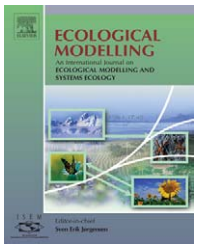
available at www.sciencedirect.com

SCIENCE DIRECT®

journal homepage: www.elsevier.com/locate/ecolmodel



# Machine learning of poorly predictable ecological data

Y. Shan[a], D. Paull[b], R.I. McKay[c,*]

[a] School of Information Technology and Electrical Engineering, University of New South Wales at ADFA, Canberra ACT 2600, Australia
[b] School of Physical, Environmental and Mathematical Sciences, University of New South Wales at ADFA, Canberra ACT 2600, Australia
[c] School of Computer Science and Engineering, Seoul National University, Seoul, Korea

ARTICLE INFO

*Article history:*
Published on line 18 January 2006

*Keywords:*
Genetic programming
Decision trees
Neural networks
Support vector machines
Southern brown bandicoot
Spatial distribution modelling

ABSTRACT

This paper reports on research using a variety of machine learning techniques to a difficult modelling problem, the spatial distribution of an endangered Australian marsupial, the southern brown bandicoot (*Isoodon obesulus*). Four learning techniques – decision trees/rules, neural networks, support vector machines and genetic programming – were applied to the problem. Support vector and neural network approaches gave marginally better predictivity, but in the context of low overall accuracy, decision trees and genetic programming gave more useful results because of the human comprehensibility of their models.



## 1. Introduction

Spatial phenomena and spatial interaction in the real world are highly intricate. This makes mathematical models, which often rely on good or strong theoretical knowledge, very hard to build, since in many cases the available theories are suspect, and at best poor. Machine learning has consequently been used to explore such datasets, and where the data are highly regular and the effects are strong, has yielded valuable insights. However ecological modelling problems frequently combine small datasets, noisy data, and weak domain theories, and thus provide severe challenges to machine learning techniques. This paper presents a comparative study of what can be obtained from a variety of machine learning techniques on one such difficult problem, the distribution of a native Australian animal, the southern brown bandicoot (*Isoodon obesulus*).

The southern brown bandicoot is a small, omnivorous, ground-dwelling marsupial which occurs in southern and eastern Australia. Habitat fragmentation, feral predators and other factors have led to a continuing decline in the distribution and abundance of the bandicoot.

In order to protect it, it is necessary to understand the relationships between its population and these factors. In 1998 and 1999, over 300 sites were surveyed in South Australia, and population data for the bandicoot, and surrounding factors, such as vegetation, soil, fire history and geomorphology were obtained. In this paper, four machine learning techniques (decision tree/rule learning, genetic programming, neural networks and support vector machines) are used to try to identify the potential relationships between the bandicoot and geographical factors.

The rest of this paper is organised as follows. Section 2 describes the modelling problem and underlying dataset, while Section 3 surveys the learning techniques used in this research. The data preparation and experimental setup are described in Section 4, while Section 5 presents the results. The results are discussed in Section 6, and the conclusions are presented in Section 7.

## 2. Bandicoot conservation and survey data

The southern brown bandicoot *I. obesulus* is a small (~0.5–1.5 kg), omnivorous, ground-dwelling marsupial occur-

* *Corresponding author*. Tel.: +82 2 880 9392; fax: +82 2 871 4912.
E-mail address: rim@cse.snu.ac.kr (R.I. McKay).

0304-3800/$ – see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.ecolmodel.2005.11.015

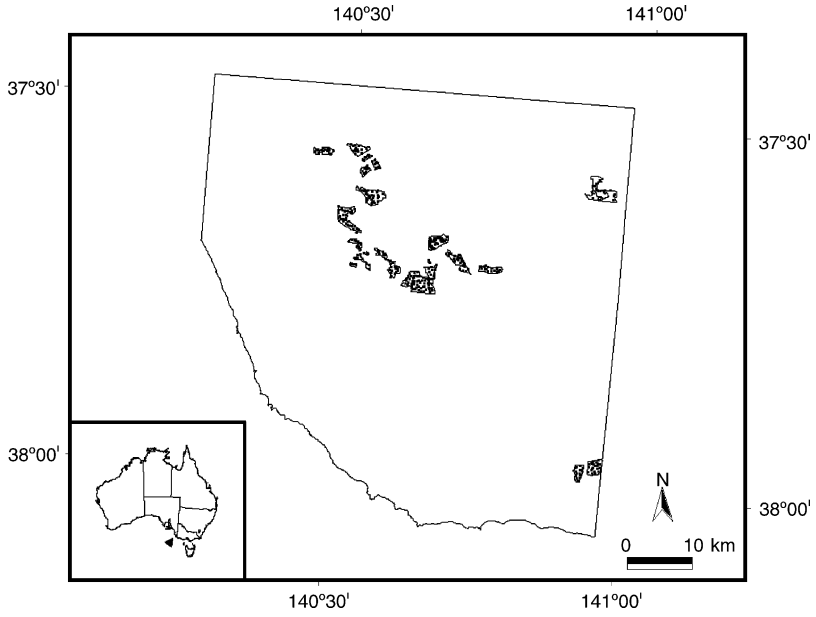



**Fig. 1 – Location of the study area and the 343 survey points used in this analysis.**

ring in southern and eastern Australia. Habitat fragmentation, feral predators and other factors have caused a decline in its abundance and one subspecies, *I. o. obesulus*, which is the focus of this paper, is listed as endangered by Australia's Environmental Protection and Biodiversity Conservation Act 1999.

In order to conserve the southern brown bandicoot, it is necessary to understand the ecological factors that influence its distribution and abundance. In 1998 and 1999, a detailed field survey was conducted for *I. obesulus* in south-eastern South Australia (Fig. 1), and information was gathered at 343 sites on bandicoot abundance and habitat characteristics including landforms, soils, vegetation and fire history.

In this paper, four machine learning techniques are applied in an effort to gain an understanding of the potential relationships between the abundance of *I. obesulus* and features of its habitat.

The field survey allocated sites proportionally to habitat sampling units, which were defined by overlaying digital maps of topography, vegetation and fire using a Geographic Information System (Paull, 1999). To maximize the sample size, a protocol was developed for making a rapid site assessment of *I. obesulus* foraging activity. When *I. obesulus* excavates subterranean invertebrates and fungi it leaves behind distinctive pits or 'iggings' and the number of these reveals the number of food items taken from a site. Counting diggings, therefore, offers a useful surrogate measure of the abundance of *I. obesulus*, which is a difficult species to detect using conventional trapping methods (Paull, 1995). Diggings were counted during active searches of 100 m × 100 m sites. After making a close examination of the ground surface, the number of diggings seen was assigned to one of four abundance classes (Abund) relative to the distance walked throughout the site: (0) no diggings, (1) 1–5 diggings per 100 m, (2) 6–20 diggings per 100 m, (3) >20 diggings per 100 m.

Predictor variables were chosen to reflect aspects of habitat use by *I. obesulus* based on the species' primary needs of shelter from predators and surface soils which contain food. Vegetation structure and soil properties were therefore emphasized in the survey (Table 1).

Landforms (Lform) of the study area influence local drainage conditions, soil formation and vegetation patterns, and they were described in three basic classes: (1) slopes (>3°, measured with a clinometer), (2) flats, (3) closed depressions. The site drainage variable (Drscore) provided information on soil wetness conditions and was assigned to one of five ranked classes based on the classification of McDonald et al. (1990): (1) very poorly drained, (2) poorly drained, (3) imperfectly drained, (4) moderately well drained, (5) well drained. A soil sample was collected at each site at a depth of 10–15 cm, which approximated the depth that *I. obesulus* digs in the study area. In a laboratory, the soil samples were oven dried at 50 °C, sieved to <2 mm (i.e. sand grain and smaller fractions), sub-sampled using a soil splitter, moistened with water and kneaded to field capacity. Soil texture (Soiltext) was then classified using the system for a wet soil bolus described in McDonald et al. (1990) and soil colour (Colour) was determined using a Munsell® Soil Colour Chart (Munsell® Color, 1994).

Vegetation inventories were made to determine plant species richness (SR), dominant species in the canopy layer (Assoc) and dominant ground layer species in the 0–1 m stratum (Gveg). *I. obesulus* uses several vegetation formations but in the study area it is most frequently found where the ground stratum is dense (Paull, 1993, 1995, 2004). To score this variable (Gld) vegetation in the 0–1 m layer was allocated to one of ten classes: (1) 10% cover, (2) 20% cover, (3) 30% cover, ... (9) 90% cover, (10) 100% cover. Detailed assessments were also made of the abundance, height and condition of *Xanthorrhoea australis*, which is a grass-tree used by *I. obesulus* to hide its nests beneath (Paull, 1995). In the study area, *I. obesulus* usually nest beneath *X. australis* with stems between 30 and 80 cm tall and foliage that hangs to the ground to form an enclosed skirt (Paull, 1995, unpublished data). Table 2 summarizes the survey method used to score the abundance, height and skirt condition of *X. australis* (Xanscore).

**Table 1 – Attributes of the survey data**

| Variable name | Description | Range of values | Type of variable |
|---|---|---|---|
| Abund | Abundance of digging class | 0, 1, 2, 3 | Numeric |
| Landsys | Land system | CRL, KLN, MBU, MSA, NGW, YOU | Unordered categorical |
| Lform | Landform description | Depression, flat, slope | Unordered categorical |
| Drscore | Drainage class | (1) very poorly drained, (2) poorly drained, (3) imperfectly drained, (4) moderately well drained, (5) well drained | Ordered categorical |
| Soiltex | Soil texture (10–15 cm depth) | clay, loam, clayey sand, loamy sand, sand | Unordered categorical |
| Colour | Soil colour (10–15 cm depth) | black, brown, very dark grey, dark grey, grey | Unordered categorical |
| SR | Plant species richness | (1) 1–5 species, (2) 6–10 species, (3) 11–15 species, (4) >15 species | Ordered categorical |
| Assoc | Dominant plant species in the vegetation canopy | eucbax (*E. baxteri*), eucobl (*E. obliqua*), eucova (*E. ovata*), eucvim (*E. viminalis*), shrubs (species of *Melaleuca* and *Leptospermum*), reed/sedge (communities of poorly drained areas), other (species of *Acacia*, *Pinus radiata*, *Eucalyptus camaldulensis*, exotic/native complex) | Unordered categorical |
| Gveg | Dominant plant species in the ground layer (0-1 m). | pteesc (*Pteridium esculentum*), xanaus (*Xanthorrhoea australis*), reed, sedge, lepto_mel (species of *Leptospermum* and *Melaleuca*), other (includes species of *Allocasuarina*, *Astroloma* and *Gahnia*) | Unordered categorical |
| Gld | Ground layer (0–1 m) vegetation cover | (1) 10%, (2) 20%, (3) 30%, (4) 40%, (5) 50%, (6) 60%, (7) 70%, (8) 80%, (9) 90%, (10) 100% | Ordered categorical |
| Xanscore | *Xanthorrhoea australis* abundance/height/condition score (see Table 2) | 1–6 | Ordered categorical |
| Litter | Litter depth class, adjusted for percentage litter cover | (1) <0.1 cm, (2) 0.1<0.25 cm, (3) 0.25<0.5 cm, (4) 0.5<1.0 cm, (5) >1.0 cm | Ordered categorical |
| N_fire | Number of fires since 1958 | (1) 0 fires, (2) 1 fire, (3) 2 fires, (4) 3 fires, (5) 4 fires, (6) 5 fires | Ordered categorical |
| F_age | Years since burning class | (1) 0–4 years (2) 5–9 years, (3) 10–14 years, (4) 15–19 years, (5) >19 years | Ordered categorical |

Fire is thought to be an important variable in structuring the habitats of *I. obesulus* (Stoddart and Braithwaite, 1979; Possingham and Gepp, 1996; Paull, 1995). A fire history of each site was therefore obtained from archived forestry records and two variables were modelled: number of fires since 1958 when accurate record keeping commenced (N_fire) and the number of years since last burning (F_age).

Leaf litter on the ground was included in the habitat assessments because it forms a potentially important microhabitat for invertebrates eaten by *I. obesulus*. Litter cover classes used were: (1) <1%, (2) 1–10%, (3) 10–30%, (4) 30–70%,

**Table 2 – Site-based scoring system for suitability of *X. australis* for nesting by *I. obesulus* based on the plant species' abundance, height and 'skirt' condition**

| | >50% of the available ground cover | 10–50% of the available ground cover | <10% of the available ground cover |
|---|---|---|---|
| A: 30–80 cm stem with a well developed skirt | 16 | 8 | 4 |
| B: <30 or >80 cm stem with well developed skirt, or 30–80 cm tall with sparse skirt | 8 | 4 | 2 |
| C: <30 or >80 cm stem with sparse skirt | 4 | 2 | 1 |
| D: any stem height but with negligible skirt | 2 | 1 | 0 |

| Total raw score | *Xanthorrhoea* abundance/height/condition class |
|---|---|
| 0 | 1 |
| 1–5 | 2 |
| 6–10 | 3 |
| 11–15 | 4 |
| 16–20 | 5 |
| >20 | 6 |

The total raw score for a site may include component scores from more than one abundance/height/condition class if the growth form of *X. australis* is not uniform. For the analysis, raw scores were reclassified into six abundance/height/condition classes.

(5) >70%. Litter depth (Litter) was also estimated then multiplied by the mid-points of the litter cover class estimates to obtain an estimate for average litter depth, which was divided into five classes: (1) <0.1 cm, (2) 0.1 < 0.25 cm, (3) 0.25 < 0.5 cm, (4) 0.5 < 1.0 cm, (5) >1.0 cm.

*I. obesulus* is distributed over a range of different land systems, which are areas or groups of areas with a recurring pattern of landforms, soils and vegetation. Undetermined aspects of land systems may influence the distribution of *I. obesulus* and for this reason the variable (Landsys) with six possible values was included.

In this research, we used four machine learning techniques to generate models, aiming to predict the abundance of bandicoots, represented as the density of bandicoot diggings.

## 3. Machine learning techniques

Previous investigations on the dataset using Generalised Linear Modelling had resulted in little understanding of the data. For example, the best forward selection model for the dataset included Gld, Drscore and Xanscore, with a residual deviance of 281.4 and only 97.4 of the sum deviance explained (Paull, unpublished data). Hence it was seen as a suitable candidate for investigation with machine learning techniques, as the potential would be—an understanding of the influences on the distribution of an endangered animal.

Four machine learning techniques, namely decision tree induction (DT, Quinlan, 1993), genetic programming (GP, Koza, 1992, neural network learning (NN, Haykin, 1994) and support vector machines (SVM, Vapnik, 1998) were applied to the dataset. Of these, two (DT and GP) directly yield models with potentially human-comprehensible meaning, while the other two (NN and SVM) do not. Techniques are available for extracting meaningful models from NNs (Tickle et al., 1998), and are being developed for SVMs (Nuñez et al., 2000), but these techniques in general involve a reduction in predictive accuracy.

### 3.1. Decision tree and rule induction

Decision tree induction (and the closely related regression tree induction) is a traditional and well-respected method for generating predictive models from data. It proceeds by recursively partitioning the data according to values of attributes; most algorithms work from the root node of the tree downward, generating progressively more refined definitions for the classes being learnt. Most algorithms are greedy, partitioning the data at each level based on which attribute gives the best value of whatever criterion is in use to judge the quality of data partitions. In most algorithms, the partitioning is continued until a further heuristic criterion, deliberately designed to generate overfitting, is triggered. The resulting tree is then pruned to give optimum generalisation performance on an independent test dataset.

Decision tree induction may be extended to generate decision rules instead: the unpruned decision tree is converted into a logically equivalent rule set, then pruning is conducted on the rule set rather than the tree. Decision rule pruning can permit generalisations which are not available in decision tree pruning, and the resulting rules can sometimes be more comprehensible to human readers.

The comprehensibility of decision tree learning comes at a significant cost: the language learnt by decision tree systems is not universal, so it may be impossible for the decision tree system to accurately fit a dataset simply because it cannot represent the information contained in the dataset.

The experiments in this research were conducted with the C4.5 package (Quinlan, 1993).

### 3.2. Neural network learning

Artificial neural networks are an abstraction from the current understanding of the functioning of the animal nervous system. In the commonest model, synapses are simulated by nodes containing a transfer function (usually nonlinear, and often sigmoid). Incident edges are divided into inputs and outputs; the inputs are summed proportionately according to weights attached to each edge, then the output values are generated by the transfer function. The simplest commonly-used representation, the feed-forward multi-layer perceptron, prescribes a layered architecture in which edges feed only forward from one level to the next; theoretically, only a single hidden (internal) layer is required, and this is the approach used in this work. A wide variety of algorithms are available for training the weights from the data; back-propagation, effectively a gradient descent algorithm for minimising the error function, is perhaps the most commonly used.

The learnt information embedded in a trained neural network is not directly human-comprehensible. Techniques are available for extracting meaningful knowledge from a neural network, but there is no guarantee that the extraction process will preserve the accuracy of the predictions made.

The experiments in this research were conducted using the Tlearn package (Plunkett and Elman, 1997).

### 3.3. Support vector machine learning

Linear support vector machine learning (Vapnik, 1998) aims to find separating hyperplanes, which will separate the dataset as reliably as possible into the distinct data classes. In the ideal case, when the data are completely linearly separable, the hyperplanes will be as far as possible from the nearest elements of the classes. In general, SVM aims to approximate this condition as nearly as possible. Nonlinear SVM replaces hyperplanes with other classes of manifolds, but the basic principle remains the same.

The learnt information embedded in support vectors is not directly human-comprehensible. Techniques are under development for extracting meaningful knowledge, but there is no guarantee that the extraction process will preserve the accuracy of the predictions made.

The experiments in this research were conducted using the LibSVM package (Chang and Lin, 2001).

### 3.4. Grammar guided genetic programming

Genetic programming (Koza, 1992) is a form of evolutionary algorithm. Evolutionary algorithms are motivated by Darwin's theory of evolution by natural selection, and search a problem

space through the interaction of stochastic variation operators (mutation and crossover) and selection operators (often also stochastic). Genetic programming is a specific form in which the problem space is specified (usually) as the set of structures which may be generated from a set of function and variable symbols, usually within a bounded depth. In this paper, we make use of grammar-guided genetic programming (GGGP, Whigham, 1995), a variant in which the problem space is specified by a grammar, usually context-free. Instead of evolving the structures themselves, GGGP evolves their parse trees, but in other respects closely resembles standard GP. Compared with canonical GP, GGGP has the following five advantages.

With the grammar constraint, the closure requirement of canonical GP (that all possible structures have a semantic meaning so that they can be evaluated) is removed, permitting the evolution of more expressive program structure.

The grammar in GGGP provides a natural and formalized way to represent background knowledge, either domain knowledge restricting the form of possible solutions or meta-knowledge about the form of acceptable solutions. With background knowledge, the search space may be dramatically reduced.

Problem-related building blocks, a kind of a priori knowledge, can be represented through the grammar, further improving search efficiency.

During the GP search process, the grammar itself can also be evolved, leading to incremental learning.

During the overall modelling process, the grammar can be readily modified manually and incrementally, to reflect the user's increasing familiarity with the problem.

While these have been of limited importance in this work to the present stage, the second point is the key to further extensions to be discussed in the conclusions.

While GGGP can, in general, represent almost any search space of interest, in the interests of comprehensibility the search space was restricted to be logically equivalent to that used by decision rules (see details below).

The experiments in this section were conducted using Brian Ross' DCTG-GP system (Ross, 2001).

## 4. Data preparation and experimental setup

The original data contained 344 sites (i.e. instances) described in terms of the attributes discussed in Section 3. One of these sites contained a missing value for one attribute. It was deemed expedient to omit that instance rather than needlessly raise the complex issues involved in dealing with missing values.

The class distribution of the instances is highly skewed—there are only 13 instances of class 3 (> 20 diggings per 100 m), 48 of class 2 (6–20 diggings), 123 of class 3 (1–5 diggings) and 159 of class 0 (no diggings).

The aim of the experiments described here was to compare and understand the behaviour of the various learning algorithms. Hence a replicated-trials approach was used, in order to gain an understanding of the overall performance of the algorithms. It is customary, in this discipline, to use n-fold cross-validation for trials, with $n - 1$-fold being used for training, and one-fold for testing, the experiments being replicated n times with a different fold used each time for the testing set. Commonly, the number of folds is set to 10. However, in the context of the small sample size and highly skewed class distribution, 10-fold cross-validation would typically result in the two classes of highest interest being sparsely represented in the testing set, potentially giving highly misleading results. Instead, the dataset was divided into approximately equal datasets (training: 170; testing: 173) 10 times, using random sampling stratified over the classes. This set of ten training and 10 corresponding testing sets was then used for the replications of all experiments.

The highly skewed nature of the data implies that the naïve hypothesis, namely that all instances are in the majority class, has reasonable performance. Thus machine learning techniques are reasonably likely to start with this hypothesis, and attempt to refine it. Unfortunately, prediction of the majority class (i.e. absence of diggings) is not particularly useful in understanding the mechanisms affecting their distribution; in fact, we attach at least equal importance to understanding the causes of the smallest class (high number of diggings). All of the learning methods considered incorporate mechanisms for weighting the importance of different classes; but the methods are different, and the differences could potentially confound an understanding of the differences between the learning algorithms. Hence rather than use the built-in weighting mechanisms, we took the straightforward approach of re-sampling the data, using replication of instances to build a new set of training and test datasets in which the class distributions were equal. These are referred to hereafter as the balanced datasets, in contradistinction to the unbalanced datasets previously described.

### 4.1. *Decision tree/rule setup*

C4.5 data preparation is straightforward, since no data transformation is required (the algorithm is independent of any scaling or other effects). C4.5 was used in its default mode: using the gain ratio criterion for splitting, with the pruning confidence level set at 25%. C4.5 was run once on each of the 10 training/testing dataset pairs, for both balanced and unbalanced datasets, making 20 runs in all. The experiments were then repeated using C4Rules to generate decision rules, for a further 20 runs.

### 4.2. *Neural network setup*

Neural network data preparation is more complex.

Neural networks require numeric data. The values of all categorical variables were mapped to a numeric range.

Efficient learning requires the variables to be scaled to the same range. Hence, all input variables were mapped onto the range [0,1].

Two possible representations for the class were tried:

*Single output*: the four class values were mapped onto the output values 0.0, 0.3333, 0.6666 and 1.0. In testing, the output value was mapped to the nearest neighbouring class value, which was taken to be the neural network predicted class.

*Multiple output*: the network had four output nodes, and classes 3, 2, 1 and 0 were represented by the outputs [1,0,0,0],

[0,1,0,0], [0,0,1,0] and [0,0,0,1]. On testing, the output class was taken to be that corresponding to the highest output activation level.

Neural network architecture is also important. In these experiments, we used an input layer of 13 nodes (corresponding to the 13 independent attributes), an output layer of either 1 or 4 nodes, as described above, and a single hidden layer, fully connected to both the input and output layers. However, the size of the hidden layer is also important. Theoretically, too small a hidden layer will not permit the complexity of the problem to be learnt, resulting in poor training data performance, whereas too large a hidden layer will permit overfitting to the training data, giving poor generalisation to the test data. However there is no a priori method to determine an appropriate size, and unlike decision trees, there is no built-in mechanism equivalent to pruning to optimise the size. Experiments were repeated at each hidden-layer size from 2 to 12 nodes, giving 440 experiments in all (1 or 4 output nodes * balanced or unbalanced * 11 hidden layer sizes * 10 replications). The experiments used the default learning rate of 0.1, and the learning algorithm was trained for 1,000,000 sweeps.

### 4.3. *Support vector setup*

As with the neural network learning, categorical attributes were mapped to numeric values, and all variables were transformed to the range [0,1]. A linear kernel was used because of the reduced parameter-setting requirement; standard parameter values were used, in particular a cost parameter of 1.

### 4.4. *Grammar guided genetic programming setup*

#### 4.4.1. *Target language*

One of the important preparatory steps for GGGP is to identify a suitable target language in which to evolve programs. On one hand, the language should be expressive enough to cover the potential solution space. On the other hand, too general a language may adversely affect the efficiency of execution. This trade-off needs to be carefully considered.

The grammar used for modelling bandicoot abundance is shown in Table 3. According to this grammar, every model generated by our GP search process, starts from

| | |
|---|---|
| S | →if BOOL abundance_3<br>else if BOOL abundance_2<br>else if BOOL abundance_1<br>else absent |

This means that the system tries to find a Boolean statement covering the cases with bandicoot abundance at level three, where the number of diggings is over 20; then another Boolean statement covering the cases with bandicoot abundance at level two, i.e. between 6 and 20; finally another Boolean statement covering the cases with abundance level 1, i.e. from 1 to 5 diggings. All the other cases will be classified into class absent. The remaining part of the grammar is self-explanatory.

In this application, we use the number of misclassified cases as the fitness. This fitness function provides no direct pressure for parsimonious (and hence general) models, although the well-known phenomenon of bloat under evolutionary pressure (i.e. the genome tends to fill up with non-effective code such as "NOT NOT") indirectly provides some degree of parsimony pressure on the effective part of the code. However to fully investigate this aspect, the depth limit in the system was varied over the range 4–11, giving 80 runs. Each run used a population of 1500 for 200 generations. The experiments use tournament selection with a tournament size of 3. Other genetic programming parameters are detailed in Table 4.

**Table 3 – Grammar used in genetic programming experiments**

| | |
|---|---|
| S | → if BOOL abundance_3<br>else if BOOL abundance_2<br>else if BOOL abundance_1<br>else absent |
| EXP | → PRE EXP<br>\| EXP OP EXP<br>\| NUM<br>\| if BOOL EXP EXP<br>\| CVAR |
| BOOL | → BOOL and BOOL<br>\| BOOL or BOOL<br>\| not BOOL<br>\| EXP CP EXP<br>\| ORD CP ORD_VALUE<br>\| NORD in NORD_VALUE_SET |
| PRE | → exp \| sqrt \| log |
| OP | → + \|−\|*\|/\| power |
| CP | →<\|>\|= |
| NUM | → N_fire \|Xanscore\| Litter_vol |
| ORD | → Drscore \|Gld\| SR |
| NORD | → Lform \|Color\| Soiltex\| Assoc\| G_veg\|Landsys |
| CVAR | → ephemeral_const |

**Table 4 – Genetic programming parameters**

| Parameter | Value |
|---|---|
| Terminals and nonterminals | See Table 3 |
| Fitness function | No. of misclassified instances |
| Generation type | Generational |
| Selection scheme | Tournament, size 3 |
| Population | 1500 |
| No. of generations | 200 |
| Initialisation | Ramped half and half |
| Initial max depth | 5 |
| Crossover probability | 0.9 |
| Mutation probability | 0.1 |
| Internal crossover probability | 0.9 |
| Terminal mutation probability | 0.75 |

## 5. Results

The results of the runs are shown in Table 5. Each entry gives the mean and standard deviation, over ten runs, of the error rate (as a percentage) for the particular combination of learning mechanism and learning parameters.

The first point to note is that none of the performances is particularly good. While some of the algorithms fit the training data accurately, this is simply the result of overfitting, as the test-set performance is much weaker.

For the unbalanced-class data, none of the methods is able to reduce the test-set error to below 46.4%; this is only a 9.1% improvement on the 55.5% error of the naïve classifier, which

**Table 5 – Machine learning error rates**

| | Unbalanced training (%) | Unbalanced test (%) | Balanced training (%) | Balanced test (%) |
|---|---|---|---|---|
| Decision tree | | | | |
| Unpruned | 15.1 ± 2.0 | 53.9 ± 4.4 | 5.3 ± 0.9 | 67.4 ± 3.6 |
| Pruned | 27.2 ± 4.2 | 47.8 ± 3.2 | 7.3 ± 1.0 | 66.6 ± 4.1 |
| Pruned rules | 32.3 ± 3.2 | 48.2 ± 3.1 | 9.0 ± 2.1 | 64.1 ± 4.2 |
| Single output neural network | | | | |
| Hidden units | | | | |
| 2 | 37.6 ± 2.8 | 50.3 ± 3.9 | 34.4 ± 4.2 | 58.1 ± 5.3 |
| 3 | 31.4 ± 3.3 | 50.7 ± 4.5 | 27.4 ± 3.2 | 58.2 ± 4.6 |
| 4 | 22.5 ± 4.1 | 51.3 ± 6.1 | 19.3 ± 3.9 | 58.0 ± 8.4 |
| 5 | 17.4 ± 4.0 | 54.3 ± 4.3 | 14.5 ± 3.3 | 59.0 ± 3.4 |
| 6 | 14.1 ± 3.5 | 55.0 ± 4.0 | 11.5 ± 2.6 | 61.5 ± 5.5 |
| 7 | 9.9 ± 4.1 | 54.5 ± 4.7 | 7.7 ± 2.4 | 61.4 ± 4.4 |
| 8 | 6.8 ± 3.3 | 54.9 ± 4.2 | 6.6 ± 1.7 | 60.4 ± 4.5 |
| 9 | 5.5 ± 1.9 | 55.9 ± 4.2 | 5.5 ± 1.7 | 59.2 ± 5.5 |
| 10 | 4.5 ± 2.4 | 56.2 ± 5.5 | 4.3 ± 1.6 | 61.0 ± 5.5 |
| 11 | 4.2 ± 2.0 | 54.6 ± 3.3 | 4.2 ± 2.3 | 57.1 ± 4.9 |
| 12 | 3.0 ± 1.1 | 53.8 ± 5.0 | 4.1 ± 1.4 | 59.1 ± 5.3 |
| Four output neural network | | | | |
| 2 | 41.4 ± 26.4 | 46.6 ± 2.9 | 38.1 ± 19.5 | 61.3 ± 4.2 |
| 3 | 24.4 ± 3.2 | 47.7 ± 3.0 | 28.1 ± 16.7 | 60.1 ± 7.4 |
| 4 | 26.5 ± 24.7 | 49.5 ± 4.2 | 23.2 ± 18.5 | 60.8 ± 5.5 |
| 5 | 36.4 ± 37.2 | 51.4 ± 3.5 | 33.1 ± 29.3 | 64.0 ± 4.5 |
| 6 | 20.0 ± 27.0 | 53.1 ± 4.0 | 17.3 ± 20.4 | 61.8 ± 4.5 |
| 7 | 25.6 ± 37.4 | 52.4 ± 2.6 | 22.2 ± 27.9 | 61.7 ± 6.6 |
| 8 | 46.8 ± 41.3 | 53.1 ± 3.7 | 42.0 ± 35.0 | 61.2 ± 4.9 |
| 9 | 5.2 ± 1.4 | 51.1 ± 2.5 | 6.4 ± 1.4 | 61.9 ± 6.0 |
| 10 | 5.2 ± 1.9 | 50.9 ± 2.8 | 5.3 ± 1.1 | 63.7 ± 4.1 |
| 11 | 17.0 ± 28.8 | 52.3 ± 2.8 | 18.2 ± 30.0 | 62.9 ± 4.7 |
| 12 | 18.4 ± 31.6 | 52.6 ± 3.0 | 18.3 ± 29.9 | 61.7 ± 4.5 |
| Support vector machine | | | | |
| | 42.5 ± 4.4 | 46.6 ± 3.2 | 43.4 ± 6.0 | 57.1 ± 5.8 |
| Grammar-guided genetic programming | | | | |
| Depth bound | | | | |
| 4 | 43.0 ± 2.5 | 49.2 ± 2.2 | 44.6 ± 2.3 | 62.5 ± 3.7 |
| 5 | 40.0 ± 3.0 | 48.0 ± 4.0 | 42.2 ± 2.6 | 62.4 ± 5.8 |
| 6 | 39.1 ± 2.5 | 49.1 ± 3.4 | 39.3 ± 3.8 | 63.4 ± 2.9 |
| 7 | 38.5 ± 2.9 | 48.2 ± 1.9 | 38.5 ± 5.5 | 63.7 ± 4.5 |
| 8 | 37.0 ± 3.1 | 46.9 ± 2.7 | 37.5 ± 5.0 | 61.2 ± 4.3 |
| 9 | 35.6 ± 2.4 | 48.3 ± 3.2 | 35.2 ± 3.6 | 61.9 ± 4.7 |
| 10 | 36.4 ± 2.7 | 46.4 ± 1.7 | 34.4 ± 3.1 | 63.6 ± 4.0 |
| 11 | 36.2 ± 5.0 | 47.7 ± 3.4 | 34.5 ± 1.9 | 63.0 ± 3.8 |

simply classifies all data into the majority class (here 'absent'). Moreover some parameter settings of the single-output neural network classifier actually gave worse performance than the naïve classifier. Of the methods tested, only the single-output neural network is clearly worse than the others.

For the balanced-class data, the performance relative to the naïve classifier (whose error rate is 75%) is somewhat improved. The worst performance, of 66.6% (omitting unpruned decision trees, whose performance is irrelevant) is still 8.4% better than the naïve classifier, while the best, that of the support vector machine is 17.9% better. The 1-output and 4-output neural network methods appear to have changed places, with the former outperforming the latter (though always within the standard deviation) for all architectures. The support vector machine gives equal best performance (with the 1-output, 11-internal-node neural network), but given the high variance of the data, not too much should be read into this.

The results are sufficiently similar, and the variances sufficiently large that pairwise *T*-tests detect few significant differences in the table. However the trends in the data are uniform enough that we may summarise it as a whole, taking into account the difference between comprehensible and non-comprehensible representations, as follows.

For the unbalanced-class data, there is little to choose between the methods, with the exception of the definitely poorer performance of the single-output neural networks.

For the balanced-class data, the non-comprehensible representation methods give marginally better performance than the comprehensible; within the former, 4-output neural networks performed worse than the other methods, while within the latter, genetic programming appeared to work substantially better than decision-tree methods.

It is probably not a coincidence that the support vector machine gave equal best performance on the balanced-class data, and equal second-best on the unbalanced-class data.

**Table 6 – Best test error achieved by each learning algorithm**

| | Unbalanced data | Balanced data |
|---|---|---|
| Decision tree | 42.2% | 59.4% |
| Neural network (single output) | (three hidden) 43.4% | (four hidden) 44.4% |
| Neural network (four outputs) | (2 hidden) 41.0% | (7 hidden) 51.6% |
| Support vector machine | 42.8% | 47.2% |
| Genetic programming | (depth 5) 42.2% | (depth 8) 54.1% |

Although increasing GGGP depth, and to an even greater extent, increasing size of the neural network hidden layer, gave improved performance on the training data, there was little effect on the test set accuracy, implying that the training set effect was almost entirely the result of overfitting.

## 6. Discussion

Although there is some indication in the data that non-comprehensible representation methods may have given slightly better learning performance on the balanced-class data, this effect is absent in the unbalanced-class data. On the other hand, the usefulness of a black-box predictor with an error rate of close to 50% (unbalanced) or 60% (balanced) is open to serious question. With comprehensible representations, since the predictive performance of the classifiers has been validated on an independent test set, we have a guarantee that the information embedded in the classifier has definite, though weak, predictive accuracy. We can gain value from the process by examining the classifiers and attempting to understand the knowledge embedded within them.

### 6.1. *Representative learned classifiers: interpretation*

In this section, we discuss the implications of some of the best models (in terms of testing set error rate and complexity) generated by the decision tree and genetic programming learners. Table 6 shows the best testing-set error rates achieved by each of the algorithms (these results are not presented for comparison purposes, but rather to give a feel for the relative performance of the models analysed).

We will examine three GP-generated rules and one decision tree, displayed in the following table. Rules 1 and 2 were generated by GGGP from the unbalanced training data using depth bound 6 and 5 respectively, and have error rates of 43.4% and 42.2%. Rule 3 was generated by GGGP from the balanced training data using a depth bound of 8, and has an error rate of 54.1%. Rule 4 was generated by decision tree learning from the unbalanced training data, and has an error rate of 42.2%.

#### 6.1.1. *Rule 1*

The conditions for abundance level 3 (Color = brown, Gld > 6, Landsys = KLN) apply to only two sites in the dataset, both of which actually have abundance 3; in fact, land system Kalangadoo is quite sparse, with only 10 instances, so this part of the rule is attempting to predict only a very small number of sites. Gld > 6 is quite sensible, since bandicoots prefer higher densities of ground layer vegetation, which provide protection from predators (but not too high, since that may impede the movement of the bandicoots themselves).

The conditions for abundance level 2 (Assoc = eucobl, Gld > 6) are consistent with a known greater abundance of diggings stringybark eucalypt *Eucalyptus obliqua* associations.

The conditions for abundance level 1 (Xanscore > 1, N_fire < 5, Gld > 6) are consistent with the view that *Xanthorrhea australis* provides predator-shelter for nests.

#### 6.1.2. *Rule 2*

The conditions for abundance level 3 (N_fire = 1, Landsys = KLN) single out 5 of the 10 Kalangadoo sites (the other 5 having N_fire = 2). The former have higher abundance levels than the latter.

**Low error rules generated by genetic programming and decision tree learning**

```
Rule 1: if  (Color = brown) and (Gld > 6) and (Landsys = KLN)))
                then Abundance 3 (Abundant)
        else if  (Assoc eq eucobl) and (Gld > 6)
                then Abundance 2
        else if  (Xanscore > 1) and (N_fire < 5) and (Gld > 6)
                then Abundance 1
        else     absent

Rule 2: if (N_fire = 1) and (Landsys = KLN)
                then Abundance 3 (Abundant)
        else if  (N_fire = 1) and (Lform = slope)
                then Abundance 2
        else if  (Gld IN {7,8})
                then Abundance 1
        else     absent


Rule 3: if       (Soiltex = sand)
                 and ((Color Not= very dark grey) or ((Xanscore > 1)
                 and (Drscore > 4))) and ((G_veg = xanaus)
                  or (Assoc = shrubs)) then Abundance 3 (Abundant)
        else if  (Drscore > 4)
                 then Abundance 2
        else if  (Xanscore > 2)
                 then Abundance 1
        else     absent

Rule 4: if Gld <= 5
                 then absent
        else if  Drscore <= 2
                 then absent
        else if  Gld <= 8 and SR <= 3
                 then Abundance 1
        else if  Gld <= 8 and SR > 3
                 then if  F_age <= 3
                          then Abundance 3 (abundant)
                      else if   Assoc = eucbax
                               then Abundance 1
                               else Abundance 2
        else if  Xanscore <= 1
                  then absent
        else if  Gld > 9
                  then absent
        else if  SR <= 2
                  then absent
                  else Abundance 1
```

The conditions for abundance level 2 are (N_fire = 1, Lform = slope). Landform = slope makes sense because the best drained sites occur on slopes. They are unlikely to ever be waterlogged, which can occur with sites on the other two landform classes (flats and depressions). One reasonable hypothesis is that well drained soil is optimal for the food of bandicoots (subterranean invertebrates) because they would probably drown in the soggy spots. Also, slopes tend to have sandy, friable soils (they are relict sand dunes from the Pleistocene) while flats can be compacted and depressions have heavy organic soils. It is possible that this would affect microhabitat conditions for invertebrates. It could be conjectured that well-drained, friable soils are better for invertebrates than heavy textured, poorly drained soils, and easier for bandicoots to dig in too (i.e. less energetic cost for metabolic return).

#### 6.1.3. *Rule 3*

The greater complexity of the conditions for abundance class 1 of this rule make interpretation difficult. The conditions for abundance classes 2 and 1 (Drscore > 4 and Xanscore > 2, respectively) are consistent with the discussions of drainage and *X. australis* density above.

#### 6.1.4. *Rule 4*

- If Gld <= 5 then absent
  Low densities of ground layer vegetation are associated with bandicoot absence because of the lack of predator protection. This part of the rule covers 53 of the testing instances.
- If Drscore <= 2 then absent
  Poor drainage is also associated with low bandicoot abundance, even if ground layer vegetation is dense. This part of the rule covers 11 of the testing instances.
- If 5 < Gld <= 8 and SR <= 3 then Abundance 1
  Intermediate levels of ground layer vegetation density favour bandicoots. This part of the rule covers 87 of the testing instances.
- If 5 < Gld <= 8 and SR > 3 and F_age <= 3 then Abundance 3 (abundant)
  F_age = 2 or 3 and intermediate levels of ground layer vegetation are both associated with high levels of bandicoot abundance; it is possible that high floral species richness is associated with high levels of vegetation density, abundance and hence high levels of the bandicoots' invertebrate foods.

#### 6.1.5. *Summary*

The rules generated by genetic programming and by decision trees are at least consistent with reasonable hypotheses about the factors influencing bandicoot distribution.

### 6.2. *Further work*

From the results obtained so far, it seems clear that further effort on this dataset would be best concentrated on support vector machines and genetic programming.

#### 6.2.1. *Further work on support vector machines*

Linear support vector machines clearly gave performance at least equal to, and quite probably better than, neural networks. Other kernel functions were not tested in this research, largely because of their strong and non-robust dependence on additional tuning parameters. While the relatively poor performance on this problem of all learning methods so far tested would not lead to optimism on the likely performance of other kernel functions, we believe it is worth continuing the work in this direction.

#### 6.2.2. *Further work on grammar-guided genetic programming*

Grammar-guided genetic programming performed highly competitively on the unbalanced-class problem, and only a little worse than the non-comprehensible representations on the balanced-class problem. However, GGGP is much more flexible, as regards representation, than the other methods considered. It is characteristic of evolutionary methods such as GGGP that they impose no assumptions on the form of the fitness function; conversely, then, the fitness function imposes no constraints on the representation used. Moreover, GGGP, since it describes the search space via a grammar, can handle any search space for which a suitable grammar and semantics may be defined; in the case of the particular package, DCTG-GP, used here, it is simply a matter of defining a logic-grammar and its associated semantics. This permits us to extend the grammar used in two further directions.

*6.2.2.1. Expressivity versus comprehensibility.* The language used in this work was chosen to provide high comprehensibility, at the expense of expressiveness, not permitting mathematical combination of numeric attributes. We could have taken the opposite choice, and use a language expressively equivalent to that used by neural networks and support vector machines, at the expense of comprehensibility. But these extremes are not the only options. In future work, we will investigate grammars in which the internal structure is given by Boolean combinations, but the final relationships incorporate linear or nonlinear mathematical relationships between the numeric attributes. We hope in this way to generate search spaces with sufficient expressive power to generate more accurate explanations, but sufficiently simple for human comprehension.

*6.2.2.2. Relational learning.* All of the representations used in this work suffer from a further limitation: they are propositional or attribute-based, rather than relational. That is, they are able to generate models which predict the class of an instance from the values of its attributes, but they are unable to take into account the relationships between this instance and any other instances.

It is generally believed in the Australian zoological community that fire is a controlling influence on the distribution of terrestrial marsupials such as bandicoots. In the case of bandicoots, this makes a lot of sense. Fire is known to modify the structure of vegetation in which bandicoots live, and may impact upon the invertebrate food supply (Paull, 1999).

However standard modeling techniques applied to this dataset have not demonstrated such an effect. One possible explanation is that wildfires may have two countervailing effects on the distribution of the bandicoot. Large-scale wild-

fires may lead to extinction, whereas small scale fires may well lead to habitat improvement. If these effects are in approximate balance, the influence of fire would only be visible to modelling techniques able to represent spatial effects. Hence, in this ecological modelling problem, spatial and temporal factors should be explicitly considered. In GGGP, it is straightforward to take spatial and temporal factors into consideration. Whigham (2000) showed that it is quite promising to incorporate spatial relationships into GGGP for ecological modelling. We plan further experiments in this area, investigating both spatial and temporal relationships, in our immediate follow-up research.

Another focus of future work is spatio-temporal representation languages. An appropriate language bias is a crucial component of learning in high dimensional problem spaces, particularly where, as here, the amount of available data is very limited. Hence, the choice of spatio-temporal representation is crucial.

A further strand of the work will focus on evolvability of the underlying grammar, permitting incremental learning and the transfer of meta-knowledge between related problems.

## 7. Conclusions

We investigated the performance of a range of machine learning methods in generating models from a species distribution dataset. The conservation problem underlying the dataset is of some importance, and methods to extract an understanding of the data are highly desirable. However the dataset has so far proven highly resistant to analysis.

Where the comprehensibility of the modelling is not an issue, and predictive accuracy is all that is required, the results suggest that support vector machines at least equal, and quite possibly outperform, the other methods. However, they yield opaque models, whose value is questionable when they are associated with extremely high error rates.

Of the methods considered, only decision trees and genetic programming directly yield comprehensible models. Genetic programming generated models with performance at least equal to decision tree learning in both predictive accuracy and comprehensibility, while providing the option to incrementally trade off expressibility and comprehensibility. Equally important, genetic programming potentially allows the incorporation of relational information which may be crucial to a full understanding of this dataset.

## Acknowledgements

We would like to thank ForestrySA for access to the study sites and archived fire records. We would particularly like to acknowledge the actions of the authors of the relevant software systems – Ross Quinlan, Jeff Elman, Chih-Chung Chang, Chih-Jen Lin and Brian Ross – in making their software freely available to researchers.

## References

Chang, C.-C., Lin, C.-J., 2001. Lib. SVM: A Library for Support Vector Machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Haykin, S., 1994. Neural Networks, A Comprehensive Foundation. Prentice-Hall Inc., New Jersey, 696 pp.

Koza, J.R., 1992. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA, 819 pp.

McDonald, R.C., Isbell, R.F., Speight, J.G., Walker, J., Hopkins, M.S., 1990. Australian Soil and Land Survey Field Handbook. Inkata Press, Melbourne, 198 pp.

Munsell® Color, 1994. Munsell® Soil Color Charts, Macbeth Division of Kollmogoran Instruments, New Windsor, New York.

Nuñez, H., Angulo, C., Catala, A., 2000. Rule Extraction from Support Vector Machines. In: Verleysen, M. (Ed.), Proceedings of the 2000 European Symposium on Artificial Neural Networks, Bruges, Belgium, pp. 107–112.

Paull, D.J., 1993. The distribution, ecology and conservation of the southern brown bandicoot (*Isoodon obesulus obesulus*) in South Australia. MA Thesis, University of Adelaide, 251 pp.

Paull, D.J., 1995. The distribution of the southern brown bandicoot *Isoodon obesulus obesulus* in South Australia. Wildl. Res. 22, 585–600.

Paull, D.J., 1999. A survey of the Distribution and Abundance of the Southern Brown Bandicoot in the South East of South Australia. Working Paper No. 1999/1, School of Geography and Oceanography, University College, Australian Defence Force Academy, Canberra, 22 pp. ISBN 0 7334 0561 4, ISSN 0819-3460.

Plunkett, K., Elman, J.L., 1997. Exercises in Rethinking Innateness: A Handbook for Connectionist Simulations. MIT Press, Cambridge, MA, 313 pp.

Possingham, H.P., Gepp, B., 1996. Assessment of fire regime options for the southern brown bandicoot *Isoodon obesulus* in South Australia using Population Viability Analysis in Fire and Biodiversity. In: The Effects and Effectiveness of Fire Management. Biodiversity Unit, Department of Environment and Heritage, Canberra, Australia, pp. 149–153.

Quinlan, J.R., 1993. C4.5: programs for machine learning. In: Morgan Kaufmann series in machine learning. Morgan Kaufmann, San Mateo, California, 302 pp.

Ross, B.J., 2001. Logic-based genetic programming with definite clause translation grammars. New Gener. Comput. 19 (4), 313–337.

Stoddart, D.M., Braithwaite, R.W., 1979. A strategy for the utilization of regenerating heathland by the brown bandicoot (*Isoodon obesulus*). J. Anim. Ecol. 48, 165–179.

Tickle, A.B., Andrews, R., Golea, M., Diederich, J., 1998. The truth will come to light: directions and challenges in extracting the knowledge embedded within trained artificial neural networks. IEEE Trans. Neural Netw. 9 (6), 1057–1068.

Vapnik, V.N., 1998. Statistical Learning Theory. Wiley, New York, NY, 736 pp.

Whigham, P.A., 1995. Grammatically-based genetic programming. In: Justinian, P., Rosca (Ed.), Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications, Tahoe City, California, USA, 9 July, pp. 33–41.

Whigham, P.A., 2000. Induction of a marsupial density model using genetic programming and spatial relationships. Ecol. Model. 131, 299–317.