Neuro-fuzzy methods for environmental modelling

M. Purvis, N. Kasabov, G. Benwell, Q. Zhou, and F. Zhang
Computer and Information Science
University of Otago
Dunedin, New Zealand
phone: +64-3-479-8318
fax: +64-3-479-8311
email: mpurvis@commerce.otago.ac.nz

This paper describes combined approaches of data preparation, neural network analysis, and fuzzy inferencing techniques (which we collectively call neuro-fuzzy engineering) to the problem of environmental modelling. The overall neuro-fuzzy architecture is presented, and specific issues associated with environmental modelling are discussed. A case study that shows how these techniques can be combined is presented for illustration. We also describe our current software implementation that incorporates neuro-fuzzy analytical tools into commercially available geographical information system software.

1 Introduction

The increasing availability of large stores of spatial data has led to demands for improved methods for analysing these data. In particular there is hope that the use of new analytical techniques in connection with "data trawling" operations may reveal hidden relationships that lie buried within these data sets. In recent years there has been interest in this respect in the application of artificial intelligence techniques, such as expert systems, artificial neural networks, fuzzy systems, and genetic algorithms [1]. Each of these approaches has specific advantages when used in the analysis of large stores of data.

# For example rule-based expert systems facilitate the input and refinement of expert knowledge in problem-solving modules. They also enable straightforward tracing of the reasoning process by which a particular outcome is achieved.

# Fuzzy systems extend the notion of ordinary rule-based expert systems to allow for the treatment of vague and uncertain knowledge in a systematic fashion. These systems can be efficiently implemented and can incorporate human-like reasoning concerning qualitative information.

# Neural networks, in contrast with the above, need not be initially supplied with expert knowledge and can be applied to virtually any data set without presuppositions concerning the distribution of the training data [2]. They then can be trained to approximate any continuos function to any desired accuracy, without a need to specify its type. Moreover they can be applied to incomplete or corrupted data and still yield acceptable results. For this reason it can be useful to use neural networks in the initial

stages of an empirical investigation, when little may be known about the nature of the spatial data set at hand. On the other hand the, the reasoning associated with neural networks is a "black box" that is hidden from the user.

# Genetic algorithm methods also require little in the way of *a priori* information and offer a general optimisation approach. But they can be difficult to apply effectively and consume considerable computational resources.

The goal of neuro-fuzzy engineering is to combine these various techniques in a coherent fashion so that their different strengths can complement each other. In the context of large spatial information data sets, the hope is first to perform data trawling without having to make presuppositions concerning the data, and, at the same time, to extract knowledge that can be used in subsequent analysis or operations. Given the large size of these data sets, it may be advantageous to perform data filtering to reduce the size of the data set prior to carrying out connectionist computational analysis, and thus spatial data filtering is also a component of neuro-fuzzy engineering for environmental modelling.

In the following, some basic neuro-fuzzy engineering methods are presented in section 2, an approach to spatial data filtering is presented in section 3, some illustrative experiments are described in section 4, and a description of our software implementation of neuro-fuzzy engineering tools is given in section 5.

2 Neuro-fuzzy engineering

The basic idea behind the neuro-fuzzy engineering approach is to pass the spatial information through a series of steps:

1. Convert real-valued spatial data into a fuzzified representation of the same information.

2. Train the fuzzified spatial information with a three-layer multilayer perceptron neural network. The output of the neural network is taken to be a fuzzy representation of the desired output.

3. The output of the neural network in step 2 is then de-fuzzified to produce individual real values of the desired output.

After the neural network in step 2 is trained to satisfaction, it is analysed in order to extract fuzzy rules that can be used in association with the defined fuzzy membership functions used in steps 1 and 3.

2.1 Fuzzy neural networks

A fuzzy neural network seeks to blend the elements of these fuzzy and neural network computations into a single connectionist architecture. There are several fuzzy neural-network architectures [1,3,4]. The model we employ (FuNN) [1] consists of five layers:

1 input variable layer

2 condition elements (input fuzzy membership function) layer

3 rule layer

4 action elements (output fuzzy member function) layer

5 output variables layer

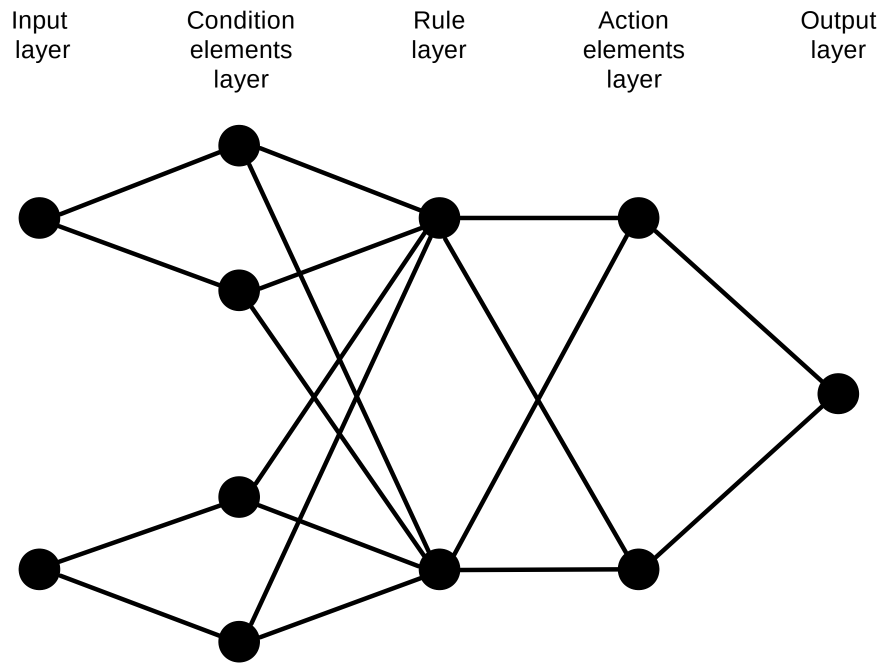


Figure 1. The fuzzy neural network (FuNN) architecture.

These elements are shown schematically in Figure 1. A bias node can also be included in this architecture but is not shown in the figure. Ordinarily, we employ triangular membership functions for the 2nd and 4th layers. In the simplified scheme shown in Figure 1, each of the two inputs can be fuzzified in the condition layer by showing the degree of their membership in a fuzzy set (such as the degree to which they are HIGH or LOW). The number of fuzzy values need not be the same for the various inputs. Thus one input could be connected to two condition layer nodes, and another input could be connected to four condition layer nodes.

2.2 Adaptive training of FuNNs

When the FuNN fuzzy neural network is trained, the connection weights are modified in order to achieve the desired performance. During this process, the method of adjusting the weights for the middle layers (layers 2, 3, and 4) is backpropagation [1]. The weights connecting the outer layers are also adjusted during training, but this is a separate process.

Adjustment of these output layers means that the triangular membership functions that are used in a FuNN to represent fuzzified descriptions of the input variables in the condition elements layer (layer 2) are adjusted during the training of the net in order to arrive at the best performance.

Figure 2 shows the initial membership functions (solid lines) of a variable *x* and the asymmetrical membership functions (dotted lines) after adaptation. These adjustments may be performed for both inputs (weights connecting layers 1 and 2) and outputs (weights connecting layers 4 and 5).

There are several ways in which the fuzzy membership functions could be adjusted during training of a FuNN. We have adopted an approach based on genetic algorithms (GA), adapted from [5]. Since triangular membership functions are employed, only the centres need to be represented in the chromosome of the GA module, making the computation not too expensive. We also restrict the permitted amount of movement of these centres by setting fixed limits along the horizontal axis (Figure 2).

Usually the magnitude of these adjustments during training is small compared with the weight changes occurring in the middle layers. The overall result is that all the weights in a the five-layer FuNN are adjusted during training to obtain optimal results.

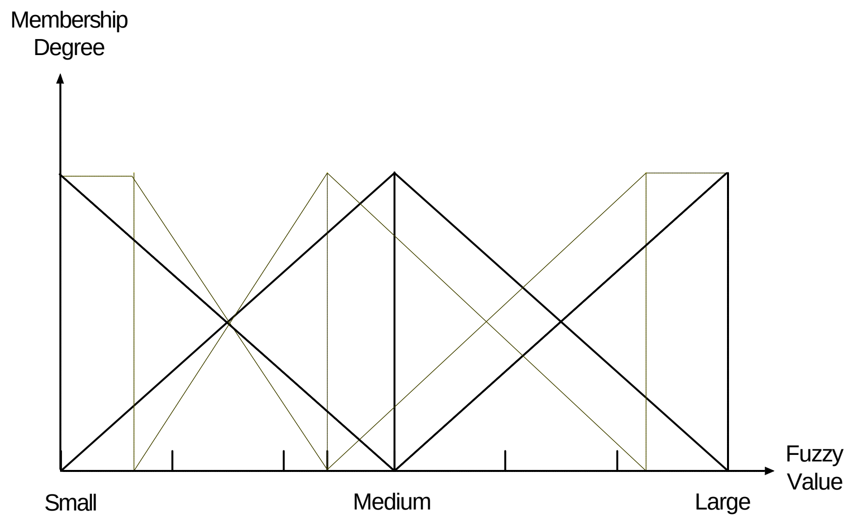


Figure 2. Initial and adjusted membership functions for a fuzzy variable used in a FuNN fuzzy neural network.

2.3 The extraction of fuzzy rules using REFuNN

The 3rd layer of the FuNN architecture is used for the insertion and extraction of structured information in the form of rules. There are several ways to extract rules from this architecture. A simple one (REFuNN) works as follows:

(I) All the connections that contribute significantly (above a certain threshold) to the activation of a neuron A_k and their corresponding rule-layer nodes, R_j, are grouped for subsequent analysis.

(ii) Condition element nodes, C_i, that support the activation of node R_j will be used in the antecedent part of a rule.

(iii) The weights of the connections between the condition-element nodes, C_i, and the rule node R_j are taken as initial relative degrees of importance of the antecedent fuzzy propositions.

(iv) The weights of the connections between rule node R_j and output node A_k define the value for the certainty degree CF_k.

For each real input value, only one fuzzy predicate (the most significant) is allowed in the antecedent part of a rule; less significant predicates are discarded. This results in a set of weighted rules, such as

If <income is HIGH, 0.6> **and** <savings is LOW, 0.57>
Then <credit-risk is MEDIUM, 2.73>

If among this collection of rules, there are several weighted rules that have the same condition elements and the same consequent elements, subject only to different degrees of importance, they are aggregated into a single rule.

The REFuNN algorithm then proceeds to extract simply fuzzy rules from the set of weighted rules. This is achieved by simply removing the weights from the condition elements. Some condition elements, however, have sufficiently large degrees of importance (above some chosen threshold) that they can trigger a rule without support from other condition elements. Such condition elements are used to form additional, separate rules. For example, if we start with an initial weighted rule

If <x_1 is A, 8.3> **and** <x_2 is B, 1.2>
Then <y is C>

and the threshold for separate rules is set at 5.0, then two separate simple fuzzy rules will be formed:

If x_1 is A **and** x_2 is B **Then** y is C
If x_1 is A **Then** y is C

In its simplest form REFuNN does not incorporate predicates representing logical negation. However an option exists to consider negation by including the cases of links with negative weights whose absolute values are above a chosen threshold. In this case the associated input labels in the formed simple fuzzy rules are listed with a "NOT" connective in front.

The resulting rules are somewhat more complicated but can offer better performance when interpreted by a fuzzy inference engine [1].

2.4 Other rule extraction approaches

Another rule extraction approach is to associate each rule layer node with a single fuzzy rule. The strongest positive connection from a condition element node for an input variable to the given rule node, along with the neighbouring condition element nodes, are represented in the corresponding rule. The connection weights of these connections are interpreted as degrees of importance attached to the corresponding condition elements. The connection weights from the rule layer to the action element layer represent the degree of certainty that the given rule will contribute to the given fuzzy output value.

Because each rule layer node is associated with a single rule, there can be more than one element in the consequent part of the rule, and the resulting rules are more complicated than those extracted by REFuNN. An example of rule extracted by this approach is the following:

If <input1 is A, 2.88> **and** <input1 is not B, 5.69> **and** <input2 is not A, 1.57> **and** <input2 is B, 1.66> **and** <input2 is not C, 1.06>
Then <output1 is not A, 2.21> **and** <output1 is not B, 3.98> **and** <output1 is C, 2.08>

An example of how this approach works in practice, along with some variations, will be presented in more detail with the case study discussion is Section 4.

2.5 Fuzzy rule insertion

The FuNN fuzzy neural network can be initialised by inserting known rules and membership functions, and this feature can be used to advantage. Since the training method of a FuNN, like that of multilayer perceptrons, is a local optimisation procedure, the FuNN can become trapped in an unsatisfactory local minimum that results in inadequate performance for the problem at hand. Initialising a FuNN with known rules can be used to avoid operating regions where known and unsatisfactory local minima exist.

In general, it should be possible to add or modify rules by means of human expert knowledge and then insert these new rules into a FuNN prior to training. The objective would be to "seed" or shift the FuNN into a more favourable operating region by incorporating human experts in the neuro-fuzzy engineering loop.

3 Spatial data filtering

The goal of data filtering in this context is to reduce the size of a large data space (so that neuro-fuzzy engineering computation can be more efficiently performed) without sacrificing the discriminating power of the original data. There are two areas where reductions can be made:

# If the data elements are highly correlated, it should be possible to reduce the number of data elements under consideration. This is likely to be true with spatial data sets, where neighbouring locations are likely to contain correlated information. With neural computations this should result in a reduction of the size of the training set.

# If the individual features associated with each data element are correlated, it should be possible to consider a subset of the features and still retain sufficient discriminating power. This should lead to smaller neural net architectures and hence speedier computation. The danger is that if some model-based assumptions concerning the data are made in order to perform feature selection, some relevant attributes may be omitted from the data set under analysis [6].

There are several techniques that have been developed to address these matters [7,8,9]. We have recently adopted the approach of Liu and Setiono [9], which appears to offer promise in both of the above areas for spatial data preparation. In addition, their method performs automatic discretisation, which can help lead to an appropriate selection of the number of condition-layer elements in a FuNN fuzzy neural network.

The method of Liu and Setiono is oriented around the $?^2$ statistic and consists of two phases. In the first phase each attribute *i* is associated with some significance level, say 0.5 at the outset. The data values of this attribute are sorted, and each value is considered to be resident of a (initially single-valued) interval. Then the $?^2$ value is calculated for each pair of adjacent intervals. Starting with the lowest $?^2$ value, adjacent intervals are merged until all pairs of intervals have $?^2$ values greater than the $?^2$ associated with the current significance level for the given attribute. This is done for each attribute. These merged intervals now represent a discretisation of the data set. With the reduced number of intervals, it is possible that there are now inconsistencies (two identical data elements associated with different output class values) in the data set. If the number of observed inconsistencies remains below a user set value, d, the above process is repeated with a decremented significance level for the attributes (and hence a larger tolerated $?^2$ value). At the end of the first phase, the data set is discretised, and the number of data elements has been reduced

In the second phase of Liu and Setiono's method, each attribute *i* is associated with an individual significance level, *sigLevel[i]*, and takes turns for merging. Consistency checking is conducted after each attribute's merging; if the consistency constraint is exceeded, attribute *i* will not participate in further merging. At the end of the second phase, no attributes can be further merged. If during this process an attribute has been merged to a single interval, then it means that attribute is not useful for discrimination and can be dropped from further consideration. Therefore feature selection has been achieved.

When the method of Liu and Setiono was applied to the well-known iris data set [10], which consists of 150 example of irises represented by four attributes that are to be classified into one of three types, 75 elements of the set were selected and a 5% inconsistency in the final data set was allowed. The result was a discretised set comprising only two of the original four attributes, and each of the two remaining attributes was discretised into four intervals. These four intervals for the two attributes could then be used as fuzzy membership values for training on a FuNN fuzzy neural network.

A preliminary example of this method of data filtering to a spatial data example is presented in the next section (4.2).

4 Examples

For illustrative purposes we consider some contrived problems that are conceptually simple, but that can involve some significant computation. They are based on the problem of determining suitable sites for public golf courses in the South Island of New Zealand.

4.1 Using a FuNN fuzzy neural network

For the first example, it is assumed that a suitable location can be determined from the observed data of mean summer temperature, mean annual rainfall, mean altitude, and the distance to the nearest of four principle urban centres of the South Island. Each of the 153,036 1 km^2 blocks of the South Island was taken to be a candidate location for a golf course, and for each block, values for the four attributes were determined and placed in a data set. In order to provide an evaluation mechanism, an artificially "correct" classification value was determined based on a set of plausible rules (refer to [11] for a description of this rule set). The output parameter of golf course suitability was taken to have five possible values, ranging from 0 (very unsuitable) to 4 (very suitable), and a map of the solution set is shown in Figure 3. Note that because of the rigid boundaries of the rules, the solution set is only piece-wise differentiable.

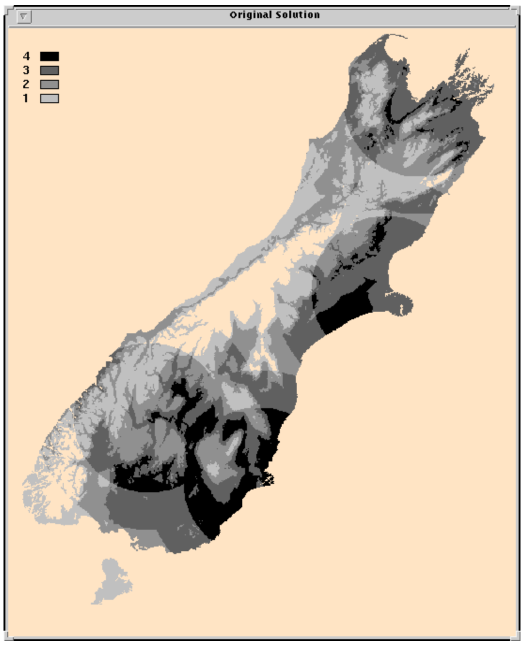


Figure 3. Solution set for golf course suitability (produced by human expert).

A FuNN fuzzy neural network was then constructed to explore the degree to which information could be extracted from this data set. The intention was to consider each of the four input variables to have five linguistic fuzzy values. A typical fuzzy membership function is shown in figure 4.

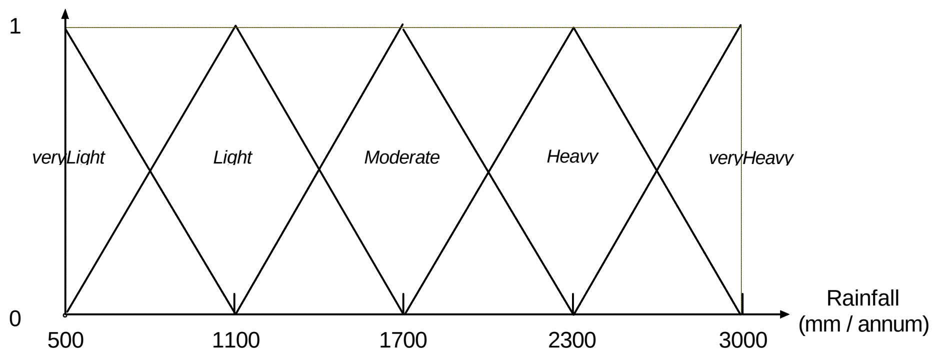


Figure 4. Membership function for the annual rainfall input variable.

The FuNN had 4 input nodes, 20 condition layer nodes, 20 hidden layer nodes, 5 action layer nodes, and one output layer node. It was trained with 1,000 fuzzified, randomly chosen samples over 400 training epics. (For this experiment, genetic algorithm adjustment of the outer layer weights was not performed.) After evaluation over the entire data set, the FuNN was found to classify 85.6% of the blocks correctly and another 14.2% were only off by one membership class. The misclassifications were mostly along the boundaries of the classes. Results are shown in Figure 5.

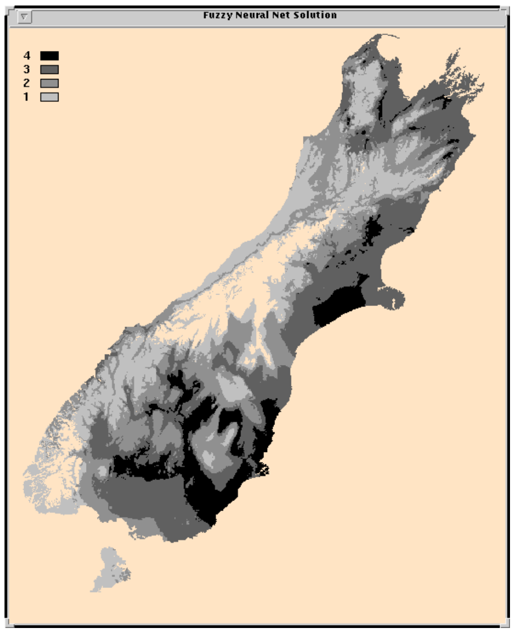


Figure 5. Evaluation of fuzzy neural network over the entire data set.

4.2 Rules extraction

Several methods for extracting rules were applied to this fuzzy neural network and compared. The first method used was the REFuNN method. An example of an extracted rule by means of this process is the following (fuzzy values are here taken to range among A, B, C, D, or E, with A the lowest and E the highest value):

If <altitude is B> **and** <rainfall is D> **and** <temperature is A> **and** <distance is B>
Then <suitability is B>.

When these simple fuzzy rules were evaluated using a min-max fuzzy inference method, 40% of the South Island blocks were classified correctly and 50% were off by one class.

Although such derived rules from REFuNN are relatively simple, the method can yield an unwieldy number of rules. In this case there were 254 rules extracted by the REFuNN method.

In order to arrive at a more manageable number of rules and achieve better performance, a second procedure was attempted:

(i) All node connections in the middles layers of the fuzzy neural network that had a negative weight value were set to zero, and the fuzzy net was then retrained. (This process can be repeated, if necessary).

(ii) Weighted fuzzy rules were then extracted. Rule components were only retained from node connections that had weight magnitudes above a chosen threshold value (4.0 in this case). As a threshold is increased in value, there will be fewer retained rule components in the extracted rules and decreased performance when the rules are interpreted.

(iii) For the inference procedure, a rule fires only if the sum of the weighted contributions from all its inputs is positive.

(iv) The degree to which an output membership value is inferred is determined by calculating the weighted sum (using confidence factors) from all activated rules. In other words, the rules fire "in parallel".

A sample extracted rule using this method is

If < altitude is A, 13.0703> **or** < altitude is not B, 16.0718> **or**
< rainfall is not B, 6.27021> **or** < rainfall is not C, 5.64556> **or**
< rainfall is E, 10.7956> **or** < distance is not A, 7.04855> **or**
< distance is C, 5.62503> **or** < distance is not D, 4.60254>
Then < suitability is A, 6.51655> **and** < suitability is not B, 14.7015> **and**
< suitability is C, 14.5121> **and** < suitability is not E, 5.76596>

The extracted rules are much more complicated than those obtained by the REFuNN method, but they are fewer in number: only 20 rules were derived this time. When this evidential inference method described in step (iv) above was applied, 61% of the blocks were classified correctly and 28% were off by one class.

In order to generate a rule set that involved simpler rules, a third rule extraction procedure was investigated:

(i) Once again the fuzzy neural network was repeatedly retrained. This time only the (positive) maximum element from each data group of the fuzzy neural network along with its neighbouring elements were retained for the next training step; other connection weights were set to zero.

(ii) Rule extraction was similar to the previous approach, but this time only elements above a threshold absolute magnitude were retained.

(iii) Inferencing was similar to that of steps (iii) and (iv) of the previous method.

A sample extracted rule is the following:

If <altitude is A, 16.48> **or**
<rainfall is C, 2.19> **or** <rainfall is E, 2.42> **or**
<temperature is A, 12.59> **or** <temperature is B, 5.10> **or**
<distance is not C, 8.57> **or** <distance is D, 5.02>
Then <suitability is A, 13.47> **and** <suitability is not B, 19.50>

Again there were 20 rules extracted, but they were somewhat simpler than those extracted with the previous method, since they had fewer antecedent components. This rule is relatively simple to interpret, since it says that if the altitude is not very low or the rainfall is high or the temperature is low or the distance to an urban centre is great, then the suitability is very low. When these rules were interpreted, 56% of the blocks were classified correctly and 26% were off by one class. The resulting performance does not quite match the previous method, but the simpler structure of the rules may make them more valuable for human interpretation, since the ultimate goal is to extract rules such that a human expert can possibly derive knowledge or insight from the extracted information. Such added insight may enable the human expert to suggest new or altered rules that can lead to another iteration of neuro-fuzzy information processing and ultimately to improved performance.

4.3 Spatial data filtering

The analyses performed so far have involved computations at each localised geographical point but have not involved the cross-referencing of multiple points in a region. In order to investigate this latter alternative, we considered a slightly altered version of the golf course siting problem. The previous solution rule set involved only a consideration of the altitude at the given site: a high (local) altitude resulted in a low "altitude suitability" and thus had a negative impact on overall site suitability. This time it was determined that an evaluation should be made of the difference between the altitude of the block in question and that of its eight neighbouring blocks. Let *alt_diff* be the maximum difference in altitude between the given block and its eight nearest neighbours. Then the "altitude suitability", *alt_suitability,* (a component of the overall site suitability) was set according to the following conditions:

If *alt_diff*	>= 377.8			**then** *alt_suitability* = 0
If *alt_diff*	>= 276.3	**and** *alt_diff*	<= 377.8	**then** *alt_suitability* = 1
If *alt_diff*	>= 188.4	**and** *alt_diff*	<= 276.3	**then** *alt_suitability* = 2
If *alt_diff*	>= 100.5	**and** *alt_diff*	<= 188.4	**then** *alt_suitability* = 3
If *alt_diff*	>= 0	**and** *alt_diff*	<= 100.5	**then** *alt_suitability* = 4

In other words, a level region was considered likely to be good for golf course siting, irrespective of whether the region was a plateau or a lowland. A new "overall golf course suitability" rule set was then used that, for simplicity, did not involve the distance to a major urban centre of the South Island; it was only a function of the local rainfall, the local temperature, and the altitude difference as described above. Thus to evaluate the suitability of a given block, it would be necessary to evaluate the attributes of that block and those of its eight nearest neighbours.

Now to make this a more realistic data trawling experiment, we supposed that the analysts for this problem did not know the size of the region that should be considered around each block for analysis. So we had them set the size of the region to be 25 blocks as shown in Figure 6.

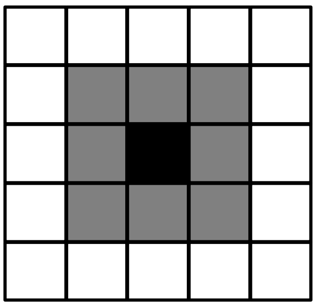


Figure 6. Region of blocks under analysis.

This 25-block region then will represent a window on the spatial data set that is located appropriately whenever a given block (the black square in Figure 6) is to be considered for site suitability. Only the grey-shaded blocks come into play for the determination of "altitude suitability". Thus only the grey-shaded blocks are used to determine the solution set, but the hypothetical analysts do not "know" this.

With the 25-block region for each site under consideration, there are now 75 attributes to be evaluated. This would necessitate a large neural network with significant computational implications. In order to lessen the computational burden that this implies, we applied the data filtering technique of Liu and Setiono described in Section 3.

1,500 blocks were randomly chosen from a central portion of the South Island, and associated with each block was a region like that shown in Figure 6. The inconsistency tolerance (d) was set to be 0.5%, so that up to 7 inconsistent elements would be allowed before attribute merging would be halted. The results are shown in Figure 7.

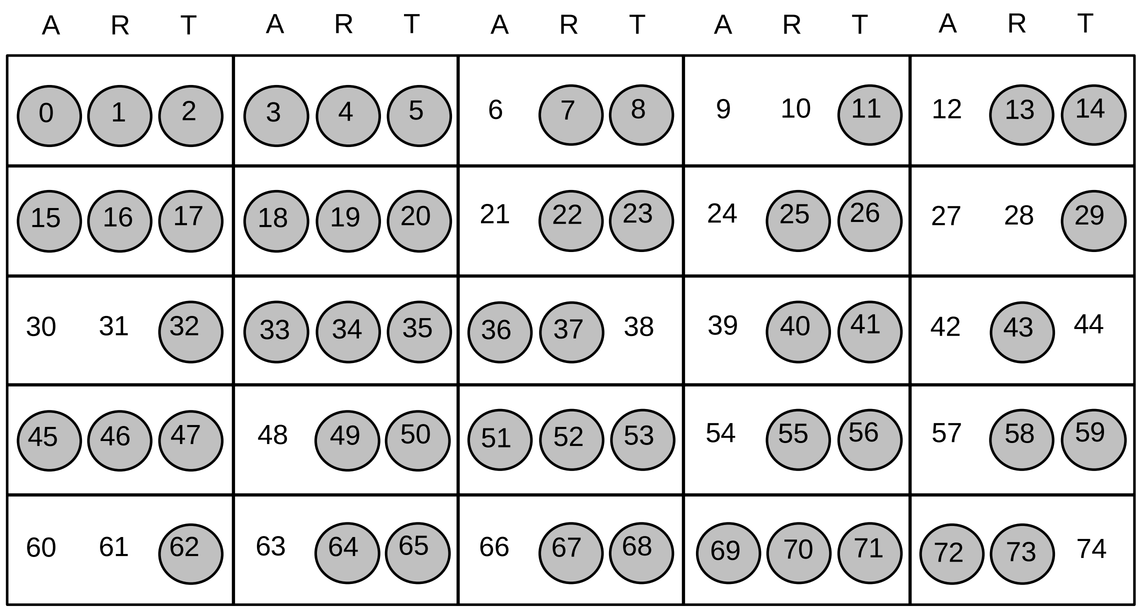


Figure 7. Attributes merged by the data filtering procedure. Merged attribute indices are shown in shaded circles.

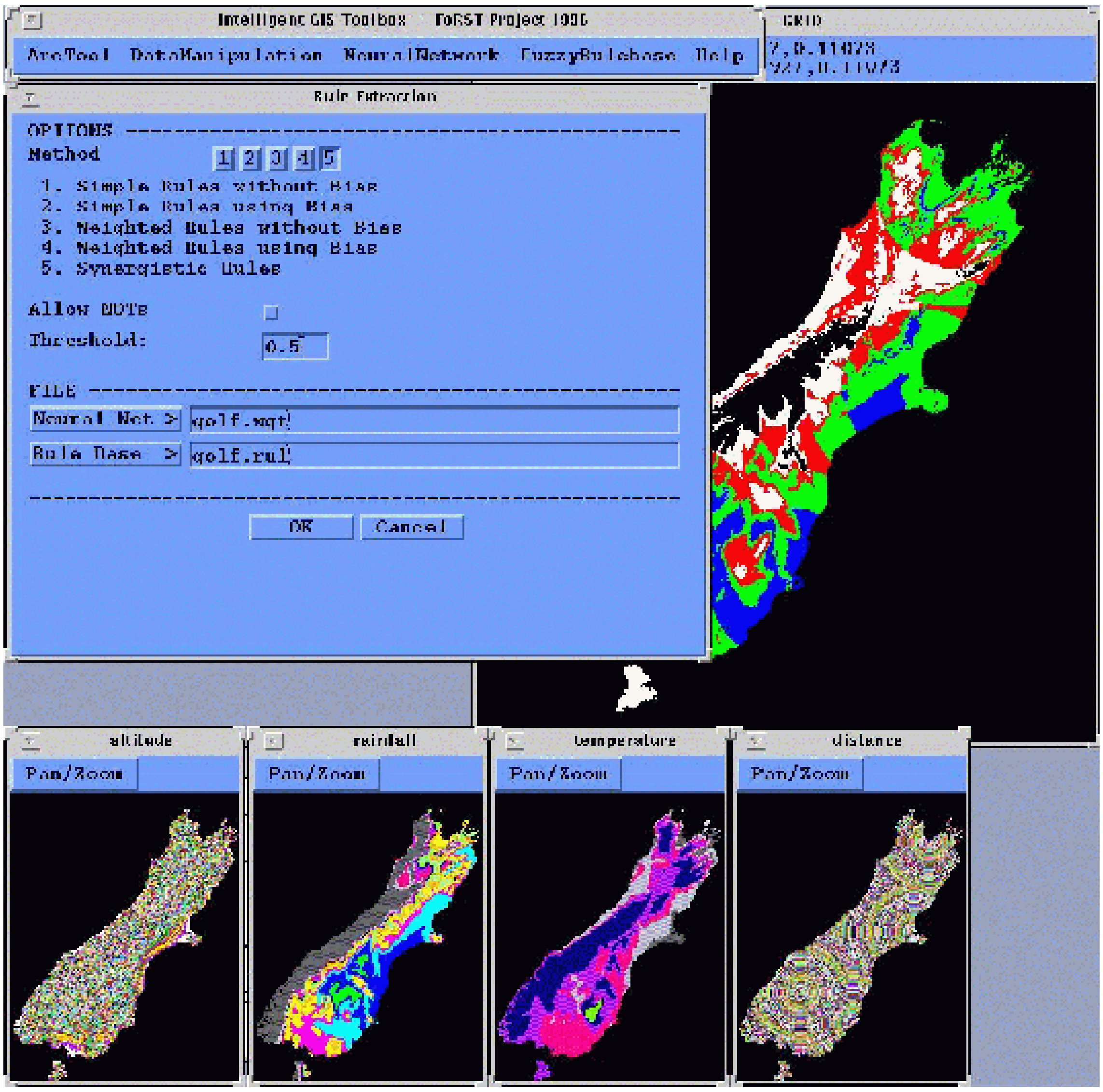
The attributes are labelled with indices from 0 to 74 and are distributed among the 25 blocks as shown in the figure. For each individual block, the altitude attribute is associated with the first index, followed by the rainfall and temperature indices. After discretisation, 104 of the 1,500 original data elements were found to be redundant, so the original data set could be reduced to 1,396. 52 of the original 75 attributes were merged away (discarded), leaving 23 features necessary for classification of the 1,396 data elements to 99.5% accuracy.

The data filtering operation correctly identified the central block's temperature attribute as essential (index 38), but also retained two outer temperature attributes. This can happen due to correlations among the data values and can be reduced if more data elements are considered for the training set. This problem is more evident with the rainfall attribute, which played a relatively small importance in the overall site suitability rule set. Also the variation in rainfall over the portion of the South Island considered for this experiment was relatively uniform.

For the altitude attribute, the central block's attribute was correctly identified as irrelevant, while 5 of the 8 neighbouring blocks's altitude attributes were retained. However 3 presumably essential blocks were discarded by the filtering procedure, and some altitude attributes for blocks outside the neighbouring region were retained. Again, the correlations in the smoothly varying data attributes enabled the data filtering procedure to classify the discretised data set to 99.5% accuracy with the 23 attributes it selected, even though not all of the attributes were the ones that we expected. They are still sufficient to achieve satisfactory classification and offer a greatly reduced data set for subsequent neuro-fuzzy analysis.

5 Neuro-fuzzy engineering with geographical information systems

In order to carry out neuro-fuzzy engineering for environmental modelling, we have integrated our spatial information toolbox with ARC/INFO geographical information system software [12] (Figures 8 & 9). This enables the user to view raster-based map data using ARC/INFO's visualisation tools and to perform toolbox operations from the ARC/INFO interface.

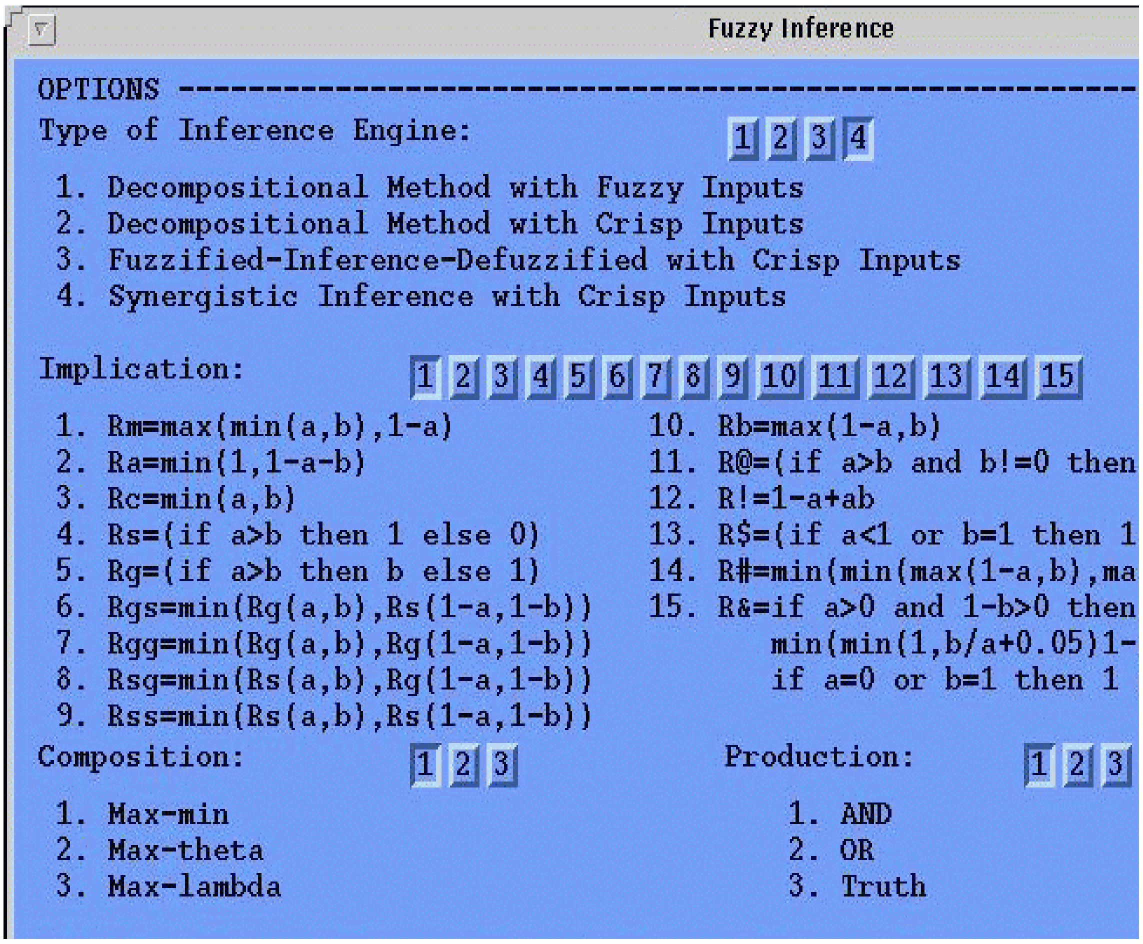


```
OPTIONS -----------------------------------------------
Method              1 2 3 4 5
  1. Simple Rules without Bias
  2. Simple Rules using Bias
  3. Weighted Rules without Bias
  4. Weighted Rules using Bias
  5. Synergistic Rules

Allow NOTs          ☐
Threshold:          0.5

FILE --------------------------------------------------
Neural Net >    golf.wgt
Rule Base  >    golf.rul
-------------------------------------------------------
```

Figure 8. Otago Spatial Information Toolbox in a GIS environment.

Plans for expansion of the toolbox include provisions for various neural net adaptation mechanisms and additional neural net rules extraction approaches that have been reported in the literature [13,14].

As more experience is gained with neuro-fuzzy engineering tools, we believe that they will become increasingly useful for environmental modelling and analysis.



```
Fuzzy Inference

OPTIONS ------------------------------------------------
Type of Inference Engine:                  1 2 3 4

  1. Decompositional Method with Fuzzy Inputs
  2. Decompositional Method with Crisp Inputs
  3. Fuzzified-Inference-Defuzzified with Crisp Inputs
  4. Synergistic Inference with Crisp Inputs

Implication:               1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

  1. Rm=max(min(a,b),1-a)              10. Rb=max(1-a,b)
  2. Ra=min(1,1-a-b)                   11. R@=(if a>b and b!=0 then
  3. Rc=min(a,b)                       12. R!=1-a+ab
  4. Rs=(if a>b then 1 else 0)         13. R$=(if a<1 or b=1 then 1
  5. Rg=(if a>b then b else 1)         14. R#=min(min(max(1-a,b),ma
  6. Rgs=min(Rg(a,b),Rs(1-a,1-b))      15. R&=if a>0 and 1-b>0 then
  7. Rgg=min(Rg(a,b),Rg(1-a,1-b))             min(min(1,b/a+0.05)1-
  8. Rsg=min(Rs(a,b),Rg(1-a,1-b))             if a=0 or b=1 then 1
  9. Rss=min(Rs(a,b),Rs(1-a,1-b))
Composition:               1 2 3          Production:          1 2 3

  1. Max-min                                 1. AND
  2. Max-theta                               2. OR
  3. Max-lambda                              3. Truth
```

Figure 9. Portion of Otago Spatial Information Toolbox menu for fuzzy rule interpretation callable from within ARC/INFO.

References

[1] N. Kasabov, *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*, MIT Press, Cambridge, MA, 1996.

[2] X. Zhuang and B. A. Engel, "Classification of Multispectral Remote Sensing Data Using Neural Networks *vs.* Statistical Methods", *Proceedings of the International Winter Meetings of the American Society of Agricultural Engineers*, Chicago, 1990.

[3] S.-I. Horikawa, T. Furuhashi, and Y. Uchikasa, "On Fuzzy Modelling Using Fuzzy Neural Networks with Back-Propagation Algorithm", *IEEE Transactions on Neural*

Networks, 3(5), 801-806, 1992

[4] M. Gupta and D. H. Rao, "On the Principles of Fuzzy Neural Networks", *Fuzzy Sets and Systems*, 61(1), 1-18, 1994.

[5] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[6] H. Liu and R. Setiono, "A Probabilistic Approach to Feature Selection -- A Filter Solution". *Machine Learning, Proc. of the 13th International Conference*, Bari, Italy, 319-327, 1996.

[7] H. Almuallim and T. G. Dietterich, "Learning Boolean Concepts in the Presence of Many Irrelevant Features", *Artificial Intelligence*, 69(1-2):279-305, 1994.

[8] K. Kira and L. A. Rendell, "The Feature Selection Problem: Traditional Methods and a New Algorithm", *AAAI-92, Proceedings of the Ninth National Conference on Artificial Intelligence,* AAAI Press, 123-128, 1992.

[9] H. Liu and R. Setiono. "Chi2: Feature Selection and Discretization of Numeric Attributes", *Proceedings of the 7th International Conference on Tools with Artificial Intelligence*, Washington D.C., 388-391, 1995.

[10] R. Fisher, "The Use of Multiple Measurements in Taxonomic Problems", *Ann. Eugenics*, **7**:179-188, 1936.

[11] M. K. Purvis, N. K. Kasabov, F. Zhang, and G. L. Benwell, "Connectionist-Based Methods for Knowledge Acquisition from Spatial Data", *Proceedings of the IASTED International Conference on Advanced Technology in the Environmental Field,*Gold Coast, Australia, 151-154, 1996.

[12] Environmental Systems Research Institute, Inc., Redlands CA, 1996.

[13] R. Setiono. "A Penalty-function Approach for Pruning Feedforward Neural Networks", *Neural Computation*, 1997, Vol. 9, No. 1, 301-320, 1997.

[14] R. Setiono. "Extracting Rules from Neural Networks by Pruning and Hidden-unit Splitting", *Neural Computation*, 1997, Vol. 9, No. 1, 321-341, 1997.