight and bias) are added to another hidden unit; two vectors with a vector angle greater than 165 degrees, we believe that their effects are mutually exclusive, and their effects cancel each other out during the calculation process, so you can delete these two directly Hidden neurons.

When a hidden layer is large, and a hidden layer contains many hidden neurons, multiple hidden neurons may appear to meet the condition that the vector angle is less than 15 degrees. In this case, the two groups with the smallest vector angle can be preferentially selected for merging. Another rigorous processing method is to try to merge any two hidden neurons in turn, then compare the results of each cropped model, and select the group of hidden neurons with the best results to merge. For a plurality of hidden neurons that meet a vector angle greater than 165 degrees, the two groups with the largest vector angle can be selected for merging.

2.4 Genetic Algorithm

Determining the effectiveness of neurons is an ill-posed problem [6] because we estimate their importance by observing the activation of neurons rather than defining them quantitatively. In this case, we use the genetic algorithm as a verification method to select valid neurons in the input.

Genetic Algorithm is a random classification search and optimization method that imitates the evolution mechanism of biological evolution in nature, resulting in Darwin's evolution theory and Mendel's genetic theory. In essence, it is an efficient, parallel, hierarchical search method, which can automatically acquire and accumulate knowledge about the search space during the search process, and at the same time adaptively control the search process to find the best solution. The implementation process of the genetic algorithm is just like the evolution process in nature. First, look for a way to "digitize" the potential solution to the problem. [8] Then initialize a population with random numbers, and the individuals in the population are these digital potential solutions. Next, after an appropriate decoding process, the fitness function is used to evaluate the fitness of each gene. Use the selection function to select the best choice according to a certain rule. Do crossover and mutation then the offspring is produced. The genetic algorithm does not guarantee that you can obtain the optimal solution to the problem, but the biggest advantage of using genetic algorithm is that you do not have to understand and worry about how to find the optimal solution, but simply "deny" some individuals who do not perform well.

2.5 Experiment details

We want to know whether the method of distinctiveness analysis is feasible for pruning hidden neurons within this dataset. In most cases, the accuracy of the model will decrease after pruning, what we want to do is to reduce the number of hidden neurons while keeping the accuracy drop within an acceptable range. If we prune too many hidden neurons at a time, the network may be severely damaged and cannot be recovered. We prune one pair of hidden neurons with the smallest or largest angle each time, then do training again. Repeat these steps until there is not any pair of hidden neurons that need to be pruned. Besides, we let the threshold angle increase then as our expectations, the number of hidden neurons will be reduced. We want to find what degree of the threshold angle is most suitable.

In the data compression task, we use the genetic algorithm. The networks have the same input as the pruning task and the output indicates whether the input data is recognized as dry sclerophyll (output=1) or not (output=0).

Table 1: specification of networks

	Method	Inputs	Hidden	Outputs	Loss function	Optimizer
Hidden neuron pruning	NN	17	Changing	2	cross-entropy	Adam
Compression	GA	Changing	14	2	cross-entropy	Adam

The default training epochs and learning rates are 1000 and 0.01. In our data set, we have 190 samples, which are split into training and testing sets, with sizes of 180 and 10 samples respectively. For both cases, we use 10-fold cross-validation to avoid overfitting and improve the reliability of test results.

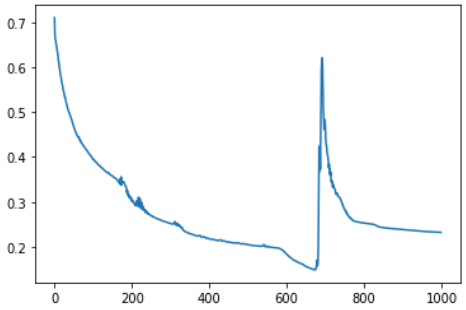
For the genetic algorithm, the DNA size is the number of features in input which is 17 in this study. The population size, crossover rate, mutation rate, and generation size are defined as 100, 0.8, 0.002, 50 respectively. Each chromosome is a binary list of 17 genes with 1 represent keeping this input feature and 0 as dropping this feature. Then we can translate each chromosome into a new dataset. We conduct a new neuron network with the number of input neurons equals to the number of remaining features. And apply 10-fold cross-validation to obtain the test accuracy. The fitness function is a combination of two parts with a certain proportion. The first part is the test accuracy multiplied by 100 with the new dataset created by the chromosome while the second part is the number of features we dropped. We use a proportion of 1:0.2, which means that we only allow a 0.2% reduction in test accuracy for each feature removed. The selection method we used is the roulette wheel selection. The probability of each chromosome being selected is its fitness divided by the sum of the fitness of the population. The higher the fitness, the higher the probability of being selected. Then for each selected parent, randomly choose another individual from the population. Perform uniform crossover at a 0.8 crossover rate and random mutate at a 0.002 mutate rate. Comparing the fitness of patent and offspring. If the child gets higher fitness, replace parent with its child otherwise keep the parent.

3 Results and Discussion

3.1 Benefit of data pre-processing

In the beginning, we try to classify without normalization, we get the results as shown below in the left. Though it does not influence the accuracy dramatically. If normalization is not performed, the difference in the value of different features in the feature vector will be large, which will cause the objective function to become "flat". In this way, when performing gradient descent, the direction of the gradient will deviate from the direction of the minimum value and take many detours, that is, makes the training harder, this is shown by the loss plot. After normalizing, training time will be reduced, and it will be more possible to make a good classification model.

Figure 1: The loss plot and results without normalization vs with normalization



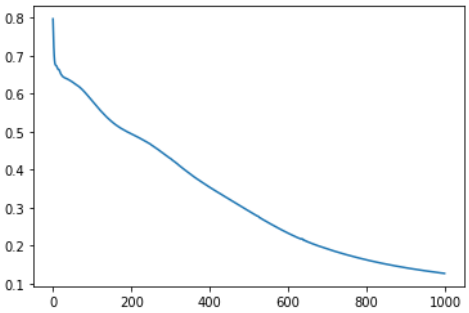


Table 2: The Confusion matrix for training and testing without normalization vs with normalization

	CORRECT	FALSE +VE	FALSE -VE
training	147	5	8
testing	19	4	7

	CORRECT	FALSE +VE	FALSE -VE
training	153	4	3
testing	23	3	4

3.2 Hidden Neuron Pruning Experiment

In the beginning, we simply calculate the vector angle between each pair of hidden neurons and prune all the hidden neurons at one time. Then, we improve it by doing iteration, that is, prune one pair of hidden neurons at a time then repeat train – pruning – train – pruning. The table below shows that the influence of different threshold angles measures the number of reserved neurons and the test results with and without iteration. The number of hidden neurons is initialized to 14.

Table 3: Distinctiveness analysis on the hidden layer

Threshold angle	No. of reserved neurons without iteration	Test accuracy without iteration	No. of reserved neurons with iteration	Test accuracy with iteration
0°	14	63.16 %	14	63.16 %
5°	12	63.16 %	11	56.67 %
10°	12	63.16 %	9	66.67 %
15°	12	63.16 %	7	70.00 %
20°	10	60.92 %	6	66.67 %
30°	8	63.33 %	4	43.33 %

It is obvious that if we prune the hidden neurons without iteration, the network is damaged and cannot resolve. This is because, after a learning process, the continued learning of new knowledge by the neural network will cause damage to the existing knowledge. But the network can resolve and accept new knowledge and become a better model. Simultaneously pruning all or most of the architecture will affect the overall performance of the entire network architecture. Because of this, if the features extracted from the input are severely damaged, even a neural network with enough depth cannot do prediction well. Thus, we decide to use hidden neuron pruning with iteration and with 15° threshold angle, since it can reduce the number of hidden neurons but not greatly reduce the accuracy.

Then we add the method to our network. In this experiment, the purpose is to observe the changes in testing accuracy with different numbers of hidden neurons in training and we try to find out which numbers of hidden neurons are the most appropriate for our dataset. When the current number of hidden neurons decreases with the number of pruned neurons, the accuracy of the test represents performance expectations.

Table 4: Test accuracy with different number of hidden neurons

Hidden neurons	Training accuracy	No. of reserved neurons	Test accuracy after pruning
14	96.96 %	12	60.53 %
12	96.14 %	10	61.58 %
10	95.15 %	7	62.63 %
8	90.76 %	6	63.68 %
6	89.71 %	4	66.84 %
4	83.51 %	4	66.84 %
2	77.19	2	62.63%

From the observations above, we find out that the training accuracy keeps falling when we reduce the number of hidden neurons from 14 to 2. Since more hidden neurons can extract more features from the input, and lead to less loss and higher accuracy when training. The test accuracy increases when we reduce the number of hidden neurons from 14 to 4

and decrease when we set 2 hidden neurons. And when the number of hidden neurons become 4, it will not prune any more. The increase at first is because of the overfitting problem. Sometimes, the machine is too entangled with this error value, it wants to reduce the error to be smaller, to complete its learning mission of this batch of data. Therefore, it passes almost every data point, so that the error value will be smaller. But it not always good. Because it cannot successfully predict other data except training data. Hidden neuron pruning makes the network have a certain degree of sparsity, which can reduce the synergy between different features. And the parameters of neurons in the entire network are only partially updated, eliminating the weakened joint adaptability between neurons and enhancing the generalization ability and robustness of the neural network. However, when the number of hidden neurons is set to 2, the fitting ability of the neural network is insufficient, that is, the neural network can only fit the specific features of a few samples, so the test accuracy decrease. Thus, for our dataset, 4 hidden neurons are optimal.

3.3 Genetic Algorithm Experiment

With the settings described in the experiment detail, the model reaches a fitness of 73.96 the 5^{th}, 7^{th}, and 10^{th} input features.

Table 5: The comparison between the original data and compressed data

Method	Number of inputs	Accuracy on the test set	Time cost
NN	17	61.58 %	17s
GA	5	73.68 %	85394s

As we expected, the accuracy will be decreased after doing data compression. But in fact, it improves quite a lot on accuracy. One of the reasons is the original data has a problem of overfitting. We have 17 features but only 190 samples. Furthermore, even though we have 17 inputs in our dataset, we encode the feature Aspect into 4 inputs and the number of independent inputs is much less than that. for example, 4 inputs are coded for Aspect. Also, the 7 inputs (T1-7) from Landsat are all associated with different wavelengths in the same location, which may not be completely independent variables. Thus, the number of independent variables is much less than 17. And some features can be indicated by another one or more features. As the result of the genetic algorithm shows, aspect, altitude, slope degree does not have much effect on our prediction, these features are all related to the topographic position which we think is important for our model. Again, T1 – T7 are all dependent, so we just keep one to represent them generally. Once we do the data comparison, we reduce the noise which affects the training and avoid the overfitting problem, so the test accuracy increases a lot. It is unexpected but reasonable.

Notably, the training of a genetic algorithm is time-consuming since we should use 10-fold cross-validation each time we calculate the fitness. For each generation, we need to run 10-fold cross-validation at least population size times. And since the neuron network used to calculate accuracy is randomly initialized, each time the calculated accuracy and fitness will change slightly, it is very hard to get a stable result. We can only find a better result rather than the best result. This is the disadvantage of the genetic algorithm.

4 Conclusion and Future Work

I used satellite imagery data from the New South Wales State Forest and supplemented it with auxiliary data from aerial photography and other available information. And the data is used to do classifications for dry sclerophyll forest supra-type using the neuron network. In our work, our prediction model achieves a 70% testing accuracy. We attempt to apply hidden neuron pruning with iteration and improve the model by about 5% accuracy. It is not much. Besides, the pruning by thresholding distinctiveness angle only considers the individual performance of each neuron while it would break the global feature constructed with multiple neurons at the same time. It does not show its advantages clearly in our model. As our consideration, this method is suitable for some complicated datasets. The performance of the genetic algorithm method reaches an acceptable test accuracy. But it is very time-consuming, it needs almost one day to run. One of the reasons is that its processing does not require gradients, which results in a huge time cost for mutation operations. Furthermore, it cannot guarantee the optimal solution since each time the accuracy is not exactly the same. Due to the limited size of the data set, the initialization of weights strongly affects network performance. Therefore, we think the regularization method [8] will be more suitable for our problem.

5 References

1. Milne, L. K., Gedeon, T. D., and Skidmore, A. K. “Classifying Dry Sclerophyll Forest from Augmented Satellite Data: Comparing Neural Network, Decision Tree & Maximum Likelihood.” In Proceedings Australian Conference on Neural Networks (pp. 160-163) (1995).
2. Bustos, R. A. and Gedeon, T. D. “Decrypting Neural Network Data: A GIS Case Study.” In Artificial Neural Nets and Genetic Algorithms (pp. 231-234). Springer, Vienna (1995).
3. Gedeon, Tamas D. "Indicators of hidden neuron functionality: the weight matrix versus neuron behaviour." Proceedings 1995 Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems. IEEE. (1995)
4. Gedeon, T. D., and D. Harris. "Progressive image compression." IJCNN International Joint Conference on Neural Networks. Vol. 4. IEEE. (1992)
5. Neuron Network Pruning- Distinctiveness, https://www.jianshu.com/p/da25ea24988a
6. Shu, L. and Jo, P. “Structure simplification of neural network for smile classification.” Australian Journal of Intelligent Information Processing Systems (2019)
7. Detailed explanation of genetic algorithm, https://blog.csdn.net/u010451580/article/details/51178225
8. Lee, Honglak, et al. "Efficient sparse coding algorithms." Advances in neural information processing systems. (2007)