Performing feature selection with a decision tree, characteristic inputs, and an evolutionary algorithm to improve neural network accuracy on GIS data

Christopher Bradley
Research School of Computer Science,
The Australian National University,
Canberra ACT 2601
Christopher.Bradley@anu.edu.au

Abstract. This study aims to compare the predictive accuracies of a neural network, decision tree, and characteristic inputs for predicting forest type, and determine if feature selection can be used to improve the neural network accuracy. Specifically, I predicted whether or not each of 184 pixels belongs to the dry sclerophyll forest type based on 14 different terrain attributes. I found that without feature selection, the neural network performed very poorly with 63.9% accuracy, the characteristic inputs achieved 68.9% accuracy, the decision tree achieved a 69.6% accuracy, and a basic decision rule achieved 71% accuracy. I found I could do feature selection using the decision tree or characteristic inputs to improve the neural network accuracy to 71.8%, and evolutionary feature detection improved the neural network accuracy to 73.3%. So, the best model came from a combination of evolutionary feature selection and a neural network.

Keywords: Forest type classification, neural network, characteristic inputs, evolutionary algorithm, feature selection

1 Introduction

Manually classifying forests using traditional field research techniques is extremely time consuming, and hence too costly for large regions [1]. However, other attributes of the land can be easily obtained and used to predict the forest classifications. For example, dry sclerophyll forests tend to have lower rainfall and poorer soil quality than wet sclerophyll [2]. Satellite imagery at different wavelengths, known as Landsat spectral bands, can also be useful for mapping specific land features such as vegetation slopes and biomass content [3].

Neural networks are powerful tools for making predictions from large amounts of data, but they suffer from the fact that it is hard to interpret why a neural network makes a given prediction. This means that neural networks are less trustworthy than rule-based systems, and often not used for decision-making purposes despite having a higher predictive accuracy. Because of this, lots of research has gone into ways to extract interpretable decision rules from neural networks and achieve similar predictive accuracy [4]. One of these ways is known as a decision tree. A decision tree simply evaluates a Boolean equation, e.g. rainfall > 400mm/year, and then branches in two directions depending on the outcome of this equation, and evaluates more Boolean equations until finally arriving at leaf node.

Another method of extracting decision rules involves finding the 'characteristic inputs' to a neural network. These are the inputs where the network is the most confident in its prediction. Finding the characteristic inputs allows us to simplify the neural network by replacing it with a function that finds the distance from a given input to each characteristic input [5]. This function is much easier to interpret and validate than the original neural network because it is a simple distance measurement, as opposed to a bunch of arbitrary weights. In general, decision trees and characteristic inputs used to interpret a neural network have lower predictive accuracies than the neural network itself, but are still useful because of their interpretability.

Feature selection is another technique that makes interpreting a neural network easier. The goal of feature selection is to select the minimum number of input features necessary, while maintaining or even improving the accuracy of the neural network [6]. This makes the network easier to explain because there are less features that the neural network can draw from to make its prediction. Using feature selection can actually improve network accuracy because the network focusses its attention on learning from the most important features and is less likely to learn false relationships. Feature selection can generally be broken down into 4 use cases: Idealized, Classical, Improving prediction accuracy, and Approximating original class distribution [7]. "Idealized" means finding a minimum set of features that seem relevant, "classical" means trying to optimise some criterion function, and I believe the other two classes are self-explanatory. This study focusses on the third class, "Improving prediction accuracy", specifically I wanted to see if I could use feature detection to improve the neural network such that it had a predictive accuracy as good as, or better than the other classifiers – the decision trees and characteristic inputs.

One way to perform feature selection is by using a genetic algorithm – a type of evolutionary algorithm [8]. A genetic algorithm works by creating a population of individuals with each individual having a different chromosome. The chromosome is a representation of the features selected by that individual. These individuals are then 'evaluated' to determine their predictive accuracy, and some are included for the next generation, some are modified slightly, some are combined through a process we call 'mating', and some are discarded. The idea is to simulate real life evolution to guide a directed search of the feature combinations. Otherwise, a naïve brute force approach would have to evaluate $2^{\#features} - 2$, to find the best combination. This is an exponential function that quickly becomes infeasible.

The purpose of this study is to compare the predictive accuracies of a neural network, decision tree, and characteristic inputs for predicting forest type, and determine if feature selection can be used to improve the neural network accuracy.

2 The Dataset

For this study, I used geographical data from Nullica State Forest in NSW provided by Professor Tom Gedeon [1]. This data contains information on the slope degree and aspect, the topographic position and altitude, the average rainfall and temperature, the type of geology, and the strength of Landsat spectral bands 1 through 7. This data was collected for 184 pixels, with each pixel being 30mx30m. Each pixel was also further characterized as a certain type of forest; Scrub, Rainforest, and Dry/Wet/Wet-dry Sclerophyll. These forest types are the output classes we want to predict. The data was provided in a preprocessed form with each value scaled to between 0 and 99. The purpose of pre-processing is to remove irrelevant features and scale different features to similar magnitudes because this makes it easier for the neural network to learn.

The aspect attribute represents the compass direction the slope is facing. This is a circular function which cannot be represented with a single linear range, so I used two separate inputs. I used the sin of the aspect to represent a roughly linear scale from west to east, and the cos of the aspect to represent a roughly linear scale from south to north. A side-effect of this transformation is that flat pixels got encoded as facing north, which is not ideal, but this flatness is also encoded in the slope degree. I discarded the original aspect column as it added no additional information.

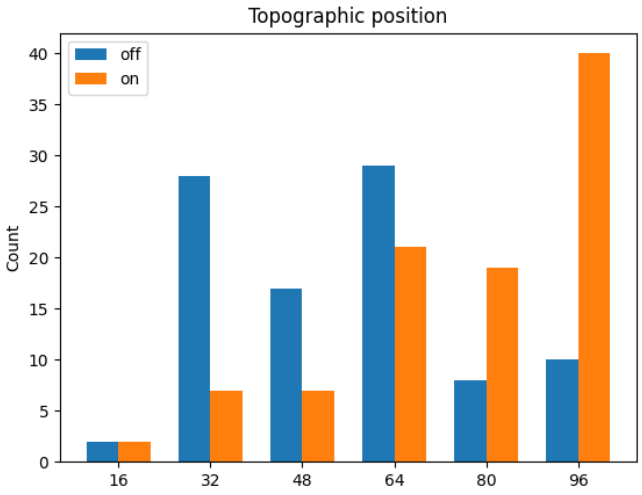
The topographic position is a categorical variable representing how far up a hill the pixel is. The data has six values from 0-100 in intervals of 16 representing a gully, lower slope, mid-lower slope, mid-upper slope, upper slope, or a ridge. I kept this as a numerical representation, it is similar to the altitude but on a local scale. I also left the slope degree, altitude, temperature and rainfall as is, to avoid introducing unnecessary bias.

I chose to ignore the geology descriptor attribute because the purpose of this study was to see if I could explain the predictions, and no information was available on what the geology descriptors were describing. So, fundamentally I could not properly explain any predictions that relied on the Geology descriptor attribute. The Landsat attributes were not matching the other distributions, so I rescaled them to 0-100. After scaling, the distributions were still highly skewed. I considered removing outliers but opted against it because I started with such a small dataset.

The forest types were originally encoded with 0 meaning the pixel was not of that forest type, and 90 meaning it was of that forest type. I changed this such that pixels of the correct forest type were represented by the value 1 to simplify probabilistic equations. I removed all output classes except the Dry Sclerophyll output class so that I could compare my results with those reported in the original paper from 1995 [1].

I visualised these distributions using side by side column graphs showing the number of inputs of a given value that corresponded to each output. This showed which inputs were likely to be useful for predicting the output, and which ones weren't. The more distinct the different output distributions were, the more likely the input is useful. One of my initial findings was that the topographic position had quite distinct outputs, with most of the Dry Sclerophyll classes having a topographic position of 80 or 90, meaning they were on an upper slope or ridge. I have shown this distribution in Figure 1 below, and the rest of the distributions can be seen by running the provided code.

Fig 1. Topographic position distribution. The orange bars represent inputs corresponding to a dry sclerophyll output, and the blue bars represent outputs that are not dry sclerophyll.



Based on this visualisation I created a simple decision rule "If Topographic position is greater than 70, then the output is Dry Sclerophyll, otherwise it isn't Dry Sclerophyll". This decision rule had a predictive accuracy of 71%. I used this as a benchmark to compare against my other classifiers.

Table 1. Decision rule confusion matrix.

	Topographic Position < 70	**Topographic Position > 70**
Dry Sclerophyll On	76	18
Dry Sclerophyll Off	37	59

3 Neural network classification

I used a simple 3-layer neural network with 14 input neurons, 16 hidden neurons, and 2 output neurons as my classification benchmark. I used CrossEntropyLoss as my loss function, Adam as my optimizer [9], and a learning rate of 0.01. I arrived at these hyperparameters through extensive experimentation. I used one output neuron to represent a Dry Sclerophyll pixel, and the other output represented that it is not a Dry Sclerophyll pixel. Since the neural network output a linear function, I took the max of these two outputs to make a distinct decision.

I split the data into a training and test set to ensure the network wasn't overfitting and could make reasonable predictions on unseen data. I used 174 training samples and 10 testing samples. I did this to try to make the most of the small dataset, and also have meaningful accuracies - e.g. 70% means exactly 7 correct and 3 incorrect predictions.

For evaluation, I initially used a fixed seed of 0, but I later changed to running the entire training/testing process on 100 different input seeds and recording the mean accuracy and standard deviation. I did this because the initial seed has a significant impact on how well the neural network learns and the resulting testing accuracy - the standard deviation was consistently greater than 10%. For a long time, I maintained a training accuracy of 80%-90% but struggled to get my testing accuracy would not go above 60%. I had a major break-through after adding 5-fold cross-validation to the training process - this gave an average testing accuracy to 63.9% and a standard deviation of 13.8%. I think using cross-validation was important because it helped to prevent overfitting, but still allowed me to train on a large number of training samples. I also tried out early stopping and backtracking to the epoch with the highest testing accuracy. These techniques improved the training and testing accuracy to above 90%, but decreased the validation accuracy meaning the network was overfitting and performing worse on unseen data.

I did a bunch of experimentation with the number of hidden neurons because the neural network should theoretically be able to able to achieve 71% accuracy with just 1 neuron, since the linearly separable decision rule could achieve this accuracy. To simplify the model, I limited the input features to just the one feature, Topographic position, since I knew this feature was a good predictor from the previous tests. I found that I could achieve 71.8% accuracy using 16 hidden neurons, but the accuracy steadily decreased as I decreased the number of hidden neurons. I believe this decrease in accuracy is due to the neural network having less opportunities to guess a good neuron configuration since there are less neurons. An interesting extension to this study would be to implement neuron pruning to help verify this assumption and decrease the number of hidden neurons. This would also make the network easier to explain. It is notable that the neural network's accuracy of 71.8% is better the decision rule at 71%. I believe this is because the network has lots of neurons, it can make different predictions for different values, so the predictions do not have to be linearly separable.

I had an interesting finding that a network with 16 neurons and 100 epochs performed similarly to a network with 64 neurons and only 25 epochs – both networks achieved ~72% accuracy. The network with 64 neurons has the advantage that training was much faster, however the network with 16 neurons has the advantage that it should be more explainable. I chose to stick to 16 neurons in the interest of keeping the network as simple as possible. I also found that increasing the number of testing samples from 10 to 20 made the average testing accuracy decrease from 71.8% to 70.8% and the standard deviation also decreased from 14.9% to 11.2% . This means the model was giving slightly worse predictions, but these predictions were slightly more reliable. This is expected because more testing samples means I can evaluate the model better, but it also means less training samples so less opportunity to learn.

4 Decision tree classification

I implemented a decision tree using scikit-learn to provide a rule-based classification benchmark. I used the same training and testing ratio as the neural network and the same number of samples, so that I could compare the two. I used varying tree depth to see how the accuracy changed with the number of rules. I found that increasing the depth of the tree caused overfitting, and the best results came from a decision tree with just 1 rule. Specifically, whether the topographic position was greater than 72. This matched up with my initial findings from the data visualisation. Interestingly, the average testing accuracy came out to be 69.6% which was slightly lower than my decision rule at 71%. I think this is probably because some of the train/test splits were suboptimal meaning the decision tree did not learn the optimal decision rule, although further work is needed to verify this.

Table 2. Decision tree trained on 100 random seeds, with each iteration having 174 training samples and 10 testing samples.

Depth	Testing Accuracy	Standard Deviation
1	69.6%	14.2%
2	65.5%	15.5%
3	64.3%	15.8%

5 Characteristic inputs

I implemented a method of finding characteristic inputs that was proposed by Zhu [5] and inspired by Gedeon [10] to choose input features for the neural network, and also make separate predictions for the output class. This method works by splitting the data into two sets, one with all the inputs data that corresponded to the Dry Sclerophyll class, and one with all the inputs that did not. Then I take the mean of each attribute to determine the characteristic input for each class. This gives me two rows, one of which I label the 'characteristic off' outputs and the other the 'characteristic on' outputs. I also recorded the overall mean to see the direction and amplitude that each characteristic parameter deviated from the mean. I then found the difference between the characteristic on inputs and the mean, and the characteristic off inputs and the mean, and summed these values to determine which input features had the largest impact on the output class.

I found that the characteristic on and off inputs had the largest differences in topographic position, sin of aspect, and cos of aspect attributes. There were moderate differences in the altitude, and temperature columns, and no significant differences in the rainfall and slope degree columns. The Landsat bands had no significant differences between the on and off outputs.

The results in table 4 below largely conform with what we would expect from nature, with a few exceptions. It makes sense that the characteristic on input has a higher topographic position because Dry Sclerophyll vegetation types grow where there is low moisture and poor soil quality [2], which is more likely to be found up high than down low in gullies, where water flows. The usefulness of the sin and cos of aspect also makes sense. A low 'sin of aspect' corresponds to east facing slopes and a high sin corresponds to west facing slopes, and since this location is on the east coast of Australia east facing slopes are likely to get more rainfall and water moisture. So west facing slopes are likely to be drier and have more dry sclerophyll forests. Similarly, northern slopes get more sun because this is in the southern hemisphere, which means northern slopes are likely to be drier than southern slopes and have dry sclerophyll vegetation. This idea matches up with the results from the cos of aspect characteristic inputs because a higher 'cos of aspect' represents a north facing slope, and the characteristic on input had a much higher 'cos of aspect'.

It was unexpected that rainfall had little impact on the characteristic inputs and if anything, it seems that the characteristic on input had a higher rainfall. This is unexpected because it goes against the idea that dry sclerophyll forests exist in drier regions. But the results are not significant, so this finding may be due to random chance. The characteristic inputs also had unexpected results for the temperature attribute. The characteristic on input had a significantly lower temperature than the characteristic off input. Intuitively this does not make sense because hotter temperatures dry things up. I do not have an answer for this right now, but it would be an interesting follow up question.

Table 3. Characteristic inputs. The 'overall' row corresponds to the overall mean of all values, the 'characteristic on' and 'characteristic off' correspond to characteristic inputs.

	Sin of aspect	Cos of aspect	Altitude	Topographic position	Slope degree	Rainfall	Temperature
Overall	56	56	33	65	53	39	54
Characteristic off	59	52	30	55	53	38	57
Characteristic on	54	60	37	76	53	41	51
Summed difference	5	8	7	21	0	3	6

Landsat Bands							
	T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**
Overall	13	13	11	52	27	49	19
Characteristic off	13	12	9	53	27	49	20
Characteristic on	14	13	12	51	27	49	20
Summed difference	1	1	3	2	0	0	2

To make predictions with the characteristic inputs I used the variance between the characteristic inputs and the inputs I wanted to predict. The variance is calculated from the sum of square differences: Take the difference between the two inputs, square each value, and sum these values together. In statistics we would often divide this value by length – 1 to get the standard deviation, but in this case we just want to compare the variance with the characteristic on input and the variance with the characteristic off input. Then, the characteristic input with the lowest variance can be used to predict the output class. If the characteristic on output gives a lower variance then we predict that it is a Dry Sclerophyll pixel, but if the characteristic off output gives a lower variance then we predict that it is not a Dry Sclerophyll pixel. This is clearly a more interpretable classification technique than a neural network since it is basically just a distance measure, we are simply asking the question "is the input sample closer to the characteristic off input or the characteristic on input?". Using this technique, I found that the characteristic inputs had a predictive accuracy of 68.9%. I should also note that I did not split the data into training and test sets for the characteristic inputs since we are taking the mean of everything, so overfitting should not be an issue. What could be an issue however is outliers since these may heavily skew the mean. I did not use any outlier removal techniques in this study, but this could be something to try in future.

To select features using the characteristic inputs I found the standard deviation of the summed differences and used this standard deviation to set a threshold and include features that exceeded this threshold. I found the standard deviation to be 5.3. I arbitrarily selected multipliers of 1, 1.5, and 2 to multiply by the standard deviation before applying the threshold, to get some different feature combinations. I found that a multiplier of 2 meant that just the Topographic position feature being selected, so this resulted in an average testing accuracy of 71.8% like before. The other two multipliers both resulted in a lower average testing accuracy, so it was better to train with just the 1 input feature.

Table 4. Characteristic feature selection. The abbreviations are as follows: SA=Sin of aspect, CA=Cos of aspect, AL=Altitude, TP=Topographic position, SL=Slope degree, RA=Rainfall, TE=Temperature, T1-T7=Landsat band tm1-tm7.

Threshold	**SA**	**CA**	**AL**	**TP**	**SL**	**RA**	**TE**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**Accuracy**	**Standard Deviation**
Std x1	0	1	1	1	0	0	1	0	0	0	0	0	0	0	63.0%	15.0%
Std x1.5	0	1	0	1	0	0	0	0	0	0	0	0	0	0	68.8%	14.8%
Std x2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	71.8%	14.9%

I also tried using the characteristic inputs to predict the output of the neural network when the neural network was trained on all features. The idea behind this was to help explain which features the neural network was relying on the most. I found similar results to those shown in table 3 – the neural network was also heavily relying on the topographic position. I chose to revert to just training the characteristic inputs on the original data because I think this is easier to implement in general since you don't need a neural network. I also think it makes more sense to do feature selection from the raw data instead of a secondary source like the neural network.

6 Evolutionary algorithm

I used the python library DEAP to implement a genetic algorithm that guided a search of all the feature combinations I could use as inputs to the neural network. I encoded the genes in a 14 bit chromosome with each bit representing a feature. If the bit was a 1 then the feature was included as an input to the neural network, but if the bit was a 0 then it wasn't. I used single point crossover, and flip bit mutations with a mutation rate of 0.05. I used a tournament mechanism for selection with a tournament size of 3. These hyper parameters were chosen somewhat arbitrarily, but gave good results. The two most important parameters were the population size and number of generations. I set these to 10 during testing and increased them to 100 for the proper runs. This took about 20 minutes to run on my machine.

Originally, I created the individuals with every gene having an equal chance of being on or off. I found that this was resulting in feature combinations with a lot of genes turned on, and the average accuracy was lower than the expected 71%. Since I knew that I could get at least 72% with just 1 gene turned on, I chose to initialise the individuals with just 1 random gene turned on, and then allowed them to turn on and off as many genes as they pleased. This gave better results since the genetic algorithm quickly identified the chromosome with just the topographic position as a good one with an average accuracy of 72% and a standard deviation of 14.9%. The genetic algorithm further improved on this chromosome by turning on the rainfall and temperature genes giving 73.3% accuracy and 13.6% standard deviation.

Table 5. Evolutionary feature selection.

	SA	**CA**	**AL**	**TP**	**SL**	**RA**	**TE**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**Accuracy**	**Standard Deviation**
Evolutionary	0	0	0	1	0	1	1	0	0	0	0	0	0	0	73.3%	13.6%

7 Conclusion and future work

My neural network achieved a predictive accuracy of 63.9% when trained on all features, my characteristic inputs achieved a predictive accuracy of 68.9%, my decision tree achieved a predictive accuracy of 69.6%, and my simple decision rule based on my data visualisation achieved a predictive accuracy of 71%. Using my decision tree or characteristic inputs for feature selection improved my neural network's predictive accuracy to 71.8%, and using my evolutionary algorithm for feature selection further improved my neural network's predictive accuracy to 73.3%.

I found that using cross-validation while training and choosing a good subset of features were really important for improving the neural network's predictive accuracy. I found the best combination of features for this dataset to be Topographic position, rainfall, and temperature, with Topographic position being by far the most important feature.

In future, it would be interesting to apply these findings to the rest of the output classes. To approach this problem, I would represent the 5 output classes in a single column with values from 1 to 5. This is because I believe there is an ordering to these classes in terms of the amount of vegetation: Scrub < Dry Sclerophyll < Wet-dry Sclerophyll < Wet Sclerophyll < Rain forest. This would mean I would need to consider alternative performance measures compared to non-ordinal classifications, so that I could better estimate the magnitude of the errors [11].

Another future task could be to create an evolutionary algorithm that directly predicts the output instead of just doing feature selection. While doing this, I would like to analyse various measures of complexity such as number of lines of code and implementation time and compare these with measures of performance such as processing time and predictive accuracy. It would be interesting to compare these complexity and performance measurements across the different techniques used in this study – neural networks, decision trees, characteristic inputs, and evolutionary algorithms. This study design could help inform which of these techniques to use for similar experiments in future.

Acknowledgement: I would like to thank Professor Tom Gedeon for providing the data for this study.

References

1. Milne, L. K., Gedeon, T. D., & Skidmore, A. K. (1995). Classifying Dry Sclerophyll Forest from Augmented Satellite Data: Comparing Neural Network, Decision Tree & Maximum Likelihood. training, 109(81), 0.
2. The Australian Government, Wet and dry sclerophyll forests, https://www.bioregionalassessments.gov.au/assessments/27-receptor-impact-modelling-hunter-subregion/2743-wet-and-dry-sclerophyll-forests
3. United States Geological Survey. What are the best Landsat spectral bands for use in my research?, https://www.usgs.gov/faqs/what-are-best-landsat-spectral-bands-use-my-research
4. Hailesilassie, T. (2016). Rule Extraction Algorithm for Deep Neural Networks: A Review. ArXiv, abs/1610.05267.
5. Zhu, Q. (2020). Predicted the authenticity of anger through LSTMs and three-layer neural network and explain result by causal index and characteristic input pattern, 3rd ANU Bio-inspired Computing conference (ABCs 2020), paper 83, 7 pages, Canberra.
6. Deniz, A., Kiziloz, H.E., Dokeroglu, T. & Cosar, A. (2017), Robust multiobjective evolutionary feature subset selection algorithm for binary classification using machine learning techniques, Neurocomputing, vol. 241, pp. 128-146.
7. Dash, M. & Liu, H. (1997), Feature selection for classification, Intelligent Data Analysis, vol. 1, no. 3, pp. 131-156.
8. Manocha, M. (2018), Feature Selection and Pruning in Network Reduction for Classification of Breast Cancer, 1st ANU Bio-inspired Computing conference (ABCs 2018), Research School of Computer Science, Australian National University
9. Kingma, D.P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. CoRR, abs/1412.6980.
10. Gedeon, T. D. & Turner H. S. (1993). Explaining student grades predicted by a neural network, Proceedings of 1993 International Joint Conference on Neural Networks
11. Cruz-Ramírez, M., Hervás-Martínez, C., Sánchez-Monedero, J. & Gutiérrez, P.A. (2014), Metrics to guide a multi-objective evolutionary algorithm for ordinal classification, Neurocomputing, vol. 135, pp. 21-31.