ISEIS

Journal of Environmental Informatics 13(1) 56-65 (2009)

Journal of Environmental Informatics

www.iseis.org/jei

# Renosterveld Conservation in South Africa: A Case Study for Handling Uncertainty in Knowledge-Based Neural Networks for Environmental Management

R. Chandra[1,*], R. Knight[2], and C. W. Omlin[1]

[1]*Department of Computer Engineering, Middle East Technical University, Güzelyurt, Turkish Republic of Northern Cyprus*
[2]*Department of Biodiversity and Conservation Biology, University of the Western Cape, Bellville 7535, South Africa*

Received 3 December 2007; revised 17 September 2008; accepted 20 October 2008; published online 12 March 2009

**ABSTRACT.** This work presents an artificial intelligence method for the development of decision support systems for environmental management and demonstrates its strengths using an example from the domain of biodiversity and conservation biology. The approach takes into account local expert knowledge together with collected field data about plant habitats in order to identify areas which show potential for conserving thriving areas of Renosterveld vegetation and areas that are best suited for agriculture. The available data is limited and cannot be adequately explained by expert knowledge alone. The paradigm combines expert knowledge about the local conditions with the collected ground truth in a knowledge-based neural network. The integration of symbolic knowledge with artificial neural networks is becoming an increasingly popular paradigm for solving real-world applications. The paradigm provides means for using prior knowledge to determine the network architecture, to program a subset of weights to induce a learning bias which guides network training, and to extract knowledge from trained networks; it thus provides a methodology for dealing with uncertainty in the prior knowledge. The role of neural networks then becomes that of knowledge refinement. The open question on how to determine the strength of the inductive bias of programmed weights is addressed by presenting a heuristic which takes the network architecture and training algorithm, the prior knowledge, and the training data into consideration.

*Keywords:* biodiversity and conservation, decision support system for environmental management, expert knowledge refinement, gradient descent learning algorithm, inductive bias, knowledge-based neurocomputing, knowledge uncertainty

## 1. Introduction

The Western Cape Region at the southern tip of Africa is one of the world's internationally recognized biodiversity hotspots. It is home to the Cape Floristic Region which exhibits one of the richest floristic diversity and endemism, i.e. approximately 9,000 plant species occur in an area of 90,000 km$^2$ of which 70% are endemic. Renosterveld is a dominant vegetation in the region along with Fynbos. The former tends to occur on fertile and fine-grained shale-granite or silcrete-derived soils where moderate rainfall – from 350 to 650 mm – prevails; the latter prefers sandy nutrient-poor soils with heavier rainfalls. It is easiest to differentiate between the two species by referring to their habitat (geology, rainfall, etc.) instead of species composition. Renosterveld is phenomenally rich in bulb species.

Renosterveld vegetation is unique to South Africa; today, a mere 2% of the original vegetation exists due to rapidly expanding agricultural activities. The threat of extinction makes Renosterveld conservation essential. The decline of Renosterveld makes the exploration of its potential medicinal, commercial and tourism value urgent; however, this process has been slow and questions regarding its economic value remain unanswered due to the lack of information collected about this plant species. The prevalence of Renosterveld in the Western Cape Region of South Africa is shown in Figure 1.

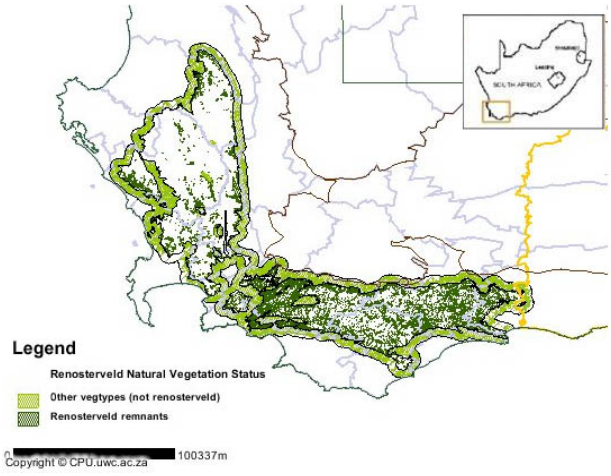


**Figure 1**. Prevalence of Renosterveld in the Western Cape, South Africa.

* Corresponding author. Tel.: +90 392 6612059; fax: +90 392 6612053.
*E-mail address:* rohitash@metu.edu.tr (R. Chandra).

ISSN: 1726-2135 print/1684-8799 online
 doi:10.3808/jei.200900140

Artificial intelligence techniques have been successfully applied to a range of environmental management problems (Davis et al., 1986; Loehle, 1987; Noble, 1987; Cortes et al., 2000; Starfield, 1991). Artificial intelligence systems can support a more natural, simple, interactive, participatory and effective approach to natural resources planning and management. Environmental decision support systems may be problem specific, i.e. tailored to narrow environmental domains but applicable to a wide range of different locations; situation and problem specific systems target specific environmental problems in specific locations (Rizzoli and Young, 1997).

The integration of symbolic knowledge with artificial neural networks is becoming an increasingly popular paradigm for solving real-world applications (Shavlik, 1994; Towell and Shavlik, 1994). The paradigm provides means for using prior knowledge to determine the network architecture, to program a subset of weights to induce a learning bias which guide network training, and to extract knowledge from trained networks. The role of neural networks then becomes that of knowledge refinement. It thus provides a methodology for dealing with uncertainty in the prior knowledge. The paradigm is general and can be applied to any species threat assessment where both expert knowledge and ground truth data are available. Identified high-threat areas can be superimposed on maps generated by geographical information systems.

In this paper, an artificial intelligence paradigm for the development of a biodiversity and conservation biology decision support system is proposed, which takes into account the local expert knowledge together with collected field data about plant habitats in order to identify areas which show potential for conserving thriving areas of Renosterveld vegetation and areas that are best suited for agriculture. The available data is limited and cannot be adequately explained by expert knowledge alone. The expert knowledge about the local conditions and the collected field data is combined in a knowledge-based neural network; a novel measure which takes into account the expert knowledge, the available field data and the neural network architecture and learning algorithm which automatically determines – prior to training – how strongly the neural network ought to rely on the expert knowledge.

Following this brief introduction to the application domain, in Section 2, a general framework for combining symbolic and neural learning called knowledge-based neural networks (KBANN) is discussed. KBANN maps prior knowledge in the form of propositional rules into feedforward neural networks thus providing an explicit inductive bias. The discussion is concluded with the open question: how strong the inductive bias should be chosen in order to achieve good training or generalization performance? A heuristic for determining the strength of this inductive bias is introduced which takes into account the neural network architecture and learning algorithm, the prior knowledge, and the training data in Section 3. This heuristic for determining a good choice of the explicit inductive bias has shown very good results both in cases where the quality of the prior knowledge was excellent (Snyders and Omlin, 2000) as well an application where the prior knowledge did not explain the sparse given data well (Omlin and Snyders, 2003). Preliminary results for this environmental management application were discussed in (Chandra and Omlin, 2005). The heuristic is applied to the problem of conservation of Renosterveld in Southern Africa in Section 4; the expert knowledge about habitats that allow the vegetation to thrive is incomplete in the sense that it does not accurately predict the presence of Renosterveld. The heuristic for determining the strength of the inductive bias outperforms both random and standard choices for the inductive bias. The paper ends with conclusions from this work and possible directions for future research in Section 5.

## 2. Expert Knowledge and Inductive Learning

### 2.1. Preliminaries

Expert knowledge about habitats in which species thrive can be expressed in the form of rules. Such rules can be implemented in expert systems or used to guide analytical learning methods such as explanation-based learning. The objective of analytical learning methods is to find a hypothesis which fits both the expert knowledge and the given samples of ground truth. They provide logically justified hypotheses which have been arrived at through deductive inference. They have the advantage that they can learn from sparse data; however, the logical justifications are only as valid as the expert knowledge they are based on. They are of limited use if the expert knowledge is either incomplete or even incorrect. Expert knowledge in the biodiversity field is often incomplete due to the variability of habitats.

Analytical and inductive learning works well under different conditions. However, most practical learning problems lie somewhere between the two extremes of plentiful data without prior knowledge and perfect prior knowledge with scarce data. Combining inductive with analytical learning methods thus holds the promise of exploiting the strengths of the two approaches while alleviating their respective weaknesses. This hybrid approach is applicable to many practical problems including computer-assisted decision support systems for biodiversity and conservation decision support systems.

### 2.2. Artificial Neural Networks

Artificial neural networks are artificial intelligence paradigms; they are machine learning tools which are loosely modeled after biological neural systems. They learn by training from past experience data and make generalization on unseen data. They have been applied as tools for modeling and problem solving in real world applications such as speech recognition, gesture recognition, financial prediction, ecological modeling and medical diagnostics (Robinson, 1994; Giles et al., 1997; Marakami and Taguchi, 1991; Hilbert and Ostendorf, 2001; Omlin and Snyders, 2003). Neural networks learn by training on past experience using an algorithm which modifies the interconnection weights as directed by a learning objective for a particular application. A neuron is a single processing unit which computes the weighted sum of its inputs. The output of the network relies on cooperation of the individual neurons. The learnt knowledge is distributed over trained networks weights. Neural networks are characterized into feedforward and

recurrent neural networks. Unlike feedforward neural networks, recurrent neural networks contain feedback connections and have shown to learn and represent dynamical systems (Giles et al., 1999). Neural networks are capable of performing tasks that include pattern classification, function approximation, prediction or forecasting, clustering or categorization, time series prediction, optimization, and control. A detailed review on the applications and performance of artificial neural networks is done in (Paliwal and Kumar, 2007). Feedforward networks contain an input layer, one or many hidden layers and an output layer. Information is passed from the input layer to hidden layer or layers and then finally to output layer. During training, error information is passed backwards from the output layer. Figure 2 shows the architecture of a feedforward network.

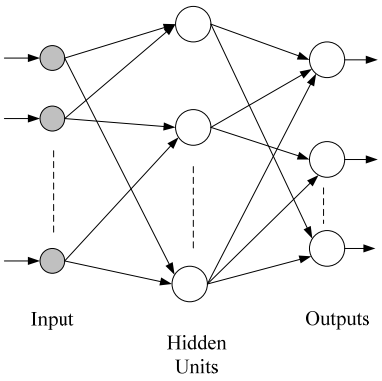


**Figure 2**. The architecture of the feedforward neural network with one hidden layer.

Backpropagation is the most widely applied learning algorithm for neural networks. It learns the weights for a multilayer network, given a network with a fixed set of weights and interconnections. Backpropagation employs gradient descent to minimize the squared error between the networks output values and desired values for those outputs. The goal of gradient descent learning is to minimize the sum of squared errors by propagating error signals backward through the network architecture upon the presentation of training samples from the training set. These error signals are used to calculate the weight updates which represent the knowledge learnt in the network. The performance of backpropagation can be improved by adding a momentum term and training multiple networks with the same data but different small random initializations prior to training (Ooyen and Nienhuis, 1992). In gradient descent search for a solution, the network searches through a weight space of errors. A limitation of gradient is that it may get trapped in a local minimum easily. This may prove costly in terms for network training and generalization performance. The success and failures of backpropagation is discussed in (Frasconi et al., 1993).

### 2.3. Integration of Symbolic Knowledge with Neural Networks

Combining symbolic and neural learning has become a well-established paradigm (Shavlik, 1994; Towell and Shavlik, 1994; Gallant, 1988; Omlin and Snyders, 2003). There are different ways in which neural and symbolic learning can be combined to solve a given learning task. Following the discussion in the introduction, the question arises whether neural networks can make effective use of explicit inductive bias and how such a bias influences the training and generalization performance. In order to introduce an explicit inductive bias in feedforward neural networks, i.e. a preference for a solution in hypothesis space, one has to investigate how neural networks represent knowledge and infer hypotheses from learning examples. Weighted connections between neurons provide an opportunity to incorporate knowledge prior to learning. The structure of the network and the programmed weights provide an explicit inductive bias.

The traditional approach to using neural networks is shown in the upper part of Figure 3. ("connectionist representation"). A network's adaptable weights are initialized with random values drawn according to some distribution. Using numerical optimization, the network is trained on some known data to perform a certain task (e.g. pattern classification) until some training criterion is met. After successful training, a network can take advantage of its generalization capabilities to perform the intended task on arbitrary data. Notice that during the entire process, the knowledge remains "hidden" in a network's adaptable connections, hence the name "connectionist representation".

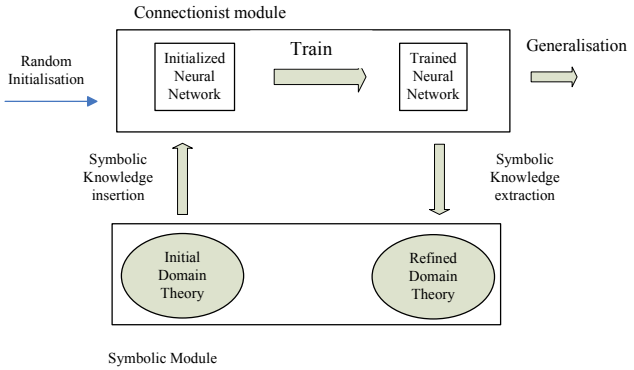


**Figure 3**. Knowledge-based neurocomputing paradigm.

The above training paradigm can be enriched with symbolic knowledge in the following way ("symbolic representation"): prior knowledge about a task (initial domain knowledge) is used to initialize a network before training. This requires a translation of the information from a symbolic into a connectionist representation. The particular method for converting the symbolic representation of knowledge into its equivalent connectionist representation depends on the kind of symbolic knowledge, the learning task, and the network model used for learning. To date, most efforts are directed towards encoding prior knowledge by programming some network weights to specified values instead of choosing small random values. The programmed weights define a starting point for the search of a solution in weight space. The premise is that a better solution will be found faster compared to starting the search from a ran-

dom point in weight space. The prior knowledge presumably defines a good starting point in the space of adaptable parameters and leads to faster learning convergence; it also provides an explicit inductive bias which focuses a network's attention on relevant input features or favors a desirable connectionist knowledge representation. Examples of this approach include pre-structuring of feedforward networks with Boolean concepts (Abu, 1990). The choice of a network architecture itself represents an implicit use of prior knowledge about an application.

Fidelity of the mapping of the prior knowledge into a network is very important since a network may not be able to take full advantage of poorly encoded prior knowledge or, if the encoding alters the essence of the prior knowledge, the prior knowledge may actually hinder the learning process. Once a network has succeeded in learning a task as measured by its performance on the training data, it may be useful to extract the learned knowledge. The question arises whether it is possible to extract an adequate symbolic representation of the knowledge learned by a network, i.e. a representation that captures the essence of the learned knowledge. In many cases, the extracted knowledge may only approximate a network's true knowledge; however, it is also possible for the extracted symbolic representation to exceed the accuracy of the knowledge stored in a trained network (Shavlik, 1994). This paper is concerned with the management of uncertainty in the expert knowledge. Validation and verification of extracted refined knowledge, although important, is beyond the scope of this discussion.

There are advantages to making effective use of prior knowledge that is common to all learning tasks: (1) It may lead to faster convergence to a solution; (2) networks trained with hints may generalize better to future examples; and (3) explicit rules may be used to generate additional training data which are not present in the original data set. The initialization of feedforward networks with Horn clauses has been the predominant paradigm for prior knowledge in the neural networks community.

### 2.4. Prior Knowledge Encoding

Prior knowledge can be used to derive an initial hypothesis from which to start the search for a solution. In knowledge-based artificial neural networks (KBANNs), an initial domain theory in the form of propositional rules is used to construct a feedforward neural network (Shavlik, 1994). The backpropagation learning algorithm is then used to refine that initial domain theory. KBANN provides an inductive bias which is more likely to generalize as predicted by the initial domain theory; backpropagation provides a generalization bias such that networks are more likely to converge toward a solution with small weights.

The method proposed in (Shavlik, 1994) is used to illustrate how Horn clauses can be encoded into feedforward networks. The construction of an initial network is based on the correspondence between entities of the knowledge base and neural networks, respectively. Supporting facts translate into input neurons, intermediate conclusions are modeled as hidden neurons, output neurons represent final conclusions; dependencies are expressed as weighted connections between neurons. The neuron outputs are computed by a sigmoidal function which takes as its argument a weighted sum of inputs.

Given a set of if-then rules, disjunctive rules are rewritten as follows: The consequent of each rule becomes the consequent of a single antecedent; it in turns becomes the consequent of the original rule. This rewriting step is necessary in order to prevent combinations of antecedents from activating a neuron when the corresponding conclusion cannot be drawn from such combinations. These rules are then mapped into a network topology as follows: a neuron is connected via weight H to a neuron in a higher level if that neuron corresponds to an antecedent of the corresponding conclusion. The weight of that connection is +H if the antecedent is positive; otherwise, the weight is programmed to −H. For conjunctive rules, the neuron bias of the corresponding consequent is set to $-(P - 1/2)H$ where P is the number of positive antecedents; for disjunctive rules, the neuron bias is set to $-H/2$. This guarantees that neurons have a high output when all or any one of their antecedents have a high output for conjunctive and disjunctive rules, respectively. If the given initial domain theory is incomplete or incorrect, a network may be supplemented with additional neurons and weights which correspond to rules still to be learned from data. A detailed study of knowledge insertion in neural networks is done in (Shavlik, 1994; Towell and Shavlik, 1994).

If an initial domain theory is sparse, the network constructed from the prior knowledge may be too small for a given learning task. In particular, the number of hidden neurons which along with their weights correspond to intermediate conclusions may be insufficient. Additional neurons may thus be added prior to training. It has generally been observed that networks initialized with correct prior knowledge train faster and generalize better compared to networks trained without the benefits of an initial domain theory (Snyders and Omlin, 2000; Omlin and Snyders, 2003).

While good empirical results have been achieved using the framework which combines neural and symbolic learning described above, the merits underlying the symbolic/connectionist approach are not yet well understood. Gaining that insight remains an important open research problem. In this paper, we will address the following open question: how should this explicit inductive bias *H* be chosen? If we give too little weight to the inductive bias, then it may not be very helpful in finding a solution. If we assign too much importance to it, then the network might not be able to find a solution, particularly when the prior knowledge and the training data do not represent similar concepts.

It is conceivable that the choice of this inductive bias depends on the application, the training data, and the network architecture. Therefore, a novel heuristic is proposed for determining the strength of the inductive bias for feedforward neural networks encoded with prior information using the KBANN method and demonstrated its usefulness for applications with excellent and poor prior knowledge, respectively (Snyders and

Omlin, 2000; Omlin and Snyders, 2003). In the next section, the details of this heuristic is presented.

## 3. Strength of Inductive Bias for KBANN

### 3.1. Motivation

Based on empirical investigations, the authors of KBANN suggest that all weights which reflect prior knowledge about a learning task be set to $H = 4$ (Towell and Shavlik, 1994). This indiscriminant choice of the inductive bias has two major drawbacks: (1) it is conceivable that different applications require different choices of the inductive bias $H$ which leads to fast convergence and good generalization performance; and (2) it does not provide a mechanism for dealing with uncertainty about the initial domain theory. This section discusses a method for choosing the strength of the inductive bias which takes these two objections into account: The choice of $H$ depends on the application represented by the initial domain theory, the network architecture, and the training data; it adjusts its confidence into the prior knowledge according to the amount and the quality of the available prior knowledge.

Consider an error function $E$ used to train a network. The idea for determining a good value for the inductive bias $H$ is to start the search for solution in a point in weight space where the gradient $\partial E / \partial H$ is maximal, i.e. we choose $H$ such that the search starts in a point where the error function in the direction of the inductive bias $H$ is steep; this avoids the need for determining $H$ through trial-and-error or traversing flat regions of the weight space during the initial training phase. Furthermore, the value $H$ which gives good training performance depends on the prior knowledge and the training data. The function $\partial E / \partial H$ takes both these dependencies into consideration. The more prior knowledge is available and the more accurate that knowledge is, the more the function $\partial E / \partial H$ influences the gradient-descent search for a solution in weight space. Steep descent makes fast convergence possible; furthermore, it is a reasonable premise that good local minima in weight space are more likely to be found at the bottom of steep ravines than in shallow areas.

### 3.2. The Neural Network Dynamics and Training Equations

The dynamics of a typical knowledge based feed forward network can be described by the following equations:

$$S_j^l = g_j\left(\sum_{i=1}^{m} S_i^{l-1} w_{ji}^l - b_j^l\right) \tag{1}$$

where $S_j^l$ is the output of the neuron $j$ in layer $l$. $g_j$ is the discriminant function, typically a sigmoidal function. $S_i^{l-1}$ is the output of the neuron $i$ in layer $l-1$ (containing $m$ neurons) and $w_{ji}^l$ the weight associated with that connection with $j$. $b_j^l$ is the internal threshold/bias of the neuron.

Weight updates for a specific pattern are done using the quadratic error function:

$$E = \frac{1}{2}\sum_{j=1}^{m}(d_j - S_j^L)^2 \tag{2}$$

where $d_j$ is the desired output for neuron $j$ in the output layer (containing $m$ neurons), and $S_j^L$ is the actual output of neuron $j$ in layer $L$, where $L$ is the output layer.

The weight updates are computed by:

$$\Delta w_{ji}^l = \alpha \delta_j^l S_i^{l-1} \tag{3}$$

where $\alpha$ is the learning rate constant. The local gradient for neuron $j$, $\delta_j^l$ can be calculated by:

$$\delta_j^l = \begin{cases} (d_j - S_j^l) S_j^l (1 - S_j^l) & \text{when layer } l \text{ is an output layer} \\ S_j^l (1 - S_j^l) \sum_{k=1}^{m} \delta_k^{l+1} w_{kj}^{l+1} & \text{when layer } l \text{ is a hidden layer} \end{cases} \tag{4}$$

### 3.3. Derivation for the Inductive Bias

In this section, a recursive procedure is derived for evaluating the gradient $\partial E(H) / \partial H$ prior to training which is similar to the error backpropagation learning algorithm. The value of the error function $E$ depends on the particular choice of $H$, thus $\partial E(H)$. For simplicity, the argument $H$ is omitted in the equations for the computation of $\partial E(H) / \partial H$ .

Consider a commonly used quadratic error function for a specific pattern:

$$E(H) = [d_0 - S_0^L(H)]^2 / 2 \tag{5}$$

where $d_0$ is the desired network output and $S_0^L(H)$ is the actual network output for a specific pattern $p$, where $L$ is the output layer. Notice that $S_0^L$ depends on the particular choice of $H$. Then, the derivative $\partial E / \partial H$ is given by:

$$\frac{\partial E}{\partial H} = -(d_0 - S_0^L)\frac{\partial S_0^L}{\partial H} \tag{6}$$

$\partial S_0^L / \partial H$ is computed recursively as follows:

$$\frac{\partial S_0^L}{\partial H} = S_0^L (1 - S_0^L) \sum_{j=1}^{m} \left( \frac{\partial w_{0j}^L}{\partial H} S_j^{L-1} + w_{0j}^L \frac{\partial S_j^{L-1}}{\partial H} \right) \tag{7}$$

where $w_{0j}^L$ is the weight connecting output of neuron $j$ in the hidden layer (containing $m$ neurons) immediately proceeding the network output layer with the output neuron $S_0^L$. The derivative $\partial w_{0j}^L / \partial H$ can be easily calculated by:

$$\frac{\partial w_{0j}^L}{\partial H} = \begin{cases} +1 & if\ w_{0j}^L = +H \\ -1 & if\ w_{0j}^L = -H \\ 0 & otherwise \end{cases} \tag{8}$$

The derivative $\partial S_j^l / \partial H$ for neurons in hidden layers $l$ can be recursively computed similarly:

$$\frac{\partial S_j^l}{\partial H} = S_j^l (1 - S_j^l) \sum_{i=1}^{m} \left( \frac{\partial w_{ji}^l}{\partial H} S_i^{l-1} + w_{ji}^l \frac{\partial S_i^{l-1}}{\partial H} \right) \quad (9)$$

where $w_{ji}^l$ connects neuron $i$, in layer $l - 1$, with neuron $j$ in the next hidden layer, $l$. The derivative $\partial w_{ji}^l / \partial H$ can easily be calculated by:

$$\frac{\partial w_{ji}^l}{\partial H} = \begin{cases} +1 \; if \; w_{ji}^l = +H \\ -1 \; if \; w_{ji}^l = -H \\ 0 \quad otherwise \end{cases} \quad (10)$$

There is a need for a "bootstrap" equation in the case where node $j$ is in the first hidden layer ($l = 0$), i.e. $S_i^{l-1}$ does not depend on $H$ since it is equal to the value of input neuron $i$. Therefore, $\partial S_i^{l-1} / \partial H = 0$ and equation $v$ simplifies to:

$$\frac{\partial S_j^l}{\partial H} = S_j^l (1 - S_j^l) \sum_{i=1}^{m} \frac{\partial w_{ji}^l}{\partial H} S_i^{l-1} \quad (11)$$

The same equations also apply to the neuron biases. This method has been successfully applied to a problem in molecular biology (Snyders and Omlin, 2000) and medical diagnosis (Omlin and Snyders, 2003). In the next section, its application to biodiversity and conservation is presented.

## 4. Prediction of Renosterveld

### 4.1. Data and Initial Domain Theory

The Renosterveld dataset was collected by Gershon Naidoo at the University of Western Cape. The dataset consists of 600 samples of field data with attributes of elevation (in meters), geographical aspect (in degrees), two soil types (quatzitic and shale/grain), slope (in degrees) and annual rainfall (in mm); the class attribute indicates the presence/absence of the vegetation. Table 1 shows the statistical distribution of the dataset. Please note that the geographic coordinates are not included as the objective is to develop a decision support system for Renosterveld conservation which is independent of the specific geographic location. The dataset was equally divided into positive and negative samples, i.e. 300 each. 80% of the available data was randomly chosen for training and the remaining 20% was used for testing.

As a preprocessing step, the attribute elevation was divided into 16 categories from 0 to above 750 meters in steps of 50 meters, the attribute aspect into 16 classes in steps of 22.5 degrees, the attribute slope into 16 categories in steps of 1 degree, and the attribute rainfall into 13 categories ranging from less than 300 mm to more than 850 mm of annual rainfall in steps of 50 mm. The attributes for soil type are binary attributes. The division of the attributes into different number of categories was done in order to explicitly represent and encode the attribute as binary values. This is further illustrated in Figure 4 where four neurons were used to represent the attribute *Elevation* which was divided in 16 intervals. Four binary neurons are sufficient to represent 16 categories. Furthermore, an extra neuron was used to represent that the elevation is between 200 to 400 meters as appeared in the expert knowledge. The data was preprocessed in such a way that the value of this "extra neuron" which represents elevation of 200 to 400 meters is high according to the corresponding data samples in the attribute elevation. The number of categories for each attribute was determined by examining the distribution of data. The range of the attribute and the rules to be encoded directly affected the choice for the number of intervals for a particular attribute.

**Table 1.** The Statistical Distribution of the Renosterveld Vegetation Dataset

| Attribute | Mean | Range |
|---|---|---|
| Elevation | 120.4 | 0 – 1296 meters |
| Aspect | 129.7 | 0 – 360 degrees |
| Quatzitic soil | - | [0, 1] |
| Shale/grain | - | [0, 1] |
| Slope | 2.7 | 0 – 36 degrees |
| Rainfall | 508.2 | 101 – 1005 mm |

**Table 2.** Neural Network Training without Prior Knowledge

| Exp. | Time (epochs) | Training (%) | Testing (%) |
|---|---|---|---|
| 1 | 611 | 97.8 | 77.4 |
| 2 | 326 | 97.2 | 71.3 |
| 3 | 1272 | 97.8 | 72.2 |
| 4 | 179 | 97.8 | 80 |
| 5 | 1500 | 96.8 | 76.5 |
| 6 | 468 | 97.2 | 73.9 |
| 7 | 189 | 97.2 | 73 |
| 8 | 207 | 97.8 | 72.2 |
| 9 | 150 | 97.8 | 77.4 |
| 10 | 277 | 97.4 | 79.1 |
| 11 | 329 | 97.8 | 73 |
| 12 | 274 | 97.4 | 73 |
| 13 | 175 | 97.4 | 78.3 |
| 14 | 221 | 97.4 | 73 |
| 15 | 246 | 97.8 | 76 |
| 16 | 437 | 97.4 | 67 |
| 17 | 345 | 97.6 | 73.9 |
| 18 | 210 | 97.2 | 74.8 |
| 19 | 222 | 97.2 | 75.7 |
| 20 | 1500 | 97.2 | 75.7 |
| 21 | 218 | 98 | 77.4 |
| 22 | 240 | 97.4 | 73 |
| 23 | 383 | 98.3 | 72.2 |
| 24 | 277 | 97.8 | 74.8 |
| 25 | 193 | 98.3 | 70 |

**Note**: The results of 25 training runs on randomly selected 80% of the training data, and the prediction performance of the trained network on the training and test data are shown, respectively.

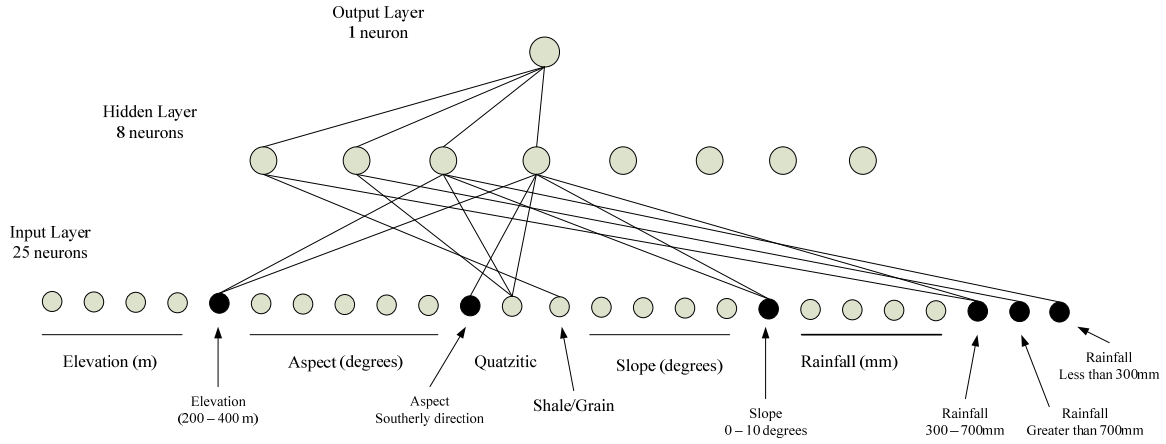


**Figure 4**. Knowledge insertion.

### 4.2. Training without Expert Knowledge

In this section, 25 training experiments done with randomly selected 80% of the available with a learning rate of 0.9. This value was selected by examining the performance of the network on learning rate values from 0.1 to 1.0. All neural networks were trained until one of the three following stopping criteria was satisfied: (1) on 100% of the training examples, the activation of every output unit was within 0.2 of the desired output; or (2) a network had been trained for 1,500 epochs; or (3) a network classified at least 97% of the training examples correctly, but had not improved it's ability to classify the training.

Neurons had sigmoidal discriminant functions and all networks were trained using the standard quadratic error function. The individual results for training time, prediction performance on the training and test sets, respectively, are shown in Table 2. The mean and 95 percent confidence intervals for training time (measured in number of epochs), training and generalization performance (both measured in percentage of correctly classified instances ) are 417 ± 151 epochs, 97.6% ± 0.1% and 74.4% ± 1.1%, respectively.

Note that 8 hidden neurons were used for all training. Trial experiments were done using 8, 10, 12 and 16 neurons in the hidden layer. The results showed that 8 neurons were sufficient for the best generalization performance; therefore, in all experiments, 8 neurons were used in order to compare different training paradigms. Furthermore, all the rules can be represented and encoded minimally using 8 neurons for knowledge based neural networks.

### 4.3. Training with Expert Knowledge

The expert knowledge about the presence of the Renosterveld vegetation is summarized in the following rules:

- If the sand is shale or granite and the rainfall is between 300 to 700 mm, then the probability of occurrence of Renosterveld is high.
- If the sand is Quartzitic and the rainfall is greater than 700 mm, then the likelihood of occurrence of Renosterveld is high.
- If the sand is of Quartzitic type and the rainfall is below 300 mm, and if the altitude is between 200 m to 400 m, and the slope is 0 to 10 degrees, then the chances of the presence of Renosterveld is high.
- If the sand is of Quartzitic type and the rainfall is between 300 mm to 700 mm, and if the altitude is between 200 m to 400 m, and the slope is 0 to 10 degrees and faces a southerly direction, then the chances of occurrences is high.

Please note that the above expert knowledge only makes statements about the presence of Renosterveld and its dependence on the attributes; they do not describe conditions for the absence of the vegetation. The expert knowledge only explains approximately 55% of the presence of Renosterveld. This information was gathered by checking the expert knowledge in relation to the field data at the preprocessing stage. Therefore, the initial domain theory which explains only 55 % of field data is far from complete.

The initial domain theory was mapped into a neural network as shown in Figure 4. The attribute values for elevation, rainfall, aspect and slope was mapped into non-overlapping intervals. Each interval was assigned a unique separate identifier in the form of a binary number. For instance, the attribute elevation was split into 16 intervals. Four binary neurons were used to represent the corresponding value in the dataset since 4 digits in binary can well represent 16 intervals as explained earlier in the data preprocessing section. Similar preprocessing and knowledge insertion for the remaining attributes results in a pre-structured neural network is shown in Figure 4. The primary objective of this chosen input encoding is to keep the number of neurons and the number of weighted connections small. A one-to-one correspondence between attribute value intervals and input neurons is possible, but leads to a much larger network; see Figure 5 for the resulting network architecture. Since smaller networks generally have better generalization performance, the discussion is limited to the former.

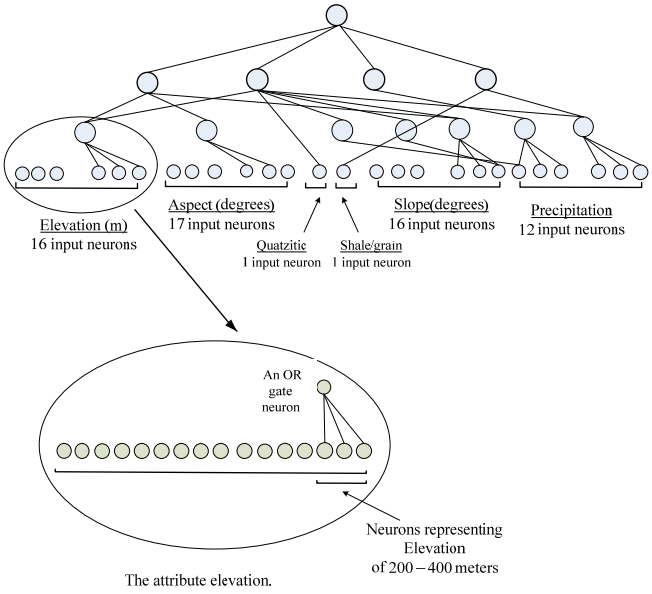


**Figure 5**. Alternate knowledge representation in the network.

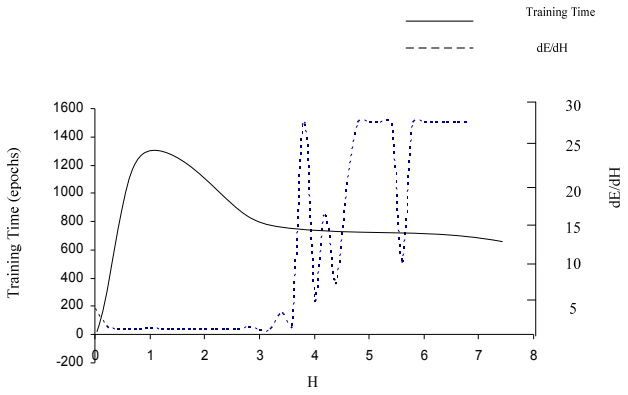


**Figure 6.** Training performance as a function of inductive bias H.

Figure 6 shows typical training times for fixed training set and prior knowledge as a function of *H*. It is observe that the proposed heuristic for choosing an explicit inductive bias yields good training time performance. From the graph of the function $\partial E / \partial H$ , it is observed that the function $\partial E / \partial H$ has a maximum near the inductive bias $H \approx 1.0$. This confirms that the initial domain theory does not fully explain the given training data. A weak inductive bias seems to indicate the programmed network's lower confidence in the prior knowledge. This is due to the small training data set and the incomplete overlapping in between the initial domain theory and the data which leads the proposed heuristic to choose a weak inductive bias. In applications where the initial domain theory and the training data represent similar concepts, it is observed that they have a synergistic effect on the training performance of neural networks (Snyders and Omlin, 2000). Note that the standard value of $H = 4$ results in significantly slower training.

The results from 25 experiments for training time, prediction performance on the training and test sets, respectively, with prior knowledge for standard inductive bias $H = 4$ and for our heuristic are shown in Table 3. The mean and 95 percent confidence intervals for training time (measured in number of epochs), training and generalization performance (both measured in percentage of correctly classified instances) are 885 ± 192 epochs, 97.3% ± 0.2% and 73.4% ± 1%, respectively, for the standard value of $H = 4$ are shown in Table 3. The corresponding performance measures for the proposed inductive bias *H* for the same application data indicates 268 ± 100 epochs, 97.5% ± 0.3 % and 74.1% ± 1.3%, respectively and is shown in Table 4.

**Table 3.** Neural Network Training with Prior Knowledge using Standard Inductive Bias H = 4

| Exp. | Time (epochs) | Std. Bias (H) | Training (%) | Testing (%) |
|---|---|---|---|---|
| 1 | 525 | 4 | 97.2 | 73 |
| 2 | 1094 | 4 | 97.2 | 71.3 |
| 3 | 1500 | 4 | 96.3 | 75.7 |
| 4 | 192 | 4 | 97.4 | 75.7 |
| 5 | 321 | 4 | 97.2 | 73.9 |
| 6 | 755 | 4 | 97.2 | 78.3 |
| 7 | 273 | 4 | 96.7 | 74.8 |
| 8 | 1395 | 4 | 97.2 | 73 |
| 9 | 853 | 4 | 98.3 | 75.7 |
| 10 | 306 | 4 | 97.4 | 73.9 |
| 11 | 328 | 4 | 98 | 75.7 |
| 12 | 343 | 4 | 98 | 67.8 |
| 13 | 1500 | 4 | 97 | 77.4 |
| 14 | 268 | 4 | 97.2 | 69.6 |
| 15 | 1121 | 4 | 97.6 | 71.3 |
| 16 | 619 | 4 | 97.6 | 73 |
| 17 | 968 | 4 | 97.2 | 77.4 |
| 18 | 1500 | 4 | 96.7 | 67.8 |
| 19 | 502 | 4 | 97.4 | 74.8 |
| 20 | 1500 | 4 | 97 | 73.9 |
| 21 | 603 | 4 | 97.2 | 71.3 |
| 22 | 1500 | 4 | 98.3 | 73.9 |
| 23 | 1500 | 4 | 96.7 | 72.2 |
| 24 | 1500 | 4 | 98 | 73 |
| 25 | 1173 | 4 | 97.2 | 71.3 |

**Note**: The results of 25 training runs on randomly selected 80% of the training data, and the prediction performance of the trained network on the training and test data are shown, respectively.

The results show that the training time for our adaptive inductive bias *H* (268 ± 100 epochs ) is significantly lower compared to the standard value of $H = 4$ (885 ± 192 epochs). Training with expert knowledge using standard value of $H = 4$ performs worse compared to training without expert knowledge (417 ± 151 epochs). This inferior training performance with carelessly chosen inductive bias *H* over the training performance without prior knowledge clearly emphasizes the need for careful selection of *H*. The results also show that there is not a significant difference in the training and generalization performance of the two comparable paradigms. This is so because of the noise in the dataset and the fact that the expert

knowledge only explains 55% of field data, which implies that the expert knowledge inserted does not affect the neural network training in a greater scale, but only provide a little hint in training. In general, it can be said that our proposed knowledge insertion technique, which uses an inductive bias *H*, has shown good performance in neural network training time.

## 5. Conclusions

Conservation of Renosterveld is a typical example of an environmental management application: the obtained results are complex and their interpretation requires expert knowledge. Knowledge-based neural networks have proven useful as they can synergistically combine this expert knowledge with inductive learning from data. The expert knowledge provides an explicit inductive bias for the network training: (1) it determines the network architecture; and (2) instead of initializing all network weights to small random values, it programs weights that correspond to prior knowledge to a value *H*. This process results in superior training and generalization performance.

**Table 4.** Neural Network Training with Prior Knowledge using Adaptive Inductive bias H.

| Exp. | Time (epochs) | Inductive Bias (H) | Training (%) | Testing (%) |
|---|---|---|---|---|
| 1 | 1500 | 1 | 97.4 | 76.5 |
| 2 | 127 | 0.9 | 97.8 | 64.3 |
| 3 | 289 | 1.2 | 97.6 | 79.1 |
| 4 | 262 | 0.9 | 97.6 | 73 |
| 5 | 222 | 1 | 97.8 | 78.3 |
| 6 | 225 | 1 | 97 | 74.8 |
| 7 | 203 | 0.9 | 98.7 | 72.2 |
| 8 | 169 | 1 | 97.4 | 68.7 |
| 9 | 171 | 0.9 | 97.8 | 75.7 |
| 10 | 301 | 0.9 | 97.2 | 76.5 |
| 11 | 186 | 0.9 | 98.5 | 67.8 |
| 12 | 218 | 0.9 | 95 | 74.8 |
| 13 | 215 | 0.9 | 97.2 | 75.7 |
| 14 | 235 | 1 | 97.6 | 75.7 |
| 15 | 198 | 1.1 | 97.2 | 74.8 |
| 16 | 266 | 1.1 | 97 | 71.3 |
| 17 | 198 | 1 | 97.6 | 75.7 |
| 18 | 326 | 1 | 97.4 | 77.4 |
| 19 | 167 | 1 | 97.2 | 73 |
| 20 | 159 | 1 | 97.6 | 76.5 |
| 21 | 239 | 1.1 | 97.8 | 73 |
| 22 | 153 | 1 | 97.2 | 73.9 |
| 23 | 178 | 1.1 | 98 | 75.7 |
| 24 | 237 | 1 | 97.6 | 73.9 |
| 25 | 256 | 1 | 97.2 | 73.9 |

* The results of 25 training runs on randomly selected 80% of the training data, and the prediction performance of the trained network on the training and test data are shown, respectively.

In this paper, the open question is addressed: What strength of the inductive bias *H* yields good training and generalization performance? It has been shown that, for this complex real-world domain, the proposed heuristic for choosing the inductive bias *H* outperforms the suggested standard inductive bias H = 4. The proposed heuristic chooses the inductive bias *H* such that the derivative $\partial E / \partial H$ of the error function *E* is maximal. The premises of this heuristic are (1) to start the search for a local minimum in weight space where gradient-descent can rapidly converge to a local minimum, and (2) that good local minima are more likely to be found at the base of deep ravines rather than in shallow areas. The experiments have shown that this heuristic is not sensitive to the values of initial weights that are not programmed; it takes the initial domain theory, the training data, the network structure and learning algorithm into consideration when choosing a value of *H*. Thus, the proposed heuristic is able to determine its confidence in the prior information and how well it explains the training data. In this application, the data and the initial domain theory did not represent sufficiently similar concepts; thus, the heuristic chose a low value for the inductive bias *H*.

A statistically significant improvement in the training performance has been observed. Therefore, it can be concluded that the proposed heuristic gives a quantitative measure for successfully dealing with uncertainty in the initial domain theory. Although the results are statistically significant, experiments with more data are necessary to fully utilize this knowledge for biodiversity and conservation biology. From a neural network learning perspective, it would be interesting to investigate whether the prior knowledge can not only be used to prestructure a neural network prior to training, but whether it can also be used during training to find a solution in weight space with superior performance and how extracted knowledge compares to the initial expert knowledge.

The proposed system can be used as a general paradigm to understand, gather and validate expert knowledge in different areas of ecological modeling where field data and expert knowledge are available. Another application of the proposed technique is in the development of intelligent decision support systems for environmental management. In the case where there is a need to gather expert knowledge from field data, knowledge extraction from trained neural networks is feasible (Towell and Shavlik, 1993). Another approach would to use decision trees to extract knowledge directly from field data.

**Acknowledgments.** We would like to thank Gershon Naidoo, Department of Computer Science, University of the Western Cape (South Africa) for collecting and making available field data on the presence and absence of Renosterveld.

## References

Abu-Mostafa, Y.S. (1990). Learning from Hints in Neural Networks, *Journal of Complexity*, 6,192-198, doi:10.1016/0885-064X(90)900 06-Y.

Chandra, R., and Omlin, C.W. (2005). A knowledge-Based Neurocomputing Decision Support System for Biodiversity and Conversation Biology, *Proc. of the International Conference on Environmental Management (ICEM) (2005). Environmental Geoinformatics and Modelling*, Vol IV, Hyderabad, India.

Cortes, U., Sanchez-Marre, M., Ceccaroni, L., Roda, I.R., and Poch, M. (2000). Artificial intelligence and enviornmental decision sup-

port systems, *Applied Intelligence*, 13, 77-91.

Davis, J.R., Hoare, J.H.L., and Nanninga, P.M. (1986). Developing a Fire Behaviour Expert System for Kakadu National Park, Australia, *J. Environ. Manage.*, 22, 215-22.

Hilbert, D.W., and Ostendorf, B. (2001). The utility of artificial neural networks for modelling the distribution of vegetation in past, present and future climates, *Ecol. Model.*, 146, 311-327, doi:10.1016/S0304-3800(01)00323-4.

Frasconi, P., Gori, M., and Tesi, A. (1993). Success and failures of backpropagation: a theoretical investigation, *Progress in Neural Networks*, Ablex Publishing.

Gallant, S.I. (1988). Connectionist expert systems, *Communications of the ACM*, 31, 152-169.

Giles, C.L., Omlin, C.W., and Thornber, K.K. (1999). Equivalence in Knowledge Representation: Automata, Recurrent Neural Networks, and Dynamical Fuzzy Systems, *Proceedings of the IEEE*, 87(9), 1623-1640.

Giles, C.L., Lawrence, S., and Tsoi, A.C. (1997). Rule inference for financial prediction using recurrent neural networks, *Proceedings of the IEEE/IAFE Computational Intelligence for Financial Engineering*, New York City, USA, pp. 253-259.

Loehle, C. (1987). Applying Artificial Intelligence Techniques to Ecological Modelling, *Ecol. Model.*, 38,191-212.

Marakami, K., and Taguchi, H. (1991). Gesture recognition using recurrent neural networks, *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through Technology*, Louisiana, USA, pp. 237-242.

Noble, I.R. (1987). The Role of Expert Systems in Vegetation Science, *Vegetation*, 69, 115-121.

Omlin, C.W., and Snyders, S. (2003). Inductive Bias in Knowledge-Based Neural Networks: Application to Magnetic Resonance Spectroscopy of Breast Tissues, *Artifical Intelligence in Medicine*, 28 (2), 121-140.

Ooyen, A., and Nienhuis, B. (1992). Improving the convergence of the back propagation algorithm, *Neural Networks*, 5, 465-471.

Paliwal, M., and Kumar, A.U. (2007). Neural networks and statistical techniques: A review of applications, *Expert Systems with Applications*, In Press.

Rizzoli, A.E., and Young, W.Y. (1997). Delivering Environmental Decision Support Systems, Software Tools and Techniques, *Environmental Modelling and Software*, 12 (23), 237-249, doi:10.1016/S1364-8152(97)00016-9.

Robinson, A.J. (1994). An application of recurrent nets to phone probability estimation, *IEEE Transactions on Neural Networks*, 5(2), 298-305.

Starfield, A.M. (1991). Qualitative, Rule Based Modelling, *BioScience*, 40, 601-604.

Snyders, S., Omlin, C.W. (2000). What Inductive Bias Gives Good Neural Network Training Performance? *Proceedings International Joint Conference on Neural Networks*, 3, 445-450.

Shavlik, J. (1994). Combining Symbolic and Neural Learning, *Machine Learning*, 14, 321-331.

Towell, G., and Shavlik, J.W. (1994). Knowledge-based Artificial Neural Networks, *Artificial Intelligence*, 70, 119-165.

Towell, G., and Shavlik, J.W. (1993). Extracting refined rules from knowledge-based neural networks, *Machine Learning*, 3(1), 71-101.