

Evolving Connectionist Systems for
Adaptive Decision Support with Application in
Ecological Data Modelling

Snjezana Soltic

A thesis submitted to
Auckland University of Technology
in fulfilment of the requirements for the degree of
Doctor of Philosophy (PhD)

2009

Knowledge Engineering & Discovery Research Institute
Faculty of Design and Creative Technologies

Primary Supervisor
Professor Nikola Kasabov, FRSNZ

Table of Contents

Table of Contents	i
List of figures	vi
List of tables	xvi
Abbreviations and acronyms	xviii
Acknowledgments	xxi
Abstract	xxiii
Chapter 1 Introduction	1
1.1 Motivation	1
1.2 Objectives	3
1.3 Research questions	4
1.4 Contributions	5
1.5 Research methodology	7
1.6 Published work	9
1.7 Thesis structure	10
Chapter 2 Connectionist-based modelling techniques for adaptive decision support systems: a review	13
2.1 Introduction	13
2.1.1 Knowledge based DSS - methods and tools	15
2.2 Artificial Neural Networks	16
2.2.1 Artificial Neural Networks for DSS	16
2.2.2 Artificial neurons	18
2.2.3 Connectionist Systems	21
2.2.4 Evolving Connectionist Systems	22
2.3 Inductive and transductive reasoning	23
2.4 Global, local and ‘personalised’ modelling	24
2.5 Feature selection and model validation	27
2.5.1 Feature selection	27
2.5.2 Model validation	30
2.6 Conclusion	31
Chapter 3 Ecological modelling, benchmark work and bio-climatic distribution data	32
3.1 Ecological modelling: benchmark works	32
3.2 Artificial neural networks in the field of ecological modelling	34
3.2.1 Artificial neural networks in ecological modelling versus other approaches	35
3.2.2 Multi-layer perceptron networks for knowledge discovery in ecology	36

3.3 Invasion of exotic pest insects – a case study problem	36
3.3.1 Climatic-habitat modelling	36
3.3.2 Field collection of eco-climatic data.....	38
3.3.2.1. Noise in insect distribution data.....	39
3.3.3 Validation of the predictive ecological models	39
3.4 Real-world bio-climatic pest distribution data and its preliminary analysis	42
3.5 Conclusion.....	46
Chapter 4 Local probability based adaptive model (LPAM) for risk evaluation and knowledge discovery.....	48
4.1 Introduction	48
4.2 The proposed model	50
4.2.1 Problem definition	50
4.2.2 ECM Clustering	50
4.2.3 Local probability evaluation	52
4.2.4 Knowledge discovery in DENFIS	53
4.2.5 Model implementation.....	54
4.3 Local knowledge extraction with the use of LPAM and its visualization.....	54
4.4 Validation of the LPAM using benchmark datasets.....	56
4.4.1 Validation on the iris dataset	57
4.4.2 Validation on the wine dataset.....	59
4.4.3 Comment on the knowledge discovery.....	63
4.5 Conclusion.....	64
Chapter 5 Modelling and prediction of establishment of the insect <i>P. citri</i> using LPAM	66
5.1 Introduction	66
5.2 Data characteristics.....	67
5.3 Clustering information	70
5.4 Establishment and climatic variables	72
5.5 Risk map.....	76
5.6 Influence of the threshold parameter	77
5.7 Informative comparison	78
5.8 Conclusion.....	79
Chapter 6 Transductive approach for ‘personalised’ modelling and knowledge discovery with case studies from biosecurity	82
6.1 Introduction	82
6.2 Data characteristics.....	83
6.3 Discussion of results.....	87

6.3.1 Model evaluation	87
6.3.2 Evaluation of the <i>A. hartii</i> distribution models	88
6.3.3 Evaluation of the <i>G. coffeae</i> distribution models	92
6.3.4 Evaluation of the <i>X. perforans</i> distribution models.....	94
6.3.5 Comparative analysis of the four approaches when used for predicting the establishment potential of three different pest insects	96
6.3.6 Knowledge discovery for different modelling techniques.....	98
6.3.7 Comment on the knowledge discovery using the ‘personalised’ modelling	101
6.4 Conclusion.....	102
Chapter 7 Spiking neural networks for evolving connectionist systems – a review.....	104
7.1 Why to use spiking neurons and spiking neural network models?	105
7.2 Spiking neural models: a review	107
7.2.1 A brief introduction to natural neurons and neural networks	107
7.2.2 Hodgkin-Huxley model	109
7.2.3 Spike response model	111
7.2.4 Leaky integrate-and-fire model	112
7.2.5 Thorpe’s model.....	113
7.2.6 Izhikevich’s model.....	114
7.3 Information encoding	115
7.3.1 Rank order coding.....	117
7.3.2 Population coding	117
7.4 Evolving spiking neural network architecture and training	121
7.4.1 Evolving spiking neural network architecture.....	121
7.4.2 Evolving spiking neural network training	123
7.5 Spiking neural networks for Ecological Decision Support: an unexplored potential	124
7.6 Conclusion.....	125
Chapter 8 An evolving spiking neural network model for taste recognition (ESNN-PC-TR)	126
8.1 Taste perception	127
8.1.1 Biological models of taste-coding	127
8.1.2 Dynamics of taste-coding	129
8.2 Artificial taste recognition models	130
8.2.1 Benchmark works	130
8.2.2 Taste recognition models based on traditional neural networks and their shortcomings.....	132
8.3 ESNN model for taste recognition	132
8.4 Software simulation of the ESNN-PC.....	136

8.5 Data characteristics.....	139
8.6 Applications of the ESNN-PC-TR model for water and wine recognition	142
8.6.1 Water recognition	143
8.6.2 Wine recognition	144
8.6.3 Comparison with existing work.....	145
8.7 Knowledge discovery from ESNN-PC-TR on the case study problems.....	146
8.7.1 Knowledge in ESNN-PC	146
8.7.2 Knowledge extracted by ESNN-PC-TR	150
8.7.3 Comment on knowledge extracted by ESNN-PC.....	152
8.8 Conclusion.....	153
Chapter 9 FPGA implementation of the taste recognition model.....	156
9.1 Literature review	157
9.1.1 The main challenges for FPGA implementations of artificial neural networks	158
9.1.2 Learning in FPGA.....	160
9.1.3 FPGA implementations of spiking neural networks.....	161
9.2 FPGA implementation of the ESNN-PC-TR model	162
9.2.1 Embedded processor system.....	164
9.2.2 A novel FPGA implementation of population encoding	164
9.2.3 A novel FPGA implementation of the rank order coder.....	166
9.2.4 A scheme for an FPGA implementation of the spiking integrate-and-fire neuron and its synapse	167
9.2.5 A method for an FPGA implementation of on-chip learning	168
9.2.6 A discussion on the size of ESNN-PC-TR when implemented on an FPGA	170
9.3 Implementation results	170
9.3.1 FPGA resource usage and operating speed	171
9.3.2 Control signals and timing.....	173
9.4 Conclusion.....	175
Chapter 10 Conclusion and future directions.....	177
10.1 Research summary	177
10.1.1 Answering thesis questions.....	180
10.2 Analysis of the proposed models.....	183
10.2.1 LPAM model strengths and weaknesses	183
10.2.2 ESNN-PC model strength and weaknesses	184
10.3 Suggestions for future work	185
10.3.1 The ESNN based multimodal taste recognition system	185
10.3.2 Smarter mobile robots.....	186

10.3.3 Pervasive computing.....	187
10.3.4 Quantum evolving spiking neural networks - QSNN.....	188
10.4 Overall conclusion.....	189
References	190
Appendix MATLAB code for LPAM.....	220

List of figures

1.1.	Overview of contributions of this thesis. The arrows going from the CI Methods to the Ecological Applications via New Models show the key contributions of this research in both computational models and their application for ecological modelling.	7
2.1.	A framework of a connectionist-based decision system proposed by Nikola Kasabov (Kasabov, 2003b).....	16
2.2.	A graphical representation of the approach taken in this thesis.	16
2.3.	An illustration of the knowledge based DSS framework. The circle nodes represent different neural networks working together in a DSS.....	17
2.4.	A conventional neuron receives n inputs and produces one output (y). g is an integration function and f is an activation function. The graphical representation of neurons is adopted from (Rojas, 1996).....	19
2.5.	The McCulloch-Pitts neuron and a step transfer function. The inputs x_i and output y are Boolean values. This unit can process only inputs equal to 1 and 0, <i>i.e.</i> $x_i \in \{0, 1\}$. Furthermore, a sample fed to one McCulloch-Pitts neuron causes either the 0 output or the 1 output.....	19
2.6.	The perceptron neuron (Rosenblatt, 1958). The weights w_i model the synaptic efficiency of the connections between two perceptrons. The inputs x_i and output y are real values. Different activation functions have been proposed over the years.	20
2.7.	Spiking neuron receiving three spikes and generating one spike. The spikes arrive via three synapses influencing the neuron's PSP. When the <i>PSP</i> exceeds a certain threshold the neuron itself generates a spike.	21
2.8.	ECOS interacts with the environment and other systems.....	22
2.9.	Examples of global (a), local (b) and 'personalised' (c) modelling on the same set of data. f_G is a global model, f_{L1} and f_{L2} are local models and f_{Pxi} is a 'personalised' model built for X_i	25
2.10.	An integrated global, local and 'personalised' multi-agent system. Local and 'personalised' agents capture local information and knowledge and the global agent captures global trends to provide computational refinements for complex data.....	26

2.11.	SNR ranking of 40 features of D_{Pest} describing the presence and absence of a pest insect. Features 30, 31 and 32 are the three highest ranked of all 40 features. Two big changes can be identified in the SNR values.....	28
2.12.	PCA analysis of the 40 features of D_{Pest} the describing presence and absence of a pest insect. The first principal component (p_1) accounts for more than 60% of the information present in the original set.	29
2.13.	Results of using GA for optimization of an ECF model trained on D_{Pest} . The selected features are shown as red and irrelevant features are shown as blue.....	30
3.1.	Confusion matrix. P_T is the number of present sites correctly predicted, A_T is the number of absent sites correctly predicted, P_F is the number of present sites incorrectly predicted as absent and A_F is the numbers of absent sites incorrectly predicted as present.....	40
3.2.	Given a threshold value θ , predictions greater than θ represent the species presence (P), otherwise the species is predicted to be absent (A). When θ changes the numbers of locations predicted as present and absent change as well. Smaller threshold values result in more locations predicted as present while increasing the threshold value results in fewer present locations.	41
3.3.	A typical conventional ROC curve (Metz, 1978). The area under the ROC curve is a measure of the model's predictive power.	42
3.4.	Spatial distribution of the insect <i>P. citri</i> 's presence. Please note that a number of island locations might appear as geographical errors.	45
3.5.	Distribution of the insect <i>P. citri</i> over a range of maximum summer temperature (T_{Smax} , °C). Each bar represents the distribution of all locations (n), where the dark areas represent the number of plots occupied by the insect (n_p). The number written on top of each bar shows the percentage of all sites occupied by the insect ($r = n_p \div n$ %).	46
4.1.	Block diagram of an LPAM. The clustering and inference stages are based on ECM and DENFIS, respectively. The local probability evaluation stage lowers the model's sensitivity to noise.	49
4.2.	ECM clustering (Kasabov & Song, 2002). The current sample X_i is represented by '*'. (a) $d_{im} < R_{cm}$, neither the cluster (C_{cm}, R_{cm}) is updated nor a new cluster is created, (b) $d_{im} > 2 \times D_{thr}$, a new cluster is created ($C_{ci} = X_i$ and $R_{ci} = 0$), (c) $d_{im} \leq 2 \times D_{thr}$, X_i belongs to the C_m cluster and C_{cm} and R_{cm} are updated..	51
4.3.	Contributions of predictor variables in two fuzzy rules. Each rule explains the influence of three variables, x_1 , x_2 and x_3 . Rule 1: higher $x_2 \Rightarrow$ lower response (a_{12}	


	is negative), the main contributor is x_3 (a_{13}). Rule 2: all positive contributors, the main contributor is x_1 (a_{21}).	56
4.4.	The iris dataset - average values and standard deviations of the features of 50 samples in each of the three different classes, each class representing one type of iris plant.	58
4.5.	The contributions of predictor variables in the two fuzzy rules for the iris Versicolour and iris Virginica dataset. In both rules P_l and P_w are more influential than S_l and S_w	60
4.6.	Two features for the iris Versicolour and iris Virginica plotted in a 2D space. (a) 2D sepal space, S_l vs. S_w (b) 2D petal space, P_l vs. P_w	60
4.7.	The contributions of predictor variables in the two fuzzy rules for the Class 0 and Class 1 wines.	63
5.1.	Distributions of the insect <i>P. citri</i> over a range of T_{Wmin} (a), R_{mean} (b), AAE (c) and MI (d). Each bar represents the distribution of all locations (n), where the dark areas represent the number of plots occupied by the insect (n_p). The number written on top of each bar shows the percentage of all sites occupied by the insect ($r = n_p \div n \%$). The distribution over the T_{Smax} is given in Chapter 3 Fig. 3.5.	69
5.2.	Contributions of predictor variables in three different fuzzy rules. Each rule explains the influence of five climate characteristics on the establishment of the pest insect. Rule 1: higher rainfall \Rightarrow lower response, the main positive contributors are $a_{1MI} = 1.91$ and $a_{1AAE} = 1.46$. Rule 2: higher MI , T_{Smax} and T_{Wmin} \Rightarrow lower response, the main positive contributor is $a_{2AAE} = 8.26$. Rule 3: all contributors have low mean values, higher R_{mean} \Rightarrow lower response, the main positive contributors are $a_{3AAE} = 11.62$ and $a_{3MI} = 8.21$	74
5.3.	The establishment probability map for the insect <i>P. citri</i> showing distributions predicted by the LPAM. Higher values represent the possible hot spots – the locations where the climate is suitable for the pest insect establishment. The map contours were created using the biharmonic spline interpolation method with 1° grid spacing available in MATLAB.	76
5.4.	The accuracy a of the predictive distribution model as a function of the threshold parameter θ . The accuracy is above 80% for $\theta = 0.8$ or 0.9	77
5.5.	Confusion matrix. R_p and R_A are the number of recorded presences and absences and P_p and P_A are the number of predicted presences and absences. 128 locations are correctly predicted as presences and 174 locations as absences.	78

6.1.	Distributions of the insect <i>A. hartii</i> over a range of TSmax (a), TWmin (b) and AAE (c). Each bar represents the distribution of all locations, where the dark areas represent the number of locations where the insect is present. The number written on top of each bar shows the percentage of all sites occupied by the insect ($r = n_p \div n \%$).	85
6.2.	Distributions of the insect <i>G. coffeae</i> over a range of T_{Smax} (a), T_{Wmin} (b) and AAE (c). Each bar represents the distribution of all locations, where the dark areas represent the number of locations where the insect is present. The number written on top of each bar shows the percentage of all sites occupied by the insect ($r = n_p \div n \%$).	86
6.3.	Distributions of the insect <i>X. perforans</i> over a range of TSmax (a), TWmin (b) and AAE (c). Each bar represents the distribution of all locations, where the dark areas represent the number of locations where the insect is present. The number written on top of each bar shows the percentage of all sites occupied by the insect ($r = n_p \div n \%$).	87
6.4.	The effect of θ on the performance of the global model built for <i>A. hartii</i> expressed in terms of sensitivity (<i>sen</i>), specificity (<i>spe</i>), Cohen's kappa (κ) and overall accuracy (<i>a</i>). The highest $\kappa_{max} = 0.80$ ('very good') and $a_{max} = 90\%$ are at $\theta_{Optimal} = 0.6$ & 0.5 . Note that a θ of 0.5 is typically used.	89
6.5.	The effect of θ on the performance of the local model built for <i>A. hartii</i> expressed in terms of sensitivity (<i>sen</i>), specificity (<i>spe</i>), kappa (κ) and overall accuracy (<i>a</i>). The highest $\kappa_{max} = 0.77$ ('very good') and $a_{max} = 88\%$ are at $\theta_{Optimal} = 0.7$. At $\theta = 0.6$ and $\theta = 0.7$ all measures but the κ values are above 0.7	89
6.6.	The effect of θ on the performance of the LPAM for <i>A. hartii</i> expressed in terms of sensitivity (<i>sen</i>), specificity (<i>spe</i>), kappa (κ) and overall accuracy (<i>a</i>). The highest $\kappa_{max} = 0.77$ ('very good') and $a_{max} = 88\%$ are at $\theta_{Optimal} = 0.3$ & 0.4 . At $\theta = 0.5$, $a = 85\%$	90
6.7.	The effect of θ on the performance of individual models built for <i>A. hartii</i> expressed in terms of sensitivity (<i>sen</i>), specificity (<i>spe</i>), kappa (κ) and overall accuracy (<i>a</i>). The highest $\kappa_{max} = 0.85$ ('excellent') and $a_{max} = 92\%$ are at $\theta_{Optimal} = 0.6$ & 0.7	91
6.8.	The κ values for the studied models. The performance of transductive model, when measured in terms of κ , is better than the performance of all other models.	91
6.9.	The effect of θ on the performance of the global model built for <i>G. coffeae</i> expressed in terms of sensitivity (<i>sen</i>), specificity (<i>spe</i>), Cohen's kappa (κ) and	

overall accuracy (a). The highest $\kappa_{max} = 0.76$ ('very good') and $a_{max} = 88\%$ are at $\theta_{Optimal} = 0.8$	92
6.10. The effect of θ on the performance of the local model built for <i>G. coffeae</i> expressed in terms of sensitivity (sen), specificity (spe), kappa (κ) and overall accuracy (a). The highest $\kappa_{max} = 0.81$ ('very good') and $a_{max} = 91\%$ are at $\theta_{Optimal} = 0.7$	93
6.11. The effect of θ on the performance of the LPAM built for <i>G. coffeae</i> expressed in terms of sensitivity (sen), specificity (spe), kappa (κ) and overall accuracy (a). The highest $\kappa_{max} = 0.71$ ('very good') and $a_{max} = 85\%$ are at $\theta_{Optimal} = 0.8$	93
6.12. The effect of θ on the performance of individual models built for <i>G. coffeae</i> expressed in terms of sensitivity (sen), specificity (spe), kappa (κ) and overall accuracy (a). The highest $\kappa_{max} = 0.52$ ('fair') is at thresholds $\theta = 0.4$ & 0.5 , $a_{max} = 76\%$ is at $\theta_{Optimal} = 0.5$	94
6.13. The κ values for the studied models. The performance of the ransductive model exhibits a different trend than performances of the other three models.	94
6.14. The effect of θ on the performance of the global model built for <i>X. perforans</i> expressed in terms of sensitivity (sen), specificity (spe), Cohen's kappa (κ) and overall accuracy (a). The highest $\kappa_{max} = 0.62$ ('good') and $a_{max} = 81\%$ are at $\theta_{Optimal} = 0.6$	95
6.15. The effect of θ on the performance of the local model built for <i>X. perforans</i> expressed in terms of sensitivity (sen), specificity (spe), kappa (κ) and overall accuracy (a). The highest $\kappa_{max} = 0.61$ ('good') and $a_{max} = 81\%$ are at $\theta_{Optimal} = 0.5$	95
6.16. The effect of θ on the performance of the LPAM built for <i>X. perforans</i> expressed in terms of sensitivity (sen), specificity (spe), kappa (κ) and overall accuracy (a). The highest $\kappa_{max} = 0.55$ ('good') and $a_{max} = 77\%$, are at $\theta_{Optimal} = 0.6$	96
6.17. The effect of θ on the performance of individual models built for <i>X. perforans</i> expressed in terms of sensitivity (sen), specificity (spe), kappa (κ) and overall accuracy (a). The highest $\kappa_{max} = 0.54$ ('fair') and $a_{max} = 77\%$, are at $\theta_{Optimal} = 0.5$. Note: 15% samples have been used to create the model.	96
6.18. The κ values for the studied models. None of the models achieved 'very good' κ values.	97
6.19. The maximum accuracies of the studied models in terms of a and κ values. Note that the maximum values were achieved at different $\theta_{Optimal}$	97

6.20.	The contributions of predictor variables in two different fuzzy rules. Each rule explains the influence of three climate characteristics on the establishment potential of <i>A. hartii</i> . LM ₁ : higher T_{Smax} , T_{Wmin} and $AAE \Rightarrow$ higher response, the main positive contributor is $a_{TWmin} = 1.57$. LM ₂ : higher $T_{Wmin} \Rightarrow$ higher response, the main positive contributor is $a_{TWmin} = 0.21$	100
6.21.	The influence of the number of neighbouring samples k on the accuracy of predicting establishment potential of <i>A. hartii</i> using transductive reasoning.....	100
6.22.	A hypothetical problem space D . The output value for X_i is calculated as a weighted average of the output values of the X_1, X_2, \dots, X_{12} data samples.	101
6.23.	A piece of ‘personalised’ knowledge – a ‘personalised’ model M_{ij} is created for each new data sample.	102
7.1.	On the left, an approximation of the structure of a tiny portion of the mammalian cortex is shown. See the text for the description of each functional part. In reality, the neurons are numerous and very closely packed. On the right, the schematic diagram of the soma as a processing unit with a number of inputs and one output is given.....	107
7.2.	This is a reproduction of the drawing by Gerstner and Kistler (Gerstner & Kistler, pg. 2, 2002) of a neural action potential (or spike) with an amplitude of about 100 mV. The spike’s duration is around 1-2 ms. In this thesis the spikes are approximated by a vertical bar, as shown on the right-hand side of this figure. .	108
7.3.	Typical forms of the post-synaptic potential of a neuron as a reaction to a received spike when the neuron is at rest. <i>EPSP</i> is caused by a spike arriving via an excitatory synapse and <i>IPSP</i> is caused by a spike arriving via an inhibitory synapse.....	109
7.4.	A postsynaptic spike is caused by four pre-synaptic spikes arriving via two excitatory synapses. When the <i>PSP</i> crosses PSP_{θ} , the post-synaptic spike is created and propagated along the axon. Note that the figure is an exaggeration of what occurs in reality, <i>i.e.</i> four spikes, each causing a voltage increment in the range of 1-2 mV are not enough to push the membrane potential <i>PSP</i> of the post-synaptic neuron above PSP_{θ} . In addition, the amplitudes of all voltages are not in proportion to the actual values.....	109
7.5.	The Hodgkin-Huxley model: schematic diagram (Gerstner, 1999). There are three channels: sodium (<i>Na</i>), potassium (<i>K</i>) and leakage channel.	110

7.6.	The leaky integrate-and-fire model of a neuron. The resistor R in parallel with a capacitor C models the soma of the neuron. The current $i(t)$ charges C . When the voltage across C exceeds PSP_{θ} the neuron produces a post-synaptic spike.	113
7.7.	The three average firing rates f_{r1} , f_{r2} and f_{r3} , where $f_{r1} < f_{r3} < f_{r2}$, have been converted to the temporal domain using the time-to-first-spike encoding scheme. The neuron associated with the highest average rate spikes first.	116
7.8.	Pseudo code for frequency encoding based on the probability in which the higher variable values result in more spikes being produced in the same period of time.	116
7.9.	The Three average firing rates f_{r1} , f_{r2} and f_{r3} , where $f_{r1} < f_{r3} < f_{r2}$, have been converted to the temporal domain using a frequency encoding scheme. The neuron associated with the highest average rate produces the highest number of spikes.	116
7.10.	Rank Order Coding. The highest value (p_2) is given the highest rank (0) and the lowest value (p_1) is given the lowest rank (2).	117
7.11.	This figure shows how two neurons are connected in the model proposed in (Natschläger & Ruf, 1998). Each connection consists of a set of connections associated with their own delays.	118
7.12.	An input variable v is encoded with six Gaussian receptive fields. The highest excited neuron (n_3) is given the shortest firing time, the second most excited neurons (n_4) is given some later firing time and so on. The lowest excited neurons (n_1 and n_6) are given increasing later firing times (t_{max}).	118
7.13.	Clusters of samples from three different classes in 2D (p_2 vs. p_1). Broadly and sharply tuned receptive fields might be required to further differentiate the data samples (Soltic, Wysoski & Kasabov, 2008).	120
7.14.	Population encoding increases the dimensionality of input data from n to $n \times m$. This might cause a problem in traditional neural networks but it is desirable when computing with spiking neural networks. Increased dimensionality allows more information to be conveyed and leads to higher system accuracy (Soltic, Wysoski & Kasabov, 2008).	120
7.15.	The structure of the evolving spiking neural network (ESNN) comprises L1 and L2 neurons. The L2 neurons are created during the learning stage. Each class C_i is represented with an ensemble of neurons (G_i), and each ensemble is trained to represent one class. For the sake of figure clarity, the weights between the input and the L1 neurons, which are equal to 1, as well as the weights w_{ji} between the	

	L1 and the L2 neurons are not shown. The w_{ji} weights are found during the training of the network.....	121
8.1.	An array of n non-selective taste sensors (represented by ) converts the chemical composition of tastants into the electrical domain. The sensors are referred to as an electronic tongue (Soltic, Wysoski & Kasabov, 2008).....	128
8.2.	Block diagram of an artificial gustatory system. The artificial gustatory system comprises a sensor array modelling the tongue, <i>i.e.</i> the electronic tongue, and a pattern recognition system modelling the taste recognition.	128
8.3.	Coding of an input sensor value v , and its corresponding orders O defined by the intersections of the vertical line representing v with all six Gaussian curves ($m = 6$). The value v is translated into orders $O = \{-, 3, 1, 0, 2, -\}$, where ‘-’ represents $p_i < \beta$	135
8.4.	The proposed taste recognition system comprises three layers: an electronic tongue that samples tastants, a GRF layer with m equally spaced Gaussian receptive fields which perform the population coding of sensor values, and an ESNN in which L1 represents the taste receptors whose values are encoded using ROC and L2 are the simple integrate-and-fire units sensitive to the order of the incoming spikes (Soltic, Wysoski & Kasabov, 2008).	137
8.5.	Pseudo-code for the clock-driven and event-driven algorithms. Note that the post-synaptic spikes in the event-driven algorithm occur at the arrival time of the pre-synaptic spikes and updating the post-synaptic potential of the neurons and checking the threshold conditions is done at the same time. If this was not the case, the algorithm would be more complex.	138
8.6.	A snapshot of the simulator’s GUI. The interface allows the ESNN-PC-TR parameters to be customised.	139
8.7.	Averages and standard deviations of the water samples. Note: the Class1 and Class2 averages are very similar, as are the Class3 and Class4 averages; the Class3 and Class4 averages are obviously of lesser magnitude than the Class1 and Class2 averages. The standard deviations in this data set are small.	140
8.8.	Water Class1 and Class3 data encoded using rank order coding (ROC) and rank order population coding (RO-POP C) with six receptive fields ($m = 6$). See Chapter 7 Section 7.4.1 for an explanation of the L1 neurons. The greyscale levels represent the levels of excitation for each of the L1 neurons. Darker levels are associated with more excited L1 neurons (Soltic, Wysoski & Kasabov, 2008)..	141

8.9. Wine dataset - average values and standard deviations of the 150 samples in each of the five different wine classes.	142
8.10. Scenario 5 dataset: accuracy (a) and number of L2 neurons <i>vs.</i> the distance threshold (S_θ), $m = 8$. The numbers next to each accuracy point represent the average number of L2 neurons created in the ten runs (n_{L2av}) (Soltic, Wysoski & Kasabov, 2008).	145
8.11. Population encoding of two values v_i and v_j is based on the Gaussian receptive fields. The values are chosen so they cause patterns of spikes, which appear out of sequence by 1 L1 neuron.	148
8.12. A theoretical pattern of weights of the two L2 neurons connected to the same set of L1 neurons and trained to differentiate between samples belonging to two classes, C_i and C_j	149
8.13. Two L2 neurons receiving spikes from six L1 neurons. As the spikes arrive the post-synaptic potentials of the two neurons, PSP_i and PSP_j , increase by different amounts depending on their synaptic weight values.	149
8.14. The $ w_{im} - w_{jm} $ values for the theoretical pattern in Fig. 8.12. As a result it is very likely that smaller (higher) input values are classified C_i (C_j).	149
8.15. The synaptic weights in one of the trained ESNN-PC-TR networks built in Scenario 2 where the ESNN-PC-TR was trained to distinguish between the two types of mineral water.	150
8.16. The $ w_{im} - w_{jm} $ values for the weights in Fig. 8.15.	151
8.17. Values of the ten samples (C_0, C_1) used to train the ESNN-PC-TR in Scenario 2.	151
8.18. The $ w_{im} - w_{jm} $ values for the weights in Scenario 4 where the ESNN-PC-TR was used to classify the two brands of wine.	152
9.1. Block diagram of the proposed FPGA implementation of the ESNN-PC-TR. ...	163
9.2. A simplified block diagram of Cyclone II FPGA: NIOS II soft processor core and the interfaces needed to connect to other chips (Altera).	164
9.3. The block diagram of the population encoding module. The encoder has an out-of-range detection circuitry to detect sensory data features too small to contribute to the membrane potential of the L2 neurons. (Zuppichich & Soltic, 2008).	165
9.4. The block diagram of the proposed parallel bit-wise rank order coding module based on an N-stage 32-bit shift register and a FSM controller. N is equal to $m \times s$, where m is the number of Gaussian receptive fields and s is the number of features per sensory data sample. (Zuppichich & Soltic, 2008).	167

9.5.	The block diagram of the proposed circuit for a spiking neuron and its synapse. Note that the circuitry required for the on-chip learning is omitted. Synapse weight values and the weight contributions are saved in two LUT. The neuron is modelled using two memories and a comparator. Note that there is one multiplier in this module. (Zuppichich & Soltic, 2008)	168
9.6.	The block diagram of the proposed distance measuring module that compares a newly created neuron against all pre-existing neurons for the same class of data. Note that there is only one multiplier in this module. (Soltic & Zuppichich, 2008)	169
9.7.	The block diagram of the proposed aggregation module. Note that there is only one multiplier in this module. (Soltic & Zuppichich, 2008).....	170
9.8.	Signals and timing during the L2 evolving stage. One L2 is created in 15 to 25 μ s but a possible aggregation introduces a further delay and one neuron is created and trained in 40 to 80 μ s. (Soltic & Zuppichich, 2008).	174
9.9.	Signals and timing during the classification stage. A sample is classified in 6 μ s (Soltic & Zuppichich, 2008).	175
10.1.	The multimodal taste recognition model based on the ESNN-PC approach.....	186
10.2.	An ESNN-PC embedded in a smart refrigerator.	187
10.3.	A ship with container cargo using embedded ESNN-PC modules for in-situ detection of unwanted species posing bio-security risks.....	187
10.4.	LPAM for local modelling in a distributed environmental monitoring system...	188

List of tables

3.1	The level of agreement between observed and predicted values (Monserud & Leemans, 1992).....	42
3.2	Climate variables describing the climate of each site.....	44
3.3	The number of samples for four pest insects used in this thesis.....	45
4.1	Example of two rules extracted by the LPAM	55
4.2	Details of the iris dataset. Averages (μ) and standard deviations (σ) of the features in the iris dataset are expressed in centimetres and rounded to 2 decimal places (2.d.p.).....	58
4.3	Two rules extracted by the LPAM for the iris Versicolour and iris Virginica dataset.	59
4.4	Details of the wine dataset. Averages (μ) and standard deviations (σ) of the features in the wine dataset are rounded to 2.d.p.....	61
4.5	Two rules extracted by the LPAM for the Class 0 and Class 1 wines.....	62
5.1	Climate variables used to model the establishment potential of the insect <i>P. citri</i>	68
5.2	Correlation matrix of the predictor variables for the insect <i>P. citri</i>	68
5.3	Details of the ECM cluster analysis.....	71
5.4	Clustering results for New Zealand. There are seven New Zealand locations (78%) in clusters where $r \leq 0.28$. The <i>P. citri</i> is more strongly associated with sites that are not clustered or associated with New Zealand sites.	71
5.5	Knowledge extracted by the LPAM.	73
5.6	The results for 20 selected locations. The first 10 locations were chosen because they were given the highest establishment potential estimates. The second 10 locations were given the lowest overall estimates.....	75
6.1	Range of three climate variables used to model the establishment potential of the following insects: <i>A. hartii</i> , <i>G. coffeae</i> and <i>X. perforans</i>	84
6.2	$\theta_{Optimal}$ for which each model performed the best when performance was measured in terms of overall accuracy a	91
6.3	The prediction results obtained using the different models for Bombay, where the species is established (1) and Wellington where the species has not yet established (0).....	99
8.1	The results of our survey of taste recognition systems, with references.	131

8.2	Details of Class1 and Class3 samples. Averages (μ) and standard deviations (σ) of Class1 and Class3 samples are expressed in nF and rounded to 2 s.f. (Soltic, Wysoski & Kasabov, 2008).....	141
8.3	Details of the six scenarios designed to test the ESNN-PC-TR taste recognition model (Soltic, Wysoski & Kasabov, 2008)	143
9.1	The number of multipliers in the Stratix IV E FPGA DSP family devices (Altera). FP – floating-point.....	160
9.2	Centre values c_i for Gaussian receptive fields ($m = 6$ and $m = 8$).	165
9.3	The accuracies of the FPGA and software implementations of ESNN-PC-TR when used for water and wine recognition.	171
9.4	Available FPGA resources and their usage.	172
9.5	Training times for software (t_{soft}) and FPGA (t_{FPGA}) implementations required to train an ESNN-PC-TR network on one sensory data sample (56 L1, $t_{dCPU} \approx 20 \mu\text{s}$) (Soltic & Zuppichich, 2008).....	173
9.6	Classification times for software (t_{soft}) and FPGA (t_{FPGA}) implementations required to classify one sensory data sample (56 L1, $t_{dCPU} \approx 20 \mu\text{s}$) (Soltic & Zuppichich, 2008).....	173

Abbreviations and acronyms

ANN	Artificial Neural Network
CART	Classification/Regression Tree
CBDS	Connectionist-based Decision System
CEM	Climatic Envelope Model
CI	Computational Intelligence
COS	Connectionist system
DA	Discriminant Analysis
DC	Direct Current
DENFIS	Dynamic Evolving Neuro-Fuzzy Inference System
DSS	Decision Support System
EA	Evolutionary Algorithms
ECF	Evolving Classifier Function
ECM	Evolving Clustering Method
ECOS	Evolving Connectionist System
EFuNN	Evolving Fuzzy Neural Network
EPSP	Excitatory Postsynaptic Potential
ESOM	Evolving Self Organizing Maps
ESNN-PC	Evolving Spiking Neural Network with Population Coding
ESNN-PC-TR	Evolving Spiking Neural Network with Population Encoding for Taste Recognition
FPGA	Field Programmable Gate Array
FuNN	Fuzzy Neural Network
GA	Genetic Algorithm
GAM	Generalized Adaptive Model
GARP	Genetic Algorithm for Rule-Set Prediction
GIS	Geographic Information System
GLM	Generalised Linear Model
GRF	Gaussian Receptive Field
GWR	Geographical Weighted Regression
HyFIS	Hybrid Neural Fuzzy Inference System
IEEE CIS	Institute of Electrical and Electronics Engineers

	Computation Intelligence Society
IDE	Integrated Development Environment
IF	Integrate-and-Fire
INNS	International Neural Network Society
IPCA	Incremental Principal Component Analysis
IPSP	Inhibitory Postsynaptic Potential
KNN	K Nearest Neighbours
LIF	Leaky Integrate-and-Fire
LOO	Leave One Out
LPAM	Local Probability-based adaptive Model
LR	Logistic Regression
LUT	Look-Up-Table
MLP	Multi-Layer Perceptron
PCA	Principal Component Analysis
PSP	Postsynaptic Potential
QSNN	Quantum spiking neural networks
RBF	Radial basis Function
ROC	Rank Order Coding
SNR	Signal-to-noise Ratio
SOM	Self Organizing Maps
SNN	Spiking Neural Network
SRM	Spike Response Model
TRC	Taste Receptor Cell
WKNN	Weighted K Nearest Neighbours

“I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.”

Snjezana Soltic

Acknowledgments

It is a pleasure to thank the many people who has helped me throughout my work on this thesis. Without the support of a number of people who helped me by answering my questions and reassured me in times of self-doubt, this thesis would never have been completed.

First and foremost, I would like to acknowledge the advice and guidance of my first supervisor Professor Nikola Kasabov. I truly could not have done this thesis without his help and support. Also, I would like to thank my second supervisor Dr. Paul Pang who provided encouragement and sound advice in the beginning when success was less certain.

I would like to thank Dr. Sue Worner from the Centre for Advanced Bio-protection Technologies at Lincoln University, New Zealand, for many useful insights into modelling predictive insect distributions. My sincere gratitude goes to Dr. Lora Peacock for allowing me to use her pest insect datasets. I appreciate the work that has gone into compiling them. I also thank Lora for her critical and useful comments while working on “Evolving connectionist systems in ecological modelling” and “A comparison of inductive and transductive models for predicting the establishment potential of the exotic scale, *Aspidiella hartii* (Cockerell), in New Zealand”. Her ecological knowledge was invaluable to me.

I thank Professor Andre Ponce de Leon F. de Carvalho from the Department of Computer Science, University of Sao Paulo, Brazil, for kindly providing the water and wine data used in the taste recognition part of this thesis.

I would like to thank the past and present members of the Knowledge Engineering and Discovery Research Institute at the Auckland University of Technology, who contributed in different ways to this work. I am thankful to Joyce D’Mello for her administrative assistance. This thesis has benefited from the comments and suggestions of Dr. Qun Song, Dougal Greer, Dr. Liang Goh and Richard Walton. I gratefully thank Dr. Simej Wysoski for the delightful discussions on thesis topics and for his assistance with my work on the taste recognition system.

I would like to show my appreciation to the School of Electrical Engineering at Manukau Institute of Technology for founding my research. Furthermore, I would like

to thank all my colleagues at Manukau Institute of Technology for providing a good academic environment and support throughout my study. I thank my graduate student Alan Zuppichich for his keen interest and fine work in assisting me in the development of the taste recognition system.

Last, but by no means least, I wish to thank my biggest supporters, my daughter Natasa, son Ivan and husband Tihomir for their tolerance. I specially thank Natasa for patiently reading my drafts.

Abstract

Ecological modelling problems have characteristics both featured in other modelling fields and specific ones, hence, methods developed and tested in other research areas may not be suitable for modelling ecological problems or may perform poorly when used on ecological data. This thesis identifies issues associated with the techniques typically used for solving ecological problems and develops new generic methods for decision support, especially suitable for ecological data modelling, which are characterised by: (1) adaptive learning, (2) knowledge discovery and (3) accurate prediction. These new methods have been successfully applied to challenging real world ecological problems. Despite the fact that the number of possible applications of computational intelligence methods in ecology is vast, this thesis primarily concentrates on two problems: (1) species establishment prediction and (2) environmental monitoring.

Our review of recent papers suggests that multi-layer perceptron networks trained using the backpropagation algorithm are most widely used of all artificial neural networks for forecasting pest insect invasions. While the multi-layer perceptron networks are appropriate for modelling complex nonlinear relationships, they have rather limited exploratory capabilities and are difficult to adapt to dynamically changing data. In this thesis an approach that addresses these limitations is proposed.

We found that environmental monitoring applications could benefit from having an intelligent taste recognition system possibly embedded in an autonomous robot. Hence, this thesis reviews the current knowledge on taste recognition and proposes a biologically inspired artificial model of taste recognition based on biologically plausible spiking neurons. The model is dynamic and is capable of learning new tastants as they become available. Furthermore, the model builds a knowledge base that can be extracted during or after the learning process in form of IF-THEN fuzzy rules. It also comprises a layer that simulates the influence of taste receptor cells on the activity of their adjacent cells. These features increase the biological relevance of the model compared to other current taste recognition models. The proposed model was implemented in software on a single personal computer and in hardware on an Altera FPGA chip. Both implementations were applied to two real-world taste datasets.

In addition, for the first time the applicability of transductive reasoning for forecasting the establishment potential of pest insects into new locations was investigated. For this purpose four types of predictive models, built using inductive and transductive reasoning, were used for predicting the distributions of three pest insects. The models were evaluated in terms of their predictive accuracy and their ability to discover patterns in the modelling data.

The results obtained indicate that evolving connectionist systems can be successfully used for building predictive distribution models and environmental monitoring systems. The features available in the proposed dynamic systems, such as on-line learning and knowledge discovery, are needed to improve our knowledge of the species distributions.

This work laid down the foundation for a number of interesting future projects in the field of ecological modelling, robotics, pervasive computing and pattern recognition that can be undertaken separately or in sequence.

Chapter 1

Introduction

1.1 Motivation

Worldwide, the necessity for bio-protection is increasing. Rapidly growing inter-country trade and tourism leads to increased biosecurity risk (Dobesberger, 2002; Peterson & Vieglais, 2001). Failure to manage the distribution of invasive pest species could seriously unbalance the ecosystems of an area. An invasion of pest species can have significant consequences for the environmental sustainability of and can substantially affect the economic development of a country (Cohen, 1998). This is particularly true for countries with smaller economies that rely heavily on their agriculture. The potential costs are considerable. For example, the failure to control the spread of the didymo alga could cost New Zealand \$57-\$285 million over the eight year period from 2004/05 to 2011/12 (NZ Ministry of Agriculture and Forestry, 2006).

Dealing with bio-protection threats and other ecological problems require the development of intelligent decision support systems (DSS) based on state-of-the-art sensor techniques, species identification systems and novel modelling techniques capable of efficient environmental and ecological data modelling, prediction and knowledge discovery (La Morgia *et al.*, 2008; Paterson *et al.*, 2008; Woodford, 2008; Park *et al.*, 2007; Gevrey *et al.*, 2003; Remm, 2004; Rushton *et al.*, 2004).

Ecological processes are difficult to model because they contain complex dynamic interactions that are not well understood (Watts & Worner, 2008; Baker, 2002; Bradshaw *et al.*, 2002; Worner, 2002; Fielding, 1999). This research aims to encourage the use of advanced computational intelligence (CI) methods, such as neuro-fuzzy networks, for extracting useful knowledge from ecological datasets. As will be shown in the course of this thesis, there are many ecological research groups across the world using traditional neural networks, mainly multi-layer perceptron (MLP) networks trained with a back-propagation algorithm. While MLP networks are appropriate for modelling complex nonlinear relationships, they have rather limited exploratory capabilities and therefore can only modestly contribute to a better understanding of ecological systems (Olden *et al.*, 2006; Kolman & Margaliot, 2005; Gevrey *et al.*, 2003). Furthermore, MLP networks cannot easily adjust their structure in an adaptive,

on-line way to accommodate new data and changes that can happen in the future (Kasabov, 2003a, 2007a). Clearly, these characteristics are limiting factors when the networks are used on ecological data which is complex, noisy and may change over time (Eyre *et al.*, 2005; Pearson *et al.*, 2002; Guisan & Zimmermann, 2000). To overcome the above problems, new improved methods are required, such as connectionist local and personalized evolving models (Kasabov, 2003a, 2007a).

While the main objective of this thesis was to develop new modelling techniques, the focus was on applying these techniques for modelling problems encountered in the area of biosecurity, particularly for modelling invasive pest insect distributions. In this thesis different modelling approaches are evaluated for the first time on bio-climatic pest insect data and the following two questions are answered: “Can local and ‘personalised’ modelling techniques be employed for modelling insect pest distributions to determine the risk of pest invasions?” and “What are the benefits of using these techniques for modelling pest distributions?”.

In addition, this research aims to increase the awareness of the wider connectionist community on the complexity and unique characteristics of ecological modelling problems. Although, studies on and guidelines for pest risk management have been produced by different government agencies to help researchers to conduct pest risk management (Baker, 2002), the researchers in the area of modelling species distribution dynamics are still facing two big obstacles that are usually not encountered in other research areas: (1) a lack of prior knowledge required to accurately model species-habitat relationships and to assess those models (Worner, 2002), and (2) a lack of good quality species distribution data (Rafoss, 2003; Baker, 2002). Particularly the real-world bio-climatic distribution data has characteristics typically not found in data from other modelling fields and therefore methods developed and tested in other research areas might not be appropriate for use on this data.

Recently, the uniqueness of the problems faced in ecological modelling has been recognised by the IEEE Computation Intelligence Society, IEEE CIS, (http://www.unesco-ihe.org/hi/sol/TF_CIEES/) and the International Neural Network Society, INNS, (http://www.unesco-ihe.org/hi/sol/SIG_CIEES/). They established two special interest groups solely dedicated to encouraging their members to develop methods that match the specific characteristics of environmental and ecological datasets.

This research was also motivated by the prospect of increasing the efficiency of environmental monitoring by introducing an intelligent taste recognition system (Gutiérrez *et al.*, 2008; 2002; Krantz-Rülcker *et al.*, 2001). The system can either be stationary or embedded into a robot. The robots could learn to taste, for instance, safe drinking water, and detect when the water becomes undrinkable. They could also express what they have learnt by using high-level knowledge representation, *e.g.* a IF-THEN rules. In this thesis we show a new concept of extracting knowledge from a trained spiking neural network with rank coded inputs. As will be shown in the course of this thesis, the use of spiking neurons for taste recognition in an adaptive system is a novel approach. There have been previous models that can perform taste recognition (Gallardo *et al.*, 2003; de Sousa *et al.*, 2002; Toko, 2000); however the model proposed here is more biologically realistic than those other models. The model is adaptive and its structure evolves as it learns to recognise tastants. The model explains what it has learnt about the tastants. The proposed taste recognition model was implemented in a field programmable gate array (FPGA) making the model suitable for integration in mobile robots used in monitoring applications where the tasting and classification of liquids is required.

1.2 Objectives

The main goal of this thesis was to propose a generic methodology for connectionist-based decision systems (CBDS) that would allow for (1) adaptation to new data, (2) knowledge discovery from real-world datasets, and (3) accurate prediction. The first objective was to propose a method for risk assessment. This thesis proposes an approach based on the dynamic evolving neuro-fuzzy inference system (DENFIS) and evolving clustering method (ECM) first used by N. Kasabov (1998). The model has a probability evaluation module for reducing the influence of noisy data samples. This approach was validated on two benchmark problems and then applied on four real-world bio-climatic datasets. In addition, it was used to prepare a predictive map for one pest insect.

The second objective was to evaluate local and ‘personalised’ modelling techniques for the purpose of building adaptive predictive models for knowledge discovery and their applicability in the field of ecological modelling. The models were evaluated in terms of prediction accuracy and their knowledge discovery ability.

The third objective was to propose a new biologically plausible neural network model. This model would be appropriate for taste recognition. This thesis proposes a model based on a very simple version of the integrate-and-fire spiking neurons with rank order coded inputs first used by S. J. Thorpe and his group for rapid visual processing (Gautrais & Thorpe, 1998; Van Rullen *et al.*, 1998). The novelty of our model is in using a layer of receptive fields to perform a population encoding of input signals before they are presented for processing to the neurons with rank order encoding. While the population encoding enhances the accuracy of the new model compared to the accuracy of the same network without this encoding, it also simulates the influence of one taste receptor cell on the activity of its adjacent taste cells (Huang *et al.*, 2005). Furthermore, the proposed model evolves its structure through learning. Therefore it is able to learn new patterns as they become available. Accordingly, we propose a new method of knowledge extraction from the proposed model. The spiking neurons, the cell-to-cell interaction layer, the adaptive nature of the proposed model and knowledge discovery make our simulations of the taste perception more biologically realistic than any other published taste recognition model. The model was used on two real-world taste datasets.

The final objective was to propose a low-cost hardware implementation of the taste recognition model. Our implementation was hosted in an Altera FPGA chip. The highlight of our FPGA implementation is that all its modules (including on-line learning) are hosted on a single FPGA chip. The FPGA-implemented taste recognition model was used on the same real-world taste datasets as its software simulation.

1.3 Research questions

In the course of this thesis, the aim is to highlight important issues and provide solutions to a number of general and specific research questions:

1. How can we enhance DSS to enable them to efficiently process dynamic real-world data and perform knowledge discovery? This will be demonstrated on case study problems of ecological modelling.
2. How can we present the information learned by such a DSS to enhance the understanding of information contained in the created model?

3. How can we take advantage of the temporal information processing in one model? This will be demonstrated on the taste recognition problem and will lead to a more biologically plausible model of taste recognition.
4. How can data collected by sensors be represented for analysis by spiking neurons in a taste recognition model? What is the advantage of doing that?
5. Are local modelling techniques suitable for modelling complex nonlinear species-environment relationships and what are the advantages of using these models?
6. Can local models be used to aid in the discovery of relationships between climatic factors and pest insect distributions?
7. Is ‘personalised’ modelling suitable for building predictive pest distribution models?
8. Can a spiking neural network be used for taste recognition tasks, *e.g.* to classify beverages?
9. What are the benefits of a hardware implementation of a connectionist system? This will be demonstrated on a novel taste recognition system built using biologically plausible spiking neurons.

1.4 Contributions

This research has made a number of contributions to the field of DSS predictive modelling and risk analysis, particularly in the field of ecology and pest insect management. The details of these contributions are given in Chapters 4, 5 and 6. First, we have proposed a local probability adaptive model named LPAM. The model was used for predicting the establishment of pest insects in new locations and to obtain new knowledge about what makes an insect invasive. This model was also used to create a risk map for a pest insect. To the best of our knowledge, a risk map for that pest insect had not been previously generated. Furthermore, we have assessed the advantages and disadvantages of building ‘personalized’ models of pest insect distributions. The accuracy and exploratory capabilities of the ‘personalized’ models were compared to the accuracy and explanatory capabilities of global and local models using three real-world eco-climatic distribution datasets.

This research work also made a number of contributions to the field of pattern recognition. The details of these contributions are given in Chapters 8 and 9. An adaptive model, named ESNN-PC, based on biologically realistic neurons was proposed and was first published in (Soltic, Wysoski & Kasabov, 2008). The model was used for taste recognition (ESNN-PC-TR) and we show that it is able to learn new tastants as they become available. A layer that simulates cell-to-cell interaction that may occur within taste buds is included in the model in the form of a Gaussian receptive field (GRF) layer. ESNN-PC-TR has been implemented in a software simulation and in hardware. An Altera Cyclone II FPGA was chosen for the hardware implementation. The FPGA implementation was optimised for speed and network size while the accuracy was kept comparable to the accuracy of the software simulations. The whole system, including the on-line learning circuitry, was implemented on one FPGA chip. The implementations, both software and hardware, have been evaluated on two real-world taste datasets. The model was used to discover knowledge about what was ‘hidden’ in those datasets.

Fig. 1.1 gives an overview of the contributions made in this thesis. On the left-hand side of the figure are the existing CI methods that are used in the new adaptive models proposed here. Design of new techniques for solving real-world problems in the field of ecological modelling was one of the main driving forces in this research work and therefore the right-hand side in Fig 1.1 shows the modelling problems from ecology that this research work contributed to.

It can be seen from Fig.1.1 that we built personalized models using a weighted k-nearest neighbour algorithm (WKNN) and we built global models based on generalized linear models (GLM). The new models proposed in this thesis, *i.e.* LPAM and ESNN-PC (ESNN-PC-TR), are shown in the middle. Both models are adaptive and evolve their structure to new data and changing environments. In addition, they can be used for knowledge discovery. While the proposed LPAM approach has been used for pest distribution modelling we could not see why this approach could not be used for other applications where knowledge discovery and adaptation to new data are desirable. Likewise, the proposed ESNN-PC offers considerable potential in modelling non-linearly separable data by means of spatio-temporal encoding.

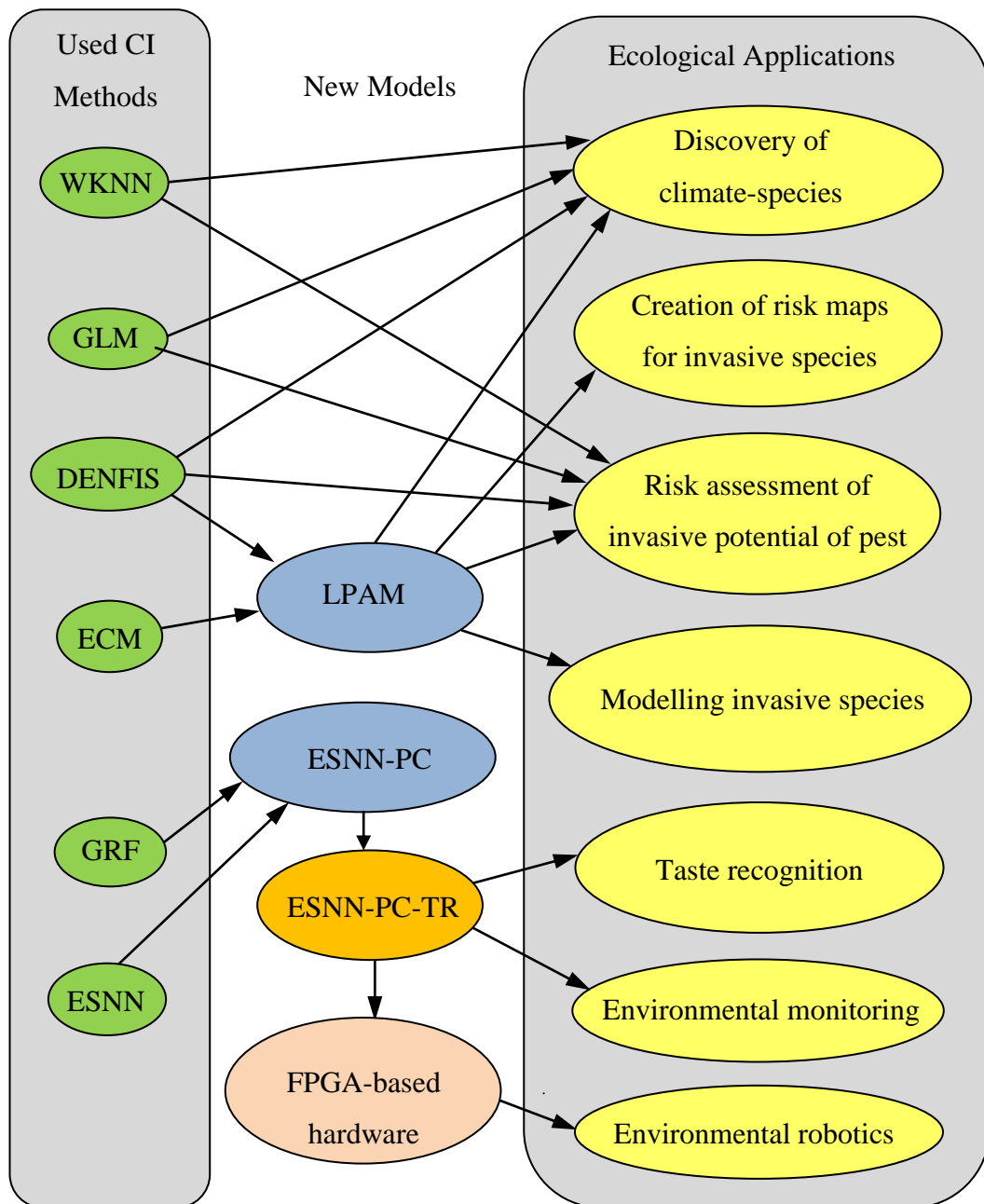


Fig. 1.1. Overview of contributions of this thesis. The arrows going from the CI Methods to the Ecological Applications via New Models show the key contributions of this research in both computational models and their application for ecological modelling.

1.5 Research methodology

We have designed two models, LPAM and ESNN-PC, using the following six steps:

1. Problem specification
2. Ecological case study identification
3. Method development and validation
4. Method application on the ecological case study data

5. Result validation
6. Conclusion about the proposed method and analysis of its future applications.

In Step 1 we identified that there is a need for adaptive predictive models for local knowledge discovery in ecological applications. Next, we identified that the task of predicting pest distributions would benefit from such a model (Step 2). It is known that neural networks are highly dependent on the quality and quantity of training data.

Therefore, in Step 3, an analysis of the data was conducted that included studying the issues associated with data collection methods and consequently the data quality. This analysis increased our awareness of the unique characteristics of real-world bio-climatic pest distribution datasets. A literature review of the existing modelling techniques used for building predictive models in ecology was also conducted. During the literature survey the typical problems reported in the published approaches were identified and a new dynamic method (LPAM), for building adaptive predictive models was developed and validated on two benchmark datasets. The method has two main features: it builds a dynamic model, and the resulting model can be used for knowledge discovery. In Step 4 the model was used for assessing the establishment potential of pest insects. The specific pest insects were chosen so that they were relevant to New Zealand's biosecurity. The performance of the new approach was compared to the performances of various local, global and personalized pest insect distribution models built using inductive and transductive techniques (Step 5). The models were assessed in terms of their accuracy and explanatory capability using leave-one-out (LOO) cross-validation. Finally, a possible extension to the model was proposed in Step 6.

The design of ESNN-PC followed the same design path as design of LPAM. First, our review of the current knowledge on taste perception and the existing taste recognition models found that the existing taste recognition models are static and cannot easily adapt to new data (Step 1). Therefore, we have concentrated on designing a more realistic model using biologically realistic neurons, encoding schemes and an on-line learning method. We identified that dynamic taste recognition models could be used for the environmental monitoring of, for instance, the quality of water or hazardous spills (Step 2). As a result, we designed an adaptive model adhesive to the current knowledge about taste perception (Step 3). The ESNN-PC model was first implemented in software and tested and evaluated on two real-world taste datasets (Step 4 and 5). Prior to implementing the model in an FPGA chip, a literature survey of published papers on

FPGA implementations of artificial neural networks was conducted. The FPGA implementation of ESNN-PC was proposed and tested on the same dataset as the software version of the network. That allowed us to explore and emphasise the benefits of hardware implementations over software implementations as well as the limitations of both implementations. In Step 6, we proposed future extensions and applications of ESNN-PC.

1.6 Published work

Different aspects of this research have been published to date in the following publications (Note that the list is given in chronological order):

- Soltic, S & Zuppicich, A. (2008). Comparison between software and FPGA implementation of ESNN in terms of processing speed. 15th Electronics New Zealand Conference ENZCon08, 24-25 November, Auckland, 56-61.
- Zuppicich, A & Soltic, S. (2009). FPGA implementation of an evolving spiking neural network. 15th International Conference on Neural Information Processing ICONIP'08, Workshop on Neurocomputing and Evolving Intelligence, 25-28 November, Auckland, New Zealand, 23-24, LNCS, vol. 5506/5507.
- Soltic, S., Wysocki, S. G. & Kasabov, N. (2008). Evolving spiking neural networks for taste recognition. International Joint Conference on Neural Networks (IJCNN2008), 1-6 June, Hong Kong, 2092-2098.
- Soltic, S. & Peacock, L. (2006b). A comparison of inductive and transductive models for predicting the establishment potential of the exotic scale, *Aspidiella hartii* (Cockerell), in New Zealand. Bulletin of Applied Computing and Information Technology, 4 (2), October 2006.
- Soltic, S. & Peacock, L. (2006a). Evolving connectionist systems in ecological modelling. Presented at the 1st Korean-New Zealand Joint Workshop on Advance of Computational Intelligence methods and Applications, 8 Feb, Auckland, New Zealand.
- Soltic, S., Pang, S., Peacock, L. & Worner, S. (2004a). Evolving computation offers potential for estimation of pest establishment. International Journal of Computers, Systems and Signals, 5 (2), pp. 37-44.

- Soltic, S., Pang, S., Kasabov, N., Worner, S. & Peacock, L. (2004b). Dynamic neuro-fuzzy inference and statistical models for risk analysis of pest establishment. International Conference on Neural Information Processing, ICONIP, LNCS 3316, 971-976.
- Soltic, S., Pang, P., Worner, S. & Peacock, L. (2003). Evolving computation offers potential as a probabilistic evaluation for pest risk maps generation. In N. Kasabov & Z. S. H. Chan (Eds.). Proceedings of the Conference on neuro-computing and evolving intelligence 2003, NCEI'03, 20-21 Nov, pp. 104-106.

1.7 Thesis structure

This thesis is divided into ten chapters, including this introductory chapter. Chapters 2 and 3 are supporting chapters for Chapters 4, 5 and 6, and Chapter 7 supports Chapters 8 and 9.

Chapter 2 gives a survey of modelling techniques for DSS concentrating on artificial neural networks. In Chapter 3, special attention is given to the modelling techniques in the field of ecological modelling and a survey of the benchmark works from this field is given. The challenges of assessing the risk of exotic pest insect invasions are also addressed. Furthermore, the real-world eco-climatic insect distribution data used in experiments described in Chapters 5 and 6 is introduced and its preliminary analysis is undertaken.

Chapter 4 presents a new adaptive model based on local probability for risk prediction, named LPAM. The real power of this dynamic model is in the knowledge extraction facility that accompanies the inference engine. The knowledge is delivered via IF-THEN fuzzy rules and this chapter proposes a visualization method for these rules. The chapter also shows the results of using LPAM on two benchmark problems.

Chapter 5 shows the results of using LPAM for assessing the establishment potential of a pest insect. We validate the model presented in Chapter 4 and compare the obtained results to the results of two more recent works where traditional neural networks were used on similar datasets.

Chapter 6 analyses global, local and 'personalized' models of pest insect distributions created through inductive and transductive reasoning. Four models, including LPAM,

were built and evaluated on three eco-climatic insect distribution datasets. The focus in this chapter is on the fundamental differences between these modelling approaches. The chapter also explores the effect of probability thresholds, used to dichotomise the likelihoods of pest occurrence at a location obtained by the models into presences and absences, on the performance of all four models.

Chapter 7 provides an introduction to models of spiking neurons and gives the supporting information and background to contributions presented in Chapter 8 and 9. This chapter looks at reasons why spiking neurons are more biologically plausible than traditional neurons for modelling human neurons and the benefits of using spiking neurons in modelling the human sensory systems. It also reviews various coding schemes used to model the information encoding in the human brain.

Chapter 8 presents a new evolving spiking neural network model (ESNN-PC). We used the model for modelling taste recognition (ESNN-PC-TR). First, the biological models of taste-coding are covered. Next, the results of our survey of the benchmark taste recognition systems are given. We discuss the shortcomings of using traditional artificial neural networks as pattern recognition tools in modelling taste perception. This is followed by a detailed description of the proposed taste recognition model and the testing undertaken to evaluate its performance. The model was used on two real-world taste datasets whose characteristics are presented in this chapter. It is demonstrated that the spiking taste recognition system can model the tasting of water and wine. Finally, the model is compared to existing work. While the proposed ESNN-PC has been used for taste recognition (ESNN-PC-TR) we could not see why this model could not be used for other applications where perception and sensory information is collected in a brain-like way (*e.g.* sound, smell, vision, ...) and where adaptive structures and an exploration of temporal patterns are desirable.

Chapter 9 gives the details of the proposed FPGA implementation of ESNN-PC. It provides schematic diagrams of all modules and time analysis of signals during learning and classification. Also, the implementation's processing speed and resource usage are evaluated. The chapter begins with a survey of the published literature on FPGA implementations of artificial neural networks. It addresses the main challenges of hosting a neural network in FPGA and the techniques used to circumvent these problems.

Chapter 10 concludes and summarises the entire thesis. It includes a review of the strengths and weaknesses of the LPAM and ESNN-PC models. It also gives directions about how this work can be further utilized in the fields of pattern recognition, robotics and pervasive computing.

Chapter 2

Connectionist-based modelling techniques for adaptive decision support systems: a review

This chapter gives a survey of modelling techniques for DSS concentrating on techniques which are inspired by the structure and operation of the human brain. In this chapter, the focus is on knowledge based DSS built using traditional artificial neural networks where input signals are rate coded and represented as numbers. The chapter also highlights the benefits of using evolving connectionist systems for real-world applications where a better understanding of modelled systems is required.

Furthermore, inductive versus transductive reasoning and global, local and ‘personalised’ modelling are compared. All these approaches are explored further in Chapter 4 and 5 using an ecological case study.

2.1 Introduction

Decision making in a complex and dynamic field is very difficult. A decision made can have an enormous impact on the life of a group of people (*e.g.* an environmental decision) or can affect an individual’s wellbeing (*e.g.* a medical decision). For instance, a decision that would allow an animal to be imported to Country A from Country B, where the animal could have been infected by a disease, could destroy the meat industry in Country A.

The development of DSS started in the 1960s. At the beginning, DSS were mainly developed to help business managers make key decisions (Turban, Aronson & Liang, 2005). In the 1970s books and journal articles related to DSS started to emerge, and the first international conference dedicated solely to DSS was held in 1981. This intensified the research on DSS. Development of computer and communication technologies further increased DSS popularity. More recently, web-based and web-enabled DSS have been developed (Han *et al.*, 2008; Karacapilidis, 2006). Today, DSS are not used only for business purposes, researchers from different research fields (*e.g.* medicine, bioinformatics, environmental science, *etc.*) have recognised the importance of DSS and

have come to appreciate the help these systems can offer. They not only use traditional decision support methods, but also propose new methods and technologies that are suitable for decision making in their own areas of expertise.

DSS can be found in almost all industries where information systems are used. The complexity and the availability of these systems range from off-the-shelf products, *e.g.* StrongWare[®], to systems designed to tackle very specialized, narrow and specific problems, such as decision aids for time critical targeting applications (McGraw, Lammers & Steinman, 2004; VonPlinsky & Crowder, 2002), waste management (Gomes, *et al.*, 2008), wildlife management (Paterson *et al.*, 2008), internet shopping (Mohanty & Bhasker, 2005) and international risk management (Han *et al.*, 2008). In the past few years, DSS have been utilized in a variety of tasks, such as:

- Marketing - design of a line of substitute products (Alexouda, 2005);
- Politics - analysis of political events (Blanning & Reinig, 2005);
- Finance - risk analysis and forecasting of evolving economic clusters in Europe (Kasabov *et al.*, 2000);
- Telecommunications - modelling the robustness of wireless networks under different operating conditions (Bose, Eryarsoy & He, 2005);
- Pest management - supporting and improving fruit production in New Zealand (Woodford *et al.*, 1999).

Since DSS have been used in a wide range of applications they have different meanings to people from different fields. Consequently, DSS differ in scope. Some DSS are created for use by only one user while others are created to be used by many users in an organisation or are even available to users through the Internet. Sauther (1997) defines DSS as computer-based systems “that bring together information from a variety of sources, assist in the organization and analysis of information and facilitate the evolution of assumptions underplaying the use of specific models“. On a smaller scale, a less ambitious definition might describe a DSS as a system, tool or technology that supports the decision-making process. In the latter case, the development of DSS is often influenced by the characteristics of its purpose and the availability of modelling data. Therefore, the evaluation of the system’s credibility is done in the context of its application. This is particularly important when analysing the performance of a predictive system where the true output values cannot be obtained.

DSS are categorised using a number of different criteria (Marakas, 2003). Knowledge based DSS are the most sophisticated DSS. They are computer-based systems able to search for hidden patterns in the problem space. Knowledge based DSS are a major point of interest in this research work and they are described in more detail in the next section.

2.1.1 Knowledge based DSS - methods and tools

Knowledge based DSS are intelligent systems with expert knowledge in the problem domain. The tools used to build these DSS are able to extract the most important features of a particular problem. These features are then used to choose between possible solutions. One of the main requirements for tools used in building knowledge based DSS is that they must be able to learn in an on-line mode. Furthermore, they must be able to accommodate new data and information without forgetting already learnt relationships. They must also have knowledge representation facilities (Kasabov & Fedrizzi 1999). Artificial intelligence techniques such as neural networks, fuzzy systems, and genetic algorithms (GA) partially meet these requirements.

A general framework of a connectionist-based decision support system (CBDS) is given in (Kasabov, 2003b). The framework consists of 5 parts shown in Fig. 2.1, where the Pre-processing unit is responsible for data filtering and feature extraction, the Neural network modules are trained on data and contain knowledge, the Higher-level knowledge unit produces final decisions, the Adaptation unit adjusts the system's structure and functionality for a superior performance and the Rule extraction unit explains the knowledge in an appropriate way.

In this work the focus is on CBDS that facilitate: (1) adaptation to new data, (2) knowledge discovery and (3) accurate prediction/classification. Fig. 2.2 shows our approach. Different types of artificial neural networks (ANN), traditional and spiking, were employed for two distinct problems. Firstly, a new approach was used for predicting the establishment potentials of different exotic pest insects. Secondly, a spiking neural network model was proposed and used for taste recognition. This model was implemented into an FPGA device. The proposed models are based on ANN and therefore the relevance of ANN when building DSS is described next.

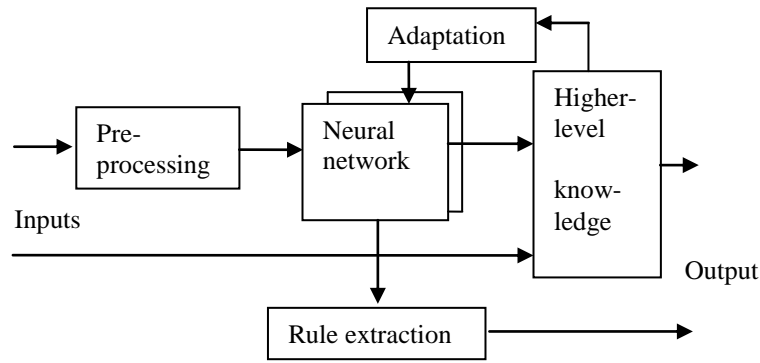


Fig. 2.1. A framework of a connectionist-based decision system proposed by Nikola Kasabov (Kasabov, 2003b).

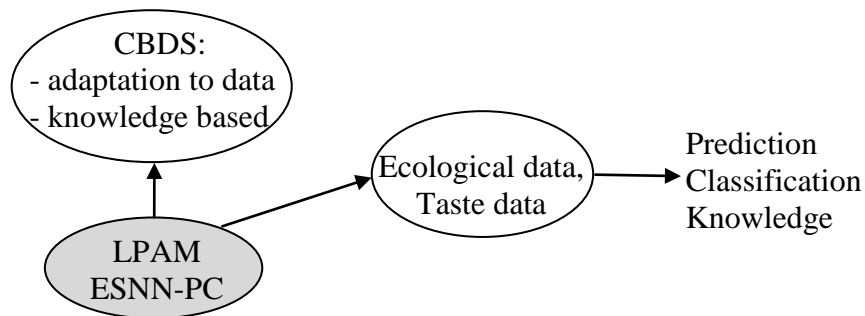


Fig. 2.2. A graphical representation of the approach taken in this thesis.

2.2 Artificial Neural Networks

2.2.1 Artificial Neural Networks for DSS

ANN have proved useful for modelling complex systems. A typical ANN is made up of a large number of simple processing units (neurons) highly interconnected in a structure than approximates the human brain. As a part of a decision making system, neural networks are able to evaluate data in a dynamic environment (Kasabov *et al.*, 2000). In the decision support context, a neural network, or a group of networks, receives a set of data, processes it and produces an output that can help the users to make a decision (Fig. 2.3).

Knowledge based neural networks are particularly interesting for building DSS because they provide facilities for knowledge extraction and insertion. Knowledge based neural networks manipulate data and knowledge. In these networks, knowledge can be expressed in the form of IF-THEN rules (Kasabov, 2003a). For example, a simple proposition rule states that IF $x_1 = A$ AND/OR $x_2 = B$ THEN $y = C$. More complex rules can be built. For instance, the first order Takagi-Sugeno fuzzy rule states that IF x_1

$= A \text{ AND } x_2 = B \text{ THEN } y = c + ax_1 + bx_2$, where A and B are fuzzy values and a , b , and c are constants. There are fuzzy rules in which importance (DI) and certainty (CF) degrees are used to represent the importance of each of the conditions to the output and strength of the rule, respectively. For example, IF $x_1 = A$ with DI_1 AND $x_2 = B$ with DI_2 THEN $y = C$ with CF_c , where DI_i represent the importance of each of the conditions to the rule output, and CF_c represents the strength of the rule.

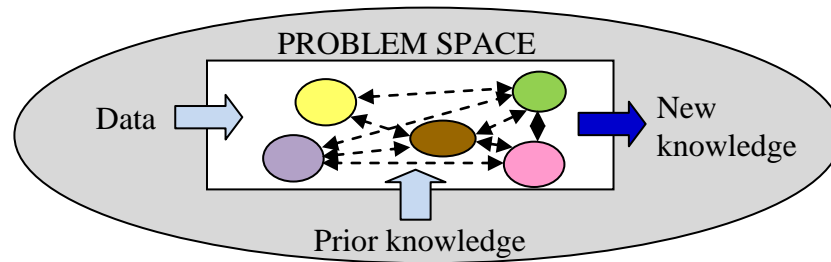


Fig. 2.3. An illustration of the knowledge based DSS framework. The circle nodes represent different neural networks working together in a DSS.

Knowledge based neural networks allow rule insertions, rule extractions, adaptation and reasoning. They can be built as a combination of different technologies in a manner that overcomes the drawbacks of the individual techniques. For example, fuzzy neural networks (FuNN) use an ANN and fuzzy logic system (Cherkassky, 1998). Most traditional ANN, such as MLP networks, suffer from the black box syndrome because they do not provide any knowledge extraction facilities or indicate how the decision was made. Thus, the knowledge that traditional ANN acquire is hard or impossible to extract. On the other hand, fuzzy logic systems are good at explaining their decisions but they are poor learners (Lin & Lee, 1991). FuNN have all the properties of neural networks, they can be trained to approximate data, they can generalize and they have the ability to learn. Furthermore, they have the properties of fuzzy systems; they are able to model uncertainty, they can reason with imprecise information and are good at explaining their decisions. Furthermore, they can be trained and rules can be inserted and extracted (Kasabov, 2001). The characteristics of FuNN have been summarized in (Kasabov & Fedrizzi, 1999):

- FuNN can be trained to approximate data;
- FuNN can be used to deal with knowledge in the form of fuzzy rules;
- FuNN are robust to catastrophic forgetting;
- FuNN can be used as replicators;
- FuNN can work both on real input data and fuzzy input data;

- When structural learning with forgetting and consecutive pruning is applied, a FuNN's structure becomes a skeleton structure, *i.e.* it contains only the important input, rule and output nodes, and the valuable connections between them.

An algorithm named the evolving fuzzy neural network (EFuNN) has been proposed for on-line, adaptive applications in (Kasabov, 1998). EFuNN are able to learn quickly, they adjust to dynamic changes in the operating environment and allow for the extraction of rules. These networks have been reported as good performers for decision making on mortgage approval (Kasabov, 2002). EFuNN were also used for adaptive speech recognition (Kasabov, 1999), camera operations recognition (Koprinska & Kasabov, 2000), classification of handwritten digits (Ng *et al.*, 2004), odour recognition (Zanchettin & Ludermir, 2004) and sensor management (Kong *et al.*, 2007). Recently, an image recognition architecture based on EFuNN for analysis of images of pest damage to apples has been proposed (Woodford, 2008).

2.2.2 Artificial neurons

Artificial neurons are simple processing units used to build an ANN. A generic artificial neuron (Fig. 2.4) receives signals via a number of connecting links (synapses). Each neuron is in a unique inner state. The inner state changes as the input signals are received. In a conventional artificial neuron, the input signals are reduced to a single value by an integration function g . Then, an activation (transfer, squashing) function f takes this single value and produces an output. The integration function g is usually the addition function and the activation function f is generally chosen to be monotonic (Bishop, 1995). The output signals are typically in the range of $[0, 1]$ or $[-1, 1]$. The input signals are normally influenced by a set of synaptic weights.

The McCulloch-Pitts model of the neuron (McCulloch & Pitts, 1943), also called the threshold logic unit, is one of the simplest models of biological neurons (Fig. 2.5). A McCulloch-Pitt unit receives and sends only binary values $x_i \in \{0, 1\}$. When a neuron receives a 1 signal via one of its input, its inner state is incremented by 1 unit. The neuron outputs 1 if its inner state exceeds some threshold value θ , and 0 if this threshold is not exceeded. The threshold θ is some fixed real number, there are no synaptic weights associated with the inputs and the transfer function f is a step function:

$$f\left(\sum_i x_i\right) = \begin{cases} 0, & \sum_i x_i < \theta \\ 1, & \sum_i x_i \geq \theta \end{cases} \quad (2.1)$$

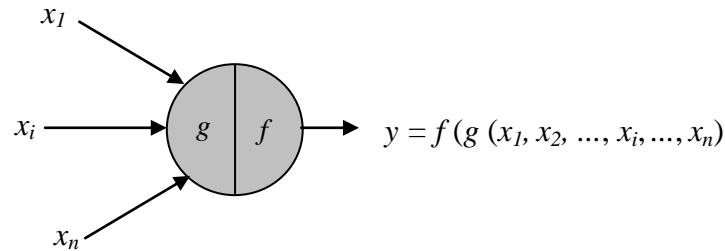


Fig. 2.4. A conventional neuron receives n inputs and produces one output (y). g is an integration function and f is an activation function. The graphical representation of neurons is adopted from (Rojas, 1996).

The McCulloch-Pitts neurons are able to solve two-class classification problems. The decision boundary between these two classes is linear. If the problem is from a two-dimensional input space the decision boundary is a line. If the problem is from an n -dimensional space, the decision boundary is a linear $n-1$ dimensional hyperplane. This processing unit is able to process the AND, OR and NOT functions.

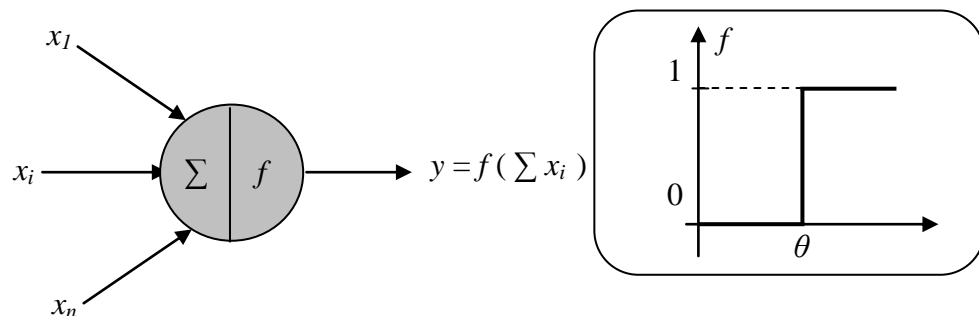


Fig. 2.5. The McCulloch-Pitts neuron and a step transfer function. The inputs x_i and output y are Boolean values. This unit can process only inputs equal to 1 and 0, *i.e.* $x_i \in \{0, 1\}$. Furthermore, a sample fed to one McCulloch-Pitts neuron causes either the 0 output or the 1 output.

In 1958 Frank Rosenblatt (Rosenblatt, 1958) proposed a more general threshold model of the biological neuron (Fig. 2.6). This model has n inputs, each associated with a weight w_i which is modelling the synaptic efficiency of the connection. The weights are adapted by a learning algorithm. The inputs and outputs are real numbers. The networks of these threshold units are called perceptrons.

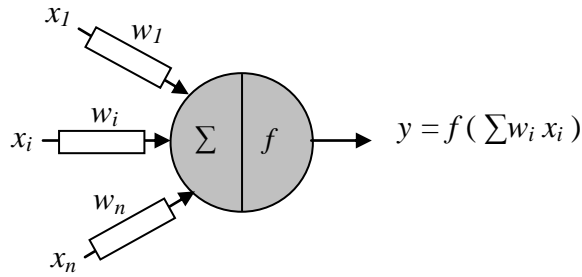


Fig. 2.6. The perceptron neuron (Rosenblatt, 1958). The weights w_i model the synaptic efficiency of the connections between two perceptrons. The inputs x_i and output y are real values. Different activation functions have been proposed over the years.

Single-layer perceptrons are able to solve problems that are linearly separable. These Perceptron Networks are the building blocks for MLP networks, one of the most popular types of neural networks. MLP neural networks have a number of hidden layers. Each layer in a MLP network has a set of adaptive weights and continuous inputs. The neurons have a differentiable activation function and the weights are trained by the powerful error back-propagation learning algorithm (Kasabov, 2007). When a sigmoid activation function is used the neurons are called sigmoid threshold gates. The sigmoid threshold gates were found to be a better model for a neuron than the Boolean threshold gates (Maass, Schnitger & Sontag, 1991). MLP networks can perform the non-linear separation of the input problem space and can solve the XOR problem.

Spiking neurons are a more biologically plausible model for biological neurons. Furthermore, spiking neurons are more computationally powerful than the McCulloch-Pitts and sigmoid neurons (Maass, 1997b). However, the computing paradigm of spiking neurons is quite different from the one employed by the traditional rate-coded neurons. The main difference between traditional neurons and spiking neurons is that spiking neurons communicate with spikes (Fig. 2.7) rather than numbers. In rate-coded ANN, each neuron sends its activation level to the post-synaptic neurons as a continuous value, typically a floating point number in the range $[0, 1]$. This continuous value represents the mean firing rate of that neuron and the exact timing of the spikes has no role.

The spikes arrive from the afferent neurons via inputs in time and influence the behaviour of the spiking neuron, changing its inner state (post synaptic potential or *PSP*). When the *PSP* of a spiking neuron exceeds a certain threshold value the neuron itself generates a spike. This type of neurons is very important in this thesis and a

separate chapter, Chapter 7, is dedicated solely to the introduction of spiking neuron models and information encoding using spikes.

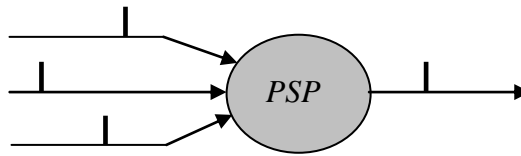


Fig. 2.7. Spiking neuron receiving three spikes and generating one spike. The spikes arrive via three synapses influencing the neuron's PSP. When the PSP exceeds a certain threshold the neuron itself generates a spike.

2.2.3 Connectionist Systems

Neural networks are also referred to as connectionist systems (COS). A COS, as seen in this thesis, consists of simple processing units and is able to learn from the samples from a problem space. It is characterised by its structure, type of neurons, its parameter set and weights, learning algorithm and goal function (Kasabov, 2003a, p. 12). MLP, radial basis function (RBF) (Kasabov, 2007), FuNN and self-organising maps (SOM) (Kohonen, 1982) are the most important examples of COS. As is stated later on, COS are the most frequently used type of neural network in ecological modelling.

Learning is very important for the success of a COS. There are two main types of learning, supervised and unsupervised learning. In supervised learning, a COS is given a set of input-output examples to learn the input-output relationship. An error correction algorithm to minimize output errors (gradient descent, least-mean-square, *etc.*) is employed and once the relationship is learnt the COS is ready to classify new data examples with unknown input-output relationships. When the examples lack the output information, unsupervised learning can be used. Unsupervised learning is used for data clustering where the structure of the data is discovered so that data samples that belong to the same cluster are similar to each other and data samples that belong to one cluster are different from the data samples of the other clusters.

The main disadvantages of COS are that they (1) suffer from catastrophic forgetting, (2) lack a knowledge representation facility, (3) have long training times and (4) are unable to change their structure to accommodate new data (Kasabov, 2003a, p. 24-25). Furthermore, the structure of a COS must be designed before the COS is trained. This is usually a daunting task for networks comprising hidden layers because there are no guidelines to determine the number and size of the hidden layers. Although the structure

of a COS is typically optimized by trial and error, genetic algorithms (GA) have been used to optimize COS structures (Castillo *et al.*, 2000; Marshall & Harrison, 1991). The main disadvantage of GA optimization procedures is their slow searching speed making GA inappropriate for the optimization of an on-line COS.

2.2.4 Evolving Connectionist Systems

Evolving connectionist systems (ECOS) overcome the problems related to the static structure of COS. Compared to COS, the structure of ECOS is not static, but rather it continuously adapts in time through its interaction with the environment and other systems (Fig. 2.8). An ECOS is necessary in situations where the complete set of training data is not known in advance. ECOS are suitable for on-line clustering as well as for on-line classification (Kasabov, 2007a; Kasabov, 2003a).

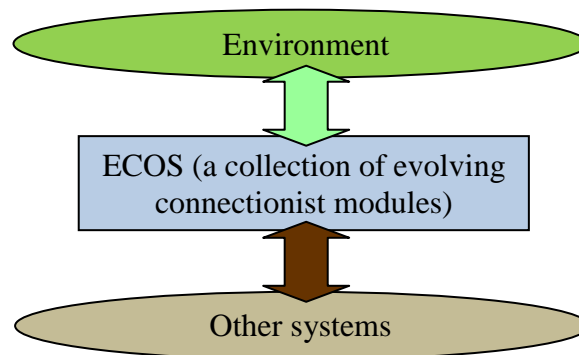


Fig. 2.8. ECOS interacts with the environment and other systems.

Kasabov defines an ECOS as a “multi-modular connectionist architecture that facilitates the modelling of evolving processes and knowledge discovery” (Kasabov & Benuskova, 2005). Therefore, an ECOS may consist of one or many evolving connectionist modules, with each module possibly using different learning algorithms and structure elements. All modules must be able to evolve in a lifelong manner. Furthermore, ECOS tools do not suffer from catastrophic forgetting and they facilitate knowledge storage without the need to store past experiences. Examples of ECOS are:

- EFuNN (Kasabov, 2001);
- Hybrid neural fuzzy inference systems (HyFIS) (Kim & Kasabov, 1999);
- DENFIS (Kasabov & Song, 2002);
- Evolving classifier function (ECF) (Huang, Song & Kasabov, 2005).

Additionally, a number of COS have been adapted to meet ECOS requirements, such as evolving self organizing maps (ESOM) (Deng & Kasabov, 2000) and incremental principal component analysis (IPCA) (Ozawa *et al.*, 2005).

The ability to adapt its structure in time makes the ECOS particularly suitable for modelling ever changeable ecological systems. Current models of ecological systems are mainly static and their suitability for predicting changeable future environments has been questioned (Guisan & Zimmermann, 2000). Having a model that can change its structure to accommodate new data as it becomes available would provide more accurate future predictions. Furthermore, the ECOS has many other desirable features, such as one-pass learning and knowledge extraction (Kasabov & Benuskova, 2005), that make it a suitable tool for ecological modelling particularly for modelling the establishment potential of exotic species. The former feature allows for building of fast on-line predictive models while the latter one makes models suitable for not only predicting future outbreaks but also delivering a set of rules explaining the current species distributions.

2.3 Inductive and transductive reasoning

Inductive reasoning is the most frequently used inference tool in machine learning. Inductive methods estimate a global model (a rule or a function) from a set of training data samples from the problem space. This model is then used to predict outcome values for an unseen data vector from the same space (deduction). Inductive models do not utilize the additional information related to the individual data samples. Regression analysis, largely used in ecological modelling, is an example of inductive reasoning.

In contrast, transductive reasoning techniques are used to build an individual (personal) model to suit a single data sample (Vapnik 1998). These techniques explore the knowledge about the location of the sample in the problem space and adjust the model to this particular sample. Transductive techniques promise better performance than inductive techniques for applications where the emphasis is on a particular data sample rather than the entire problem space. They also offer advantages over inductive reasoning for small and moderate sets (Derbeko, El-Yaniv & Meir, 2003; Blum & Chawla, 2001). This type of modelling was proven to suit clinical and medical applications, where the focus is usually on a single patient rather than on a group of people (Kasabov & Pang, 2004). In the past few years, transductive reasoning has been

successfully implemented for a range of problems, *e.g.* medical diagnostic (Kukar & Grošelj, 2005; Song & Kasabov, 2005; Kukar, 2004; Anderson *et al.*, 1991), benchmark dataset classification (El-Yaniv & Gerzon, 2005; Mohan & Kasabov, 2005; Bosnić *et al.*, 2003; Chen, Wang & Dong, 2003; Ho & Wechsler, 2003), text classification (Zelikovitz, 2004; Joachims, 1999), face surveillance (Li & Wechsler, 2004) and gene classification (Pang & Kasabov, 2004). We suggest that this approach can be appropriate for risk analysis in environmental studies where the focus is on a particular location and species rather than on the whole world. This is further investigated in Chapter 6.

2.4 Global, local and ‘personalised’ modelling

Modelling techniques can be divided into three main categories, global modelling, local modelling and ‘individualised’ or ‘personalised’ modelling. Consider a collection of data samples D , a collection of training data samples $D_{tr} \subset D$ and a collection of testing data samples $D_t \subset D$, where $D_{tr} \neq D_t$.

Global modelling results in the creation of one model from all training data samples (D_{tr}). The model is described with only one function (Fig. 2.9a), *e.g.* using a multiple linear regression results in a regression function f_G where $Y = f_G(X)$, $D_{tr} = \{X_1, X_2, \dots, X_k, Y\}$ is a domain data set, X_i are features of D_{tr} , and Y is the vector under estimation. A global model captures trends in data that are valid for the whole problem space and the information included in local patterns may not be discovered. MLP are an example of connectionist systems trained using global learning techniques where the weights between the network’s nodes are adjusted based on the error between the network’s outputs and the known outputs in the training set D_{tr} . The input samples are repeatedly presented to the network until the errors between the predicted and the known outputs are minimized (Bishop, 1995). Learning a new data sample causes MLP to forget all previously learnt patterns. This is a problem when an MLP has to learn new input patterns and/or has to adapt to a change in the modelling environment.

In local modelling a number of local models are created from all available training samples (Fig. 2.9b) where each model represents a cluster of training data, with its own function f_{Li} . These f_{Li} are able to capture local patterns contained in the problem subspaces. The local modelling approach relies on a clustering algorithm to partition data into a certain number of clusters. Samples in a cluster are similar to each other and

dissimilar to samples from other clusters. Euclidean distance is the most commonly used similarity and difference measure (Xu and Wunsch II, 2005):

$$D_{ij} = \sqrt{\sum_{l=1}^d (X_{il} - X_{jl})^2} \quad (2.2)$$

where d is the dimension of the data and X_i and X_j are two samples from the problem space. One of the main characteristics of this modelling is that it does not cause catastrophic forgetting. When a new data sample becomes available only a small part of the problem space is fine-tuned (adapted) to accommodate this new sample.

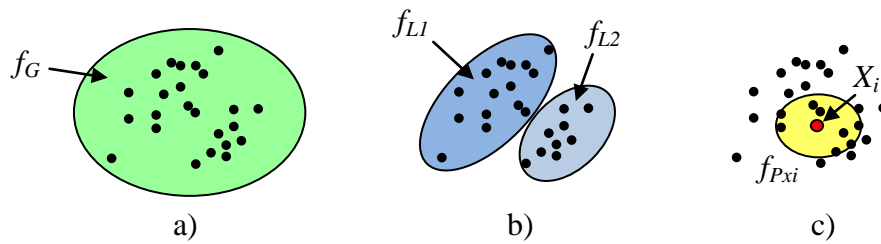


Fig. 2.9. Examples of global (a), local (b) and ‘personalised’ (c) modelling on the same set of data. f_G is a global model, f_{L1} and f_{L2} are local models and f_{Pxi} is a ‘personalised’ model built for X_i .

Global modelling and local modelling are based on inductive reasoning. Personalized models are products of transductive reasoning. A ‘personalised’ model f_{Pxi} is built for every single data sample X_i from D (Fig. 2.9c). Given a data sample X_i from D , a model is created based on the knowledge acquired from the nearest training data samples to X_i . A parameter specifying the number of nearest neighbours must be optimised for the desired accuracy. ‘Personalised’ modelling is particularly useful in cases when not only might a new data sample become available but this new data sample is described by a different set of features. Both global and local modelling typically assumes a fixed set of features. The k-nearest neighbour (KNN) method can be used to create ‘personalised’ models. This algorithm finds the k nearest data samples X_j to data sample X_i (usually using the Euclidean distance measure) and calculates the output value for X_i as the average of the output values of all X_j samples. When a weighted KNN (WKNN) method is used for building the personalized model for X_i , the model is influenced by the distances from the k nearest samples to X_i (Kasabov, 2007b):

$$y_i = \frac{\sum_{j=1}^k w_j y_j}{\sum_{j=1}^k y_j} \quad (2.3)$$

where y_i is the output value for X_i from D_i and w_j are weights based on the distances d between X_i and all k nearest samples to the data sample X_i

$$w_j = \frac{\max(d) - (d_j - \min(d))}{\max(d)} \quad (2.4)$$

Kasabov proposed the weighted-weighted k-nearest neighbour algorithm (WWKNN) where the ‘personalised’ models are calculated not only based on knowledge about neighbouring samples but also based on the importance of each feature in terms of its discriminative power within the neighbourhood of k vectors (Kasabov, 2007b). This approach allows an evaluation of the importance of each feature to the ‘personalised’ knowledge contained in the model f_{Pxi} .

To take advantages of the desirable characteristics of global, local and ‘personalised’ models integrated (hybrid) models could be prepared (Kasabov, 2007a), where the local and ‘personalised’ models discover the local trends in the data and the global model discovers global patterns valid for the whole problem space (Fig. 2.10). The changes caused by the availability of new data samples affect a subset of local models and the global model will need to adapt less often.

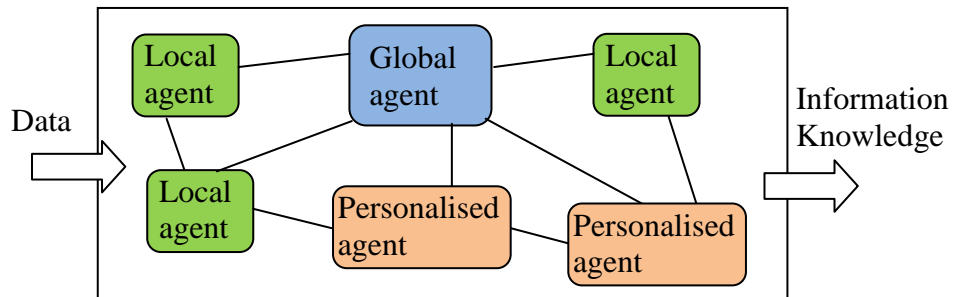


Fig. 2.10. An integrated global, local and ‘personalised’ multi-agent system. Local and ‘personalised’ agents capture local information and knowledge and the global agent captures global trends to provide computational refinements for complex data.

Our experiments show that all three modelling approaches are useful for modelling complex real-world problems. However, integrating them in a single multi-agent system

is a challenging task. We explore and evaluate global, local and ‘personalized’ modelling in more detail in Chapters 4, 5 and 6 by using an ecological case study.

2.5 Feature selection and model validation

2.5.1 Feature selection

The accuracy of connectionist systems is dependent on the quality of the input data. Often the input data contains a limited number of samples with high feature dimensionality. One example is bio-climatic data where there are often a small number of locations where a species is present while each location is described by a high dimensional vector of climate variables (features). Highly dimensional data is a challenge for neural networks. In fact, irrelevant features not only slow down the processing but can also degrade the performance of a learning algorithm. Therefore, feature selection techniques are used as part of data pre-processing to reduce the number of features and more importantly to find the most relevant ones. Features can be selected (1) prior to a model being created (filter techniques) or (2) on the basis of how well the created model performs using the chosen features (wrapper techniques). Because wrapper techniques evaluate the performance of a learning algorithm on various subsets of available features they are more computationally expensive than the filter techniques. However, given a learning algorithm the wrapper techniques can find a better suited subset of features.

Correlation analysis, signal-to-noise ratio analysis (SNR) and principal component analysis (PCA) are very popular feature selection techniques. The most commonly known correlation measure shows the strength of the linear relationship between two features x_i and x_j based on their mean values (μ) and standard deviations (δ):

$$\text{corr}(x_i, x_j) = \frac{\sum_{k=1}^n (x_{ik} - \mu_{x_i})(x_{jk} - \mu_{x_j})}{\delta_{x_i} \delta_{x_j}} \quad (2.5)$$

where n is the number of samples in a dataset D . The value of corr lies between 1 and -1, where $\text{corr}(x_i, x_j) = 1$ and $\text{corr}(x_i, x_j) = -1$ mean complete correlation between x_i and x_j and $\text{corr} = 0$ means x_i and x_j are totally independent features. The features that are highly correlated with other features are removed from D prior to the start of modelling. The challenge is to determine which correlations are too high and consequently which

feature must be removed from the modelling. This technique cannot capture correlations that are not linear in nature.

The SNR method uses a feature's mean and standard deviation values to evaluate the importance of this feature in the modelling dataset:

$$SNR_x = \frac{abs(\mu_{ix} - \mu_{jx})}{\delta_{ix} \delta_{jx}} \quad (2.6)$$

where μ_{ix} and μ_{jx} are the mean values of the feature x for the classes i and j and δ_{ix} and δ_{jx} are the standard deviation values of x for the classes i and j . Fig. 2.11 shows a graphical representation of the SNR ranking of 40 climate variables (features) which describe the presence and absence of a pest insect, the dataset D_{Pest} . The dataset consists of 46 samples, where 27 samples belong to the class 'Pest absent' and 19 to the class 'Pest present'. The analysis was done in a neuro-computing decision support environment called NeuCom (NeuCom, 2008). Features 30, 31 and 32 (the red oval) are ranked the highest. There are two big changes in the SNR values and therefore modelling could start by using the first three highest ranked features and if more features are required, they could be included until an accurate model is prepared.

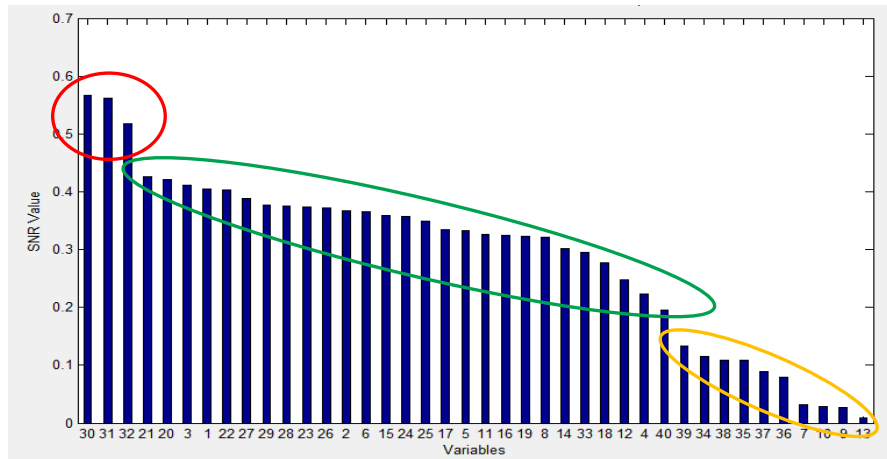


Fig. 2.11. SNR ranking of 40 features of D_{Pest} describing the presence and absence of a pest insect. Features 30, 31 and 32 are the three highest ranked of all 40 features. Two big changes can be identified in the SNR values.

The PCA method transforms the original feature space $X = \{x_1, x_2, \dots, x_n\}$ into an orthogonal space $P = \{p_1, p_2, \dots, p_n\}$ defined with a new set of uncorrelated variables called principal components p_i . Principal components p_i are a linear combination of the

original data features. The first principal component p_1 contains the most important characteristic of the original dataset. The method itself does not reduce the dimensionality of the problem. The reduction is achieved by replacing the original set D by a set of data samples where each sample is characterised by k principal components ($k < n$). However, the method transforms the original data space into a new data space where variables lose their original meaning. Fig 2.12 shows the results of the PCA analysis of D_{Pest} . It can be seen that p_1 accounts for more than 60% of the information present in the original set. When p_2 is used as well, the two features account for $\sim 90\%$ of the information. Other p_i together account for $\sim 10\%$ of the information present in the original set. So it would be sensible to use only p_1 and p_2 .

Fig. 2.13 shows the results of a feature selection task where the GA search algorithm was used for feature selection on D_{Pest} for the ECF method. Features were selected while the model was being built (wrapper method). In this feature selection scheme the feature's importance is measured in terms of the classification performance of the trained model. Here, the best model was created using 25 features shown in the 'Feature Extraction Results' part of the modelling tool. The accuracy of the best ECF model was 91.7%. All together 100 subsets of the original features were created and tested ('Generations' = 100). The major problem with this technique is that the process of finding the best set of features is rather lengthy. It took around 12 minutes to complete the feature selection process on this relatively small dataset. When feature selection is done on larger datasets the process can take hours which might be inappropriate for some modelling tasks.

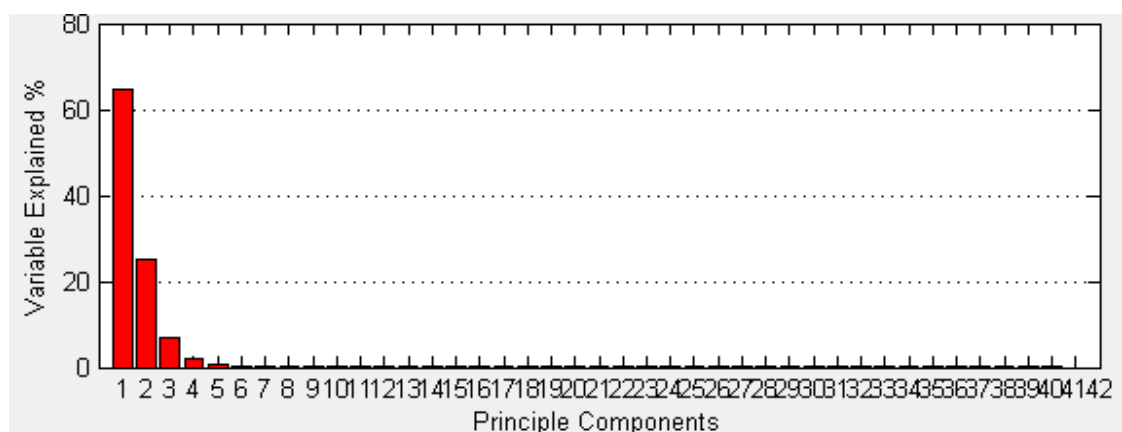


Fig. 2.12. PCA analysis of the 40 features of D_{Pest} the describing presence and absence of a pest insect. The first principal component (p_1) accounts for more than 60% of the information present in the original set.

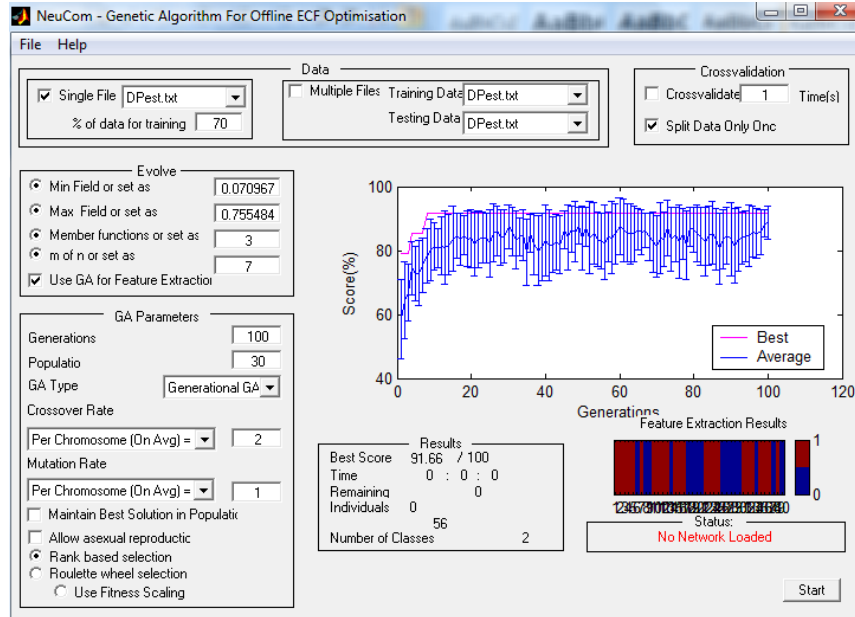


Fig. 2.13. Results of using GA for optimization of an ECF model trained on D_{Pest} . The selected features are shown as red and irrelevant features are shown as blue.

2.5.2 Model validation

Model validation generally implies a measure of the classification or predictive power of a model. Three main approaches exist for this task. The first approach is to use two sets of data, a training set and a test set. For this purpose a single data set D is divided, preferably randomly, into two sets: D_{tr} containing data samples for training a model and D_t containing data samples for testing a model. Ratios of 80%-20% (*i.e.* 80% of all data samples are for training and 20% for testing), 70%-30% and 50%-50% are usually used.

The second approach is to create k subsets of the original dataset D and use $k-1$ subsets for training and one subset for testing. The evaluation is repeated with all possible combinations of $k-1$ subsets and the accuracy is calculated in terms of the mean accuracy over all k experiments. This approach is called k -fold cross validation.

The third approach is called leave-one-out cross-validation. This approach is necessary for smaller datasets D where there are not enough samples to create an independent set for training and testing. If a training set is too small, the resulting model might have unsatisfactory accuracy. During the evaluation, one sample at a time is taken from D and moved to D_t , the rest of the samples are used for training. The process is repeated for every single data sample from D and the accuracy is measured in terms of the mean accuracy over all n experiments, where n is the number of samples in D .

The real question is what modelling accuracy classifies a model as excellent? Could a model having 80% accuracy be considered as a good model or are only models with accuracies above 90% good? We think that a model must be evaluated in the context of its application and it is particularly instructive to compare it to the performance of other models built using the same set of data. Unfortunately, that is often not possible when real-world datasets are used because the access to the data is generally restricted. It is even harder to evaluate the quality of knowledge acquired by a model especially in cases where prior knowledge about the modelled problem is absent or incomplete. In Chapter 4, Section 4.4.1 the knowledge delivered by LPAM is illustrated on the Iris benchmark dataset (Asuncion & Newman, 2007). The data samples in the Iris dataset are defined by four features and examining these attributes in various 2D spaces was not a problem. However, when the number of features increases this analysis becomes cumbersome.

2.6 Conclusion

This chapter presents existing connectionist techniques and techniques for data pre-processing and model evaluation. The most important feature selection techniques are described and their performance on a pest distribution dataset explained. Inductive and transductive reasoning and global, local and ‘personalised’ modelling are presented and contrasted. Special attention is given to the issues associated with MLP networks because they are the most frequently used connectionist systems in ecological modelling. While, they are a well understood and very powerful modelling techniques they have very limited knowledge discovery capabilities. They are also hard to adapt to new data, a characteristic that is becoming increasingly desirable.

This chapter also talks about evolving connectionist systems highlighting the benefits of using them for real-world ecological modelling applications. Two characteristics make ECOS favourable over MLP for modelling changeable ecological processes: (1) the ability to adapt and (2) its knowledge representation capabilities. The first characteristic allows for the building of adaptive models that alter their structure and knowledge to accommodate environmental changes (possibly due to climate change). The second characteristic has the potential of increasing our knowledge of the interactions within and between various ecological systems.

Chapter 3

Ecological modelling, benchmark work and bio-climatic distribution data

This work explores the use of new connectionist structures to deal specifically with problems from biosecurity. Therefore, special attention is given to the modelling techniques in the field of ecological modelling and a survey of the benchmark works from this field is given. In particular, the challenges of assessing the risk of an exotic pest insect invasion using connectionist techniques are addressed. This chapter also acknowledges the specific characteristics of the problems encountered within predictive insect habitat modelling and the frustrations associated with such modelling are highlighted. Finally, the real-world environmental data used in the case studies described in Chapters 5 and 6 is presented. Overall, this chapter gives the support and background information to the contributions presented in Chapter 4, 5 and 6 where existing and novel modelling techniques for risk estimation with case studies from biosecurity are presented.

3.1 Ecological modelling: benchmark works

Over the years, a variety of methods have been used to predict the likelihood of a species' establishment upon its introduction into an area where the species is not normally found. These methods range from those using a graphical approach, where charts of insect distributions versus climatic characteristics are hand plotted and then carefully studied to find favourable and unfavourable climates in the present distribution of an insect (Cook, 1931), to those using on line computer based decision support tools, such as BIOSECURE (Barker *et al.*, 2002) and GARP (Anderson, Lew & Peterson, 2003; Peterson & Vieglais, 2001). In particular, geographic information systems (GIS) provide many facilities to help with the risk assessment process (Cohen, 1998). GIS allow for the storage of data as well as data analysis and data visualization (Monserud & Leemans, 1992). They are suitable for spatial analysis but their predictive modelling capabilities are limited. Therefore, GIS are often used together with a pattern recognition technique in a hybrid system. A GIS and a regression technique have been used to model the habitat requirements of leopard in west and central Asia

(Gavashelishvili & Lukarevskiy, 2008), assess spatial patterns of fern flora in New Zealand (Lehmann, Leathwich & Overton, 2002; Zaniewski, Lehmann & Overton, 2002), estimate the establishment potential of a bacterial disease of potato in Norway (Rafoss, 2003) and forecast distributions of a threatened endemic moth species in Spain (Chefaoui & Lobo, 2008). A GIS has been also combined with an artificial neural network to model the habitat preferences of a sea cucumber (Drumm, Purvis & Zhou, 1999).

In ecological predictive modelling classical pattern recognition techniques have been extensively used on their own. For example, logistic regression (LR) was used to investigate the distribution of British ground beetle species (Eyre *et al.*, 2005) as well as the distributions of families of aquatic invertebrates in Himalayan streams (Manel, Williams & Ormerod, 2001). Generalized adaptive models (GAM) were built to model the distribution of birds in Spain (Suárez-Seoane, Osborne & Alonso, 2002) and to explore the distribution of a variety of flora and fauna species (Pearce & Ferrier, 2000a, 2000b). A fuzzy logic based model of the algal biomass concentration in the eutrophic Taihu Lake in China (Chen & Mynett, 2003) was built to predict the algal biomass concentration in that lake. Furthermore, SOM models were built to examine the worldwide distribution of insects (Gevrey *et al.*, 2006; Céréghino, Giraudel & Compin, 2001). Traditional ANN, particularly those based on MLP, are frequently used for building predictive habitat distribution models. Fish habitats have been modelled using classical statistical methods (Binns & Eiserman, 1979) and ANN (Park *et al.*, 2003; Brosse *et al.*, 1999; Aurelle *et al.*, 1999). ANN based predictive models of bird habitats in two coastal lakes in the USA (Özesmi & Özesmi, 1999), a coastal habitat of New Zealand fur seals (Bradshaw *et al.*, 2002) and a worm in Belgium (Willems *et al.*, 2008) have been built. Climatic envelope models (CEM) based on Mahalanobis distances (Farber & Kadmon, 2003) and ANN (Pearson *et al.*, 2002) were used to assess the influence of climate on species distributions. Artificial neural networks have been particularly popular for modelling aquatic habitats (Li *et al.*, 2007; Olden, Poff & Bledsoe, 2006; Jeong, Kim & Joo, 2006) and the dynamics of pest insects (Zhang & Zhang, 2008; Lippitt *et al.*, 2008; Watts & Worner, 2008).

An excellent review of predictive distribution models in ecology is given in (Guisan and Zimmermann, 2000). The paper has been cited 219 times in the first five years of it being published (Jørgensen, 2005). It reviews modelling techniques used for building predictive models by looking at the techniques' strengths and weaknesses. It also

discusses the issues related to model evaluation, credibility and applicability. This paper also points out the importance of the data sampling strategy to the building of accurate species distribution models.

3.2 Artificial neural networks in the field of ecological modelling

The use of ANN for ecological modelling is not a new concept. In Lek *et al.* (1996), ANN based models are described as a powerful alternative approach to traditional statistical methods for modelling in ecology. In particular, their ability to detect complex nonlinear relationships between independent and dependent variables makes ANN suitable for modelling very complicated and highly nonlinear relationships that characterise ecological models (Lek and Guégan, 1999). Often ecological models based on ANN often outperform the statistical ecological models (Jeong, Kim & Joo, 2006; Paruelo & Tomasel, 1997; Lek *et al.*, 1996).

The most frequently used ANN model is the MLP model. The MLP model has been used for building various prediction models (Sahoo, Ray & Wade, 2005; Lankin *et al.*, 2001; Bowers & Shedrow, 2000), species identification systems (Vaňhara *et al.*, 2007), insects dynamics models (Zhang & Zhang, 2008), habitat models (Willems *et al.*, 2008; Watts & Worner, 2008; Park, Rabinovich & Lek, 2007; Zhang, Gove & Heath, 2005; Bradshaw *et al.*, 2002; Drumm, Purvis & Zhou, 1999; Manel, Dias & Ormerod, 1999; Manel *et al.*, 1999; Özesmi & Özesmi, 1999), and climate change impact models (Krasnopolsky *et al.*, 2008; Pearson *et al.*, 2002). However, the lack of knowledge extraction facilities in an MLP network is a huge disadvantage when this type of network is used for ecological modelling where the understanding of the influence of each independent variable on the dependent variable, *e.g.* the environment on a species, is important (Gevrey, Dimopoulos & Lek, 2003; Bradshaw *et al.*, 2002). In an MLP model the learnt relationships are encoded as weights; a format that cannot be easily interpreted (Kolman & Margaliot, 2005). Furthermore, MLP are static networks and must be retrained every time when new data becomes available. Therefore, a network built for one set of conditions cannot be easily adapted to another set of conditions.

3.2.1 Artificial neural networks in ecological modelling versus other approaches

Over the past two decades, researchers have evaluated and compared the performance of models based on ANN to models using traditional statistical methods. The relationships in ecology are often non-linear and complex and ANN seem to be better suited for the task of modelling in this field than the traditional statistical methods (Watts & Worner, 2008). However, which of those two types of models performs better is still debatable. ANN outperformed the statistical models in the early work carried out by Paruelo and Tomsel (1997) and Lek and his team (1996) and were later confirmed as a powerful ecological data analysis tool by Lencioni *et al.* (2007) and Gevrey *et al.* (2003). In the experiments by Manel, Dias and Ormerod (1999) ANN were compared to discriminant analysis (DA) and logistic regression LR. They found that the accuracies of the ANN (89-100%) were slightly greater than those of the DA (81-95%) and LR (75-92%) models. However, Manel also found and reported in another paper from the same year (Manel *et al.*, 1999) that when the ROC curves were used to assess the performance of the models for predicting species distribution, the LR always outperformed the ANN. Their work highlighted the necessity of finding a suitable and transparent approach for assessing the performances of predictive ecological models that will allow researchers to evaluate their models and communicate their results in an adequate and transparent way. Also, they found that the neural networks are more complex to compute than conventional statistical methods and require a weight analysis to identify possible casual relationships between species distributions and predictor variables. However, in a comparison with a local modelling method called geographical weighted regression (GWR) a MLP feed-forward network was found to be inferior (Zhang, Gove & Heath, 2005). In this case the researchers found that the GWR method builds better models for data collected across a large area. Such data often contains a number of sub-areas with unique relationships between variables that may not be detected and captured by a global model.

The lack of standardised rules for assessing performance of predictive models (Austin, 2007) can make the results of comparison analysis misleading, particularly if the results are not carefully read. A good example is the work done by Williams and Poff (2006), where the performances of ANN, evolutionary algorithms (EA) and classification/regression tree (CART) models were compared. While the ANN were presented with 4-class problems, the four EA models were built for the same problem

with each model built to classify only one 2-class problem. The EA approach was found more accurate than the ANN. In this particular case, the authors pointed out that the results from the two different models are not directly comparable, as it is easier to classify the 2-class data than 4-class data.

3.2.2 Multi-layer perceptron networks for knowledge discovery in ecology

Without any doubt, the most popular artificial neural network in ecology is the multi-layer perceptron feed-forward neural network. One of the main drawbacks of this type of network when used for ecological modelling is its black box nature. Research, such as that carried out by Bradshaw *et al.* (2002), Gevrey *et al.* (2003) and Olden *et al.* (2006), concentrated on the problem of extracting knowledge from trained MLP networks. The ultimate goal of such research is to find the method that will allow an evaluation of influences of each of the predictor variables on the output obtained from a trained MLP network by studying its architecture and weight values. Gevrey *et al.* (2003) lists seven different methods that are considered suitable for analysis of the contribution of each predictor variables. While some of them are better known and used for feature selection purposes in other modelling fields, *i.e.* stepwise selection, some are more unusual. For example, one of the methods uses the products of all weight values between input and output neurons and all hidden neurons as a measure of the importance of the exploratory variables on the calculated output.

3.3 Invasion of exotic pest insects – a case study problem

3.3.1 Climatic-habitat modelling

This thesis' case studies assess the risk of exotic insect species invasions using connectionist techniques. If the invasive potential of an exotic insect can be predicted then its establishment and the subsequent negative consequences might be prevented. Therefore, the most important goal of a predictive model is to give predictions of insects' occurrence at new, unoccupied locations. Ideally, these models should give not only the predictions of an insect's occurrence at unoccupied locations, but also a set of ecologically meaningful explanatory (habitat) variables that capture the relationships between the insect's distribution and the climatic and possible other characteristic of its habitat. Explanatory variables extracted from the insects' current distribution data

would not only greatly improve predictions but also our knowledge of the principles that govern the ecology of insect invasion. Unfortunately, meaningful explanatory variables do not always arise from predictive distribution models.

Studies have been conducted to identify which characteristics make some insect species more invasive than others. For an insect to establish in a new location the conditions for establishment must be fulfilled:

1. The insect has to reach the location preferably, but not exclusively, in superior numbers (Williamson & Fitter, 1996; Worner, 2002).
2. The insect's biological characteristics and the environment of the location being invaded must be favourable to its establishment (Baker, 2002).

Given that the environment is important to the establishment potential of an insect, techniques that compare the characteristics of a species' ecological niches to those in the threatened area have proven popular for building predictive and distribution models (Rafoss, 2003; Robertson *et al.*, 2003; Baker, 2002; Worner, 2002; Manel, Williams & Ormerod, 2001; Peterson & Vieglais, 2001). These types of methods search for the relationships between a species and the habitat in which it occurs. Information about the sampled areas where the species can be found is used to predict how likely the species is to be present or absent in some other (unsampled) area.

Particularly, climate matching techniques that compare the climates in the insects' native habitat with those in the threatened area are often used to predict if an area might be invaded. The climatic characteristics of an area are frequently used as the factors influencing the establishment potential of pest insects (Rafoss, 2003; Denter *et al.*, 2002; Lehmann, Leathwick & Overton, 2002; Peterson & Vieglais, 2002; Cohen, 1998; Cook, 1931). Temperature, relative humidity, soil moisture, and their combined effects are the most important climatic factors (Baker, 2002). The climatic based approach is supported by studies on the impact of climate on the likelihood of an insect's establishment and spread which emphasizes the importance of climatic characteristics of an area for establishment success or failure (Baker, 2002; Dobesberger, 2002; Dobesberger, 2000). Acknowledging that climatic based risk forecasting is very popular, expert software systems, such as CLIMEX, have been designed to explore the relationships between global species distributions and world climates.

In climatic based modelling the estimates of the likelihood of an insect's establishment are solely based on the area's climatic characteristics, making the assumptions that biotic factors, such as the availability of hosts and natural enemies, and intrinsic factors, such as reproductive characteristics and the genetic adaptability of the modelled species, are much less important than the climate. Although research analysing the importance of the host plants to the distribution of species are rare, they do exist (Pilson, 1992).

3.3.2 Field collection of eco-climatic data

Field collection of biological data about species distributions is costly and time-consuming. Often data is collected and analysed but not made available to the wider research community. Therefore, direct comparisons of the performance of two or more models and modelling techniques are very rare (Austin, 2007).

Because the data is hard and expensive to collect, the effects of the size of the collected data on the accuracy of biological predictive distribution models have been studied (Stockwell & Peterson, 2002; Pearce & Ferrier, 2000b). It was found that abundant data does not guarantee high accuracy. Also, there are usually many candidate exploratory variables available (Rushton, Ormerod & Kerby, 2004), but which exploratory variable(s) should be used is hard to decide due to a lack of knowledge about the characteristics of invasive species. Weak relationships between exploratory variables and the presence of the species lead to poor prediction accuracies (Eyre *et al.*, 2005; Pearce & Ferrier, 2000a). To overcome this problem feature extraction techniques, such as PCA and stepwise selection, can be used when there is little known about these relationships (Lencioni *et al.*, 2007; Remm, 2004; Barendregt & Bio, 2003; Chen & Mynett, 2003, Robertson *et al.*, 2003; Céréghino, Giraudel & Compin, 2001).

There are two distinctive types of distribution data. The information about the species' habitat can be collected either only about the locations where the species is known to be present (presence-only data), or about the locations where the species is considered absent and about the locations where the species is known to be present (presence/absence data). It has been suggested that the number of presence and absence data samples should be kept equal because unequal group sizes can influence the accuracy of the model (Fielding & Bell, 1997). Often, the presence data is the only data currently available (Araújo & Williams, 2000).

In 2006, the neural network community recognised the specific characteristics and importance of environmental modelling problems (Cawley *et al.*, 2007; Harva, 2007). In August 2006, two special interest groups, one of INNS and the other of IEEE CIS, were established to encourage the use of computational intelligence technologies in the areas of environmental studies as well as to encourage its own members to develop and apply methods that match the complexity of environmental problems.

3.3.2.1. Noise in insect distribution data

In general, there is a substantial amount of noise in environmental datasets. Furthermore, the noise is often non Gaussian and common techniques for handling noisy data may not be suitable. Unfortunately, how much noise might be in the modelling dataset is not known at the modelling stage and it is extremely hard if not impossible to deduce. For example, consider a presence/absence dataset for an insect with a large number of erroneous or false absences. One location could have been recorded as unsuitable for the insect's establishment because at the time of data collection the insect was not found at this location, while in fact the insect may be absent at the location simply because it has never reached it and therefore has never had the opportunity to establish, rather than the fact that the climate is unsuitable for its establishment. Also, some species are very dynamic and may hide during the sampling period. Clearly these situations introduce prediction errors that cannot be verified (Andreson, Lew & Peterson, 2003).

False absences can significantly affect the accuracy of prediction models. In an experiment, the models predicted the presence of six insect pests for more than 92% of the data of each insect, but the same models correctly predicted absences of the same insects with only 59% to 77% accuracy (Ulrichs & Hopper, 2007). Clearly, the failure to incorrectly identify sites suitable for invasion is more serious than incorrectly predicting extra presences (Fielding, 1999).

3.3.3 Validation of the predictive ecological models

The kappa statistic (Cohen, 1960) has been used for assessing the accuracy of predictive ecological models (Willems *et al.*, 2008; Park, Rabinovich & Lek, 2007; Watts & Worner, 2008; Lütolf, Kienast & Guisan, 2006; Eyre *et al.*, 2005; McPherson, Jetz & Rogers, 2004; Robertson *et al.*, 2003; Farber & Kadmon, 2003; Hirzel & Guisan, 2002; Pearson *et al.*, 2002; Manel, Williams & Ormerod, 2001; Cowley *et al.*, 2000). Cohen's

kappa coefficient (κ) is a statistical measure of the agreement between the predictions and the observed values. This threshold-dependent measure is based on the confusion matrix (Fig. 3.1), where P_T is the number of present sites correctly predicted by the model, A_T is the number of absent sites correctly predicted by the model, while P_F (false negatives – omission error) and A_F (false positives – commission error) are the numbers of incorrect predicted presences and absences, $P_T + P_F + A_T + A_F = n$ and n is the total number of data samples. R_P and R_A are recorded values and P_P and P_A are predicted values.

	R_P	R_A
P_P	P_T	P_F
P_A	A_F	A_T

Fig. 3.1. Confusion matrix. P_T is the number of present sites correctly predicted, A_T is the number of absent sites correctly predicted, P_F is the number of present sites incorrectly predicted as absent and A_F is the numbers of absent sites incorrectly predicted as present.

In prediction models where the output is a continuous value, the P_T , P , A_T and A_F values are obtained by applying a threshold value θ . Typically in ecology, θ equal to 0.5 is used (Fielding and Bell, 1997). If a model generates prediction values within the range $[0, 1]$ and $\theta = 0.5$, then all predictions above 0.5 ($p \geq 0.5$) are designated as presences (P, 1) and all prediction below 0.5 ($p < 0.5$) are absences (A, 0). When the threshold value changes, all values in the confusion matrix change as well, affecting the performance of the prediction model (Fig. 3.2). Lowering the threshold value results in more locations predicted as possible present locations while increasing the threshold value results in fewer locations predicted as possible present locations. Therefore, it is very important to find an optimal threshold value. Unfortunately, the problem of finding the optimal value is not trivial when dealing with streaming new data where a species' presence/absence is not known (Strauss & Biedermann, 2007). This significantly influences the chances of proposing species-habitat relationship models capable of delivering accurate predictions over longer periods of time.

Cohen's kappa values are calculated using (3.1). A kappa value of 1 ($\kappa = 1$) denotes a model with perfect performance, while $\kappa = 0$ indicates a model with a very poor discrimination power. The model's performance can be ranked using Table 3.1 (Monserud & Leemans, 1992; Landis & Koch, 1977). For example, a κ value of 0.4 or

greater is associated with a model showing ‘Fair’ performance. For a ‘Good’ performing prediction model a κ value of 0.55 or greater is required.

$$\kappa = \frac{\frac{P_T + A_T}{n} - \frac{(P_T + P_F)(P_T + A_F) + (A_T + A_F)(A_T + P_F)}{n^2}}{1 - \frac{(P_T + P_F)(P_T + A_F) + (A_T + A_F)(A_T + P_F)}{n^2}} \quad (3.1)$$

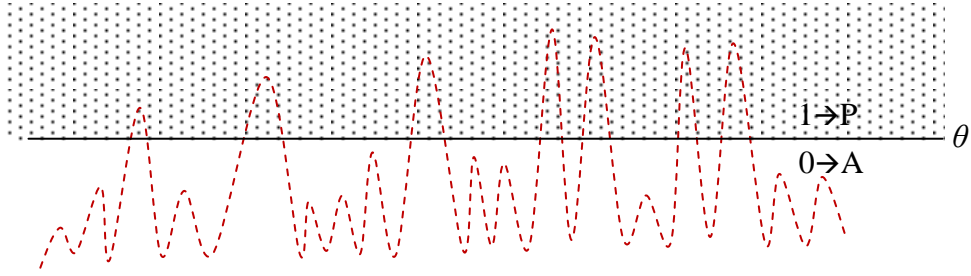


Fig. 3.2. Given a threshold value θ , predictions greater than θ represent the species presence (P), otherwise the species is predicted to be absent (A). When θ changes the numbers of locations predicted as present and absent change as well. Smaller threshold values result in more locations predicted as present while increasing the threshold value results in fewer present locations.

The confusion matrix is used to calculate the model’s sensitivity (*sen*) and specificity (*spe*) using (3.2). Sensitivity and specificity are measures of a model correctly being able to predict true presences and true absences, respectively. The overall accuracy of the model (*a*) is given by (3.3).

$$sen = \frac{P_T}{P_T + A_F} \quad spe = \frac{A_T}{P_F + A_T} \quad (3.2)$$

$$a = \frac{P_T + A_T}{n} \quad (3.3)$$

Relative operating characteristic (ROC) curves (Metz, 1978) have also been used for assessing the performance of predictive biological models (Gavashelishvili & Lukarevskiy, 2008; La Morgia, Bona & Badino, 2008; García *et al.*, 2007; Gibson *et al.*, 2004; Schadt *et al.*, 2002; Suárez-Seoane, Osborne & Alonso, 2002; Osborne, Alonso & Bryant, 2001). ROC curves are based on the confusion matrix as well. A ROC curve is a plot of a model’s sensitivity (*sen*) against its specificity (*spe*) across a range of threshold values. Fig. 3.3 shows a typical conventional ROC curve as given in (Metz, 1978). Given a model, the larger the area under the curve, the more accurate the model.

TABLE 3.1

The level of agreement between observed and predicted values
(Monserud & Leemans, 1992)

κ	Agreement
< 0.05	No agreement
0.05 – 0.20	Very poor
0.20 – 0.40	Poor
0.40 – 0.55	Fair
0.55 – 0.70	Good
0.70 – 0.85	Very good
0.85 – 0.99	Excellent
> 0.99	Perfect

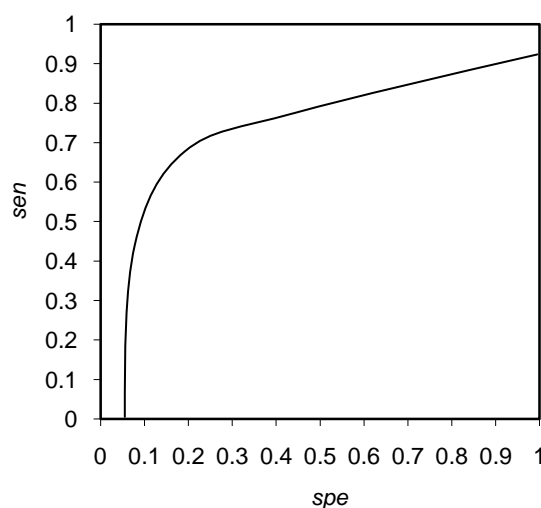


Fig. 3.3. A typical conventional ROC curve (Metz, 1978). The area under the ROC curve is a measure of the model's predictive power.

3.4 Real-world bio-climatic pest distribution data and its preliminary analysis

In this thesis a meteorological dataset introduced in (Peacock, 2005) was used. This dataset is comprised of the climatic characteristics of 6458 worldwide sites and the presence and absence of 36 pest insects at each of those sites. Fifteen of the species from the dataset are present in New Zealand and twenty-one are not. Each site (a sample location) is characterised by forty climate variables (Table 3.2) and is given a class label indicating insect distribution, *i.e.* insect present (1) or insect absent (0). All pest insect

data was very sparse. There are many more locations where the insects are considered absent than the locations where the pest insects have been found. The climate variables listed in Table 3.2 describe the temperature, rainfall and soil moisture characteristics of the sampled sites. One of the main characteristics of the dataset is that each insect has been recorded absent at significantly more sites than it has been recorded as present at. This may significantly influence the accuracy of the predictive model, when estimating the true absence of an insect. Therefore, in all experiments the number of absent sites was kept roughly equal to the number of present sites by randomly selecting absent sites from the pool of available absent data. Furthermore, it is likely that the absent data is very noisy while the present data is not. It is important to notice that the data does not contain any knowledge about the significance of any of the forty climate variables on the establishment potential of any of the 36 species.

The number of sampled sites differs from pest to pest. Table 3.3 shows the number of samples for four pest insects used in this work. While *Planococcus citri* (*P. citri*) is recorded as being present in NZ the other three species are not, however they are often intercepted at New Zealand borders (S. P. Worner, personal communication, 2003) and therefore impose a threat to the New Zealand economy. In New Zealand *A. hartii*, *G. coffeae* and *X. perforans* have the status of an unwanted insect pest, as they have been recognised as pests of potential economic importance (NZ Ministry of Agriculture and Forestry). To investigate possible data constraints we performed the following explanatory analysis:

- Step 1: To avoid the influence of the large number of absences on the accuracy of the model, the number of locations where the pest is recorded absent was kept equal to the number of locations where the pest insect is recorded as present.
- Step 2: To identify possible geographical outliers, the spatial distributions of the locations of the newly formed datasets were checked. For example, Fig. 3.4 shows the spatial distribution of the pest insect *P. citri*'s presence. This data set includes a number of island locations, which might appear in Fig. 3.4 as geographical errors.
- Step 3: The insects' responses to the most influential explanatory variables were investigated. For this purpose (i) the climate space of each predictor variable was divided into ten equally spaced ranges, and (ii) the climate distributions of the insects in those ranges were assessed.

TABLE 3.2

Climate variables describing the climate of each site.

Variable	Description
T_{Smax}	Maximum summer temp ($^{\circ}C$)
T_{Wmin}	Minimum winter temp ($^{\circ}C$)
DD_5	Degree days above $5^{\circ}C$ (days)
DD_{15}	Degree days above $15^{\circ}C$ (days)
T_{Sum1}	Temperature – 1 st month of summer ($^{\circ}C$)
T_{Sum2}	Temperature – 2 nd month of summer ($^{\circ}C$)
T_{Sum3}	Temperature – 3 rd month of summer ($^{\circ}C$)
T_{Aut1}	Temperature – 1 st month of autumn ($^{\circ}C$)
T_{Aut2}	Temperature – 2 nd month of autumn ($^{\circ}C$)
T_{Aut3}	Temperature – 3 rd month of autumn ($^{\circ}C$)
T_{Win1}	Temperature – 1 st month of winter ($^{\circ}C$)
T_{Win2}	Temperature – 2 nd month of winter ($^{\circ}C$)
T_{Win3}	Temperature – 3 rd month of winter ($^{\circ}C$)
T_{Spr1}	Temperature – 1 st month of spring ($^{\circ}C$)
T_{Spr2}	Temperature – 2 nd month of spring ($^{\circ}C$)
T_{Spr3}	Temperature – 3 rd month of spring ($^{\circ}C$)
R_{mean}	Mean total rainfall (mm)
R_{Sum1}	Rainfall – 1 st month of summer (mm)
R_{Sum2}	Rainfall – 2 nd month of summer (mm)
R_{Sum3}	Rainfall – 3 rd month of summer (mm)
R_{Aut1}	Rainfall – 1 st month of autumn (mm)
R_{Aut2}	Rainfall – 2 nd month of autumn (mm)
R_{Aut3}	Rainfall – 3 rd month of autumn (mm)
R_{Win1}	Rainfall – 1 st month of winter (mm)
R_{Win2}	Rainfall – 2 nd month of winter (mm)
R_{Win3}	Rainfall – 3 rd month of winter (mm)
R_{Spr1}	Rainfall – 1 st month of spring (mm)
R_{Spr2}	Rainfall – 2 nd month of spring (mm)
R_{Spr3}	Rainfall – 3 rd month of spring (mm)
APE	Annual potential evapotranspiration (mm)
AAE	Annual actual evapotranspiration (mm)
MI	Moisture index (soil moisture) (mm)
MI_{300}	Soil moisture index at 300mm soil depth (mm)
AMI_{50d}	Annual soil moisture deficit at 50mm (mm)
AMI_{50s}	Annual soil moisture surplus at 50mm (mm)
AMI_{150d}	Annual soil moisture deficit at 150mm (mm)
AMI_{300d}	Annual soil moisture deficit at 300mm (mm)
AMI_{300s}	Annual soil moisture surplus at 300mm (mm)
AMI_{700d}	Annual soil moisture deficit at 700mm (mm)
AMI_{700s}	Annual soil moisture surplus at 700mm (mm)

TABLE 3.3

The number of samples for four pest insects used in this thesis.

PEST	<i>Planoccus</i>	<i>Aspidiella</i>	<i>Geococcus</i>	<i>Xyleborus</i>
	<i>citri</i>	<i>hartii</i>	<i>coffea</i>	<i>perforans</i>
Samples	454	78	75	191
Present	223	36	38	92
Absent	231	42	37	99

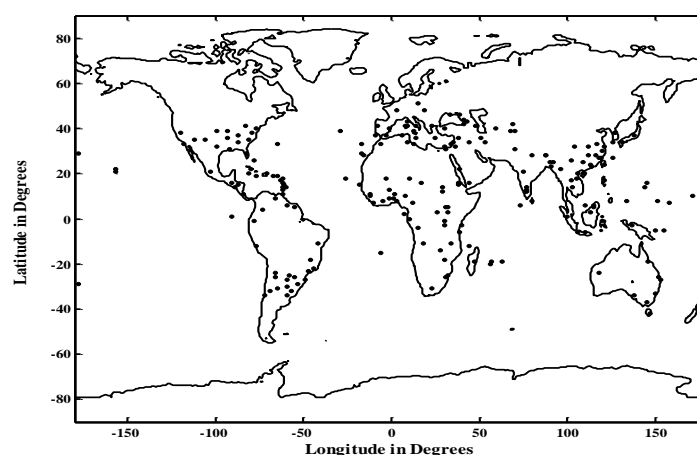


Fig. 3.4. Spatial distribution of the insect *P. citri*'s presence. Please note that a number of island locations might appear as geographical errors.

Fig. 3.5 shows the distribution of the insect *P. citri*. Each bar on the histogram in Fig. 3.5 shows the total number of sampled locations (n) for the particular T_{Smax} range. The black portion of each bar represents the number of locations the species is present in (n_p) and the gray portion represent the number of locations where the species is considered absent (n_a). The ratio r between sites where the species is present with respect to all locations within each range is written on the top of each bar, *i.e.* $r = n_p \div n$ expressed as %. It can be seen that the value of r increases with an increase in T_{Smax} . This indicates that summer temperature may be one of the dominant factors influencing this insect's distributions.

Step 4: Correlation analysis on the explanatory variables was carried out for each species. The results of the analyses, together with an expert's knowledge about the species' dynamic behaviour were used to decide which climatic attributes of their habitats were to be used as predictor variables.

However, it must be noted that the real factors regulating the distributions of the studied insects are unknown.

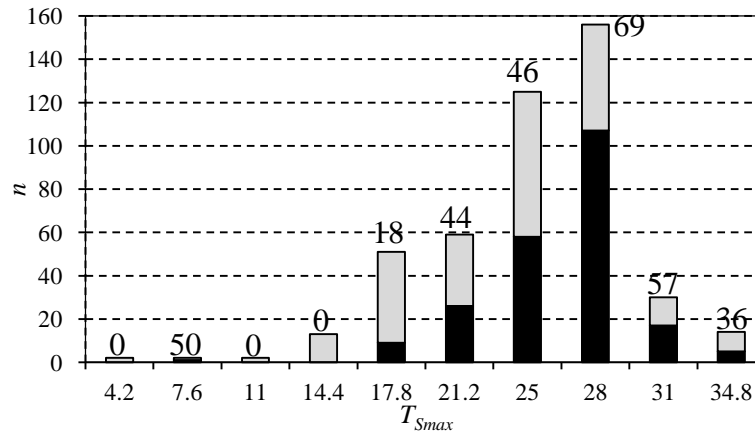


Fig. 3.5. Distribution of the insect *P. citri* over a range of maximum summer temperature (T_{Smax} , °C). Each bar represents the distribution of all locations (n), where the dark areas represent the number of plots occupied by the insect (n_p). The number written on top of each bar shows the percentage of all sites occupied by the insect ($r = n_p \div n$ %).

3.5 Conclusion

In this chapter, special attention is given to the modelling techniques and problems in the field of ecological modelling. By far the most utilized neural network in ecology is MLP. Despite the disadvantages associated with MLP networks and the fact that a number of new and improved connectionist models have been designed and applied in other research areas, new models are rarely considered as possible tools in ecology.

Much of this chapter focuses on building predictive habitat distribution models for the assessment of the risk of exotic pest insect invasions. In general, building predictive models of an insect establishment potential is a difficult task due to the complexity of the problem and availability of data. Namely, the models should give the prediction of the insect's occurrence at new unoccupied locations, as well as deliver a set of ecologically meaningful explanatory variables that capture the relationship between the insect establishment and the characteristics of its habitat. Clearly, MLP networks are not the ideal models for this task. While they can give very accurate predictions their exploratory capability is minimal.

A frustration associated with modelling predictive distribution models is that the accuracy of such models is greatly influenced by the quality of the modelling data. Sampling data about an insect's habitat is a costly and time consuming task, often

resulting in a noisy and/or sparse dataset. Particularly, insect distribution data may contain a large amount of noise in the absence data and much less or no noise in the presence data. As a result, the accuracy of predicting absences may be much lower than the accuracy predicting presences. Unfortunately, quality benchmark datasets readily available to researchers for testing their approaches are rare. As a result, comparative studies of models are uncommon.

Climate matching techniques that use data on climate and species distributions are increasingly used for modelling insect habitats. The importance of climate on the invasive characteristics of insects has been recognised. However, the information about what makes an insect invasive is lacking, making the process of assessing the true accuracy of predictive habitat models difficult.

We conclude that using dynamic habitat models in ecology is a new concept, as the vast majority of models currently used are static. Static models cannot operate in dynamic environments and improve their performance over the time and therefore they cannot be easily transferred into a climatically unpredictable future environment. Furthermore, MLP-based models suffer from the black-box syndrome and these models cannot easily provide an interpretation of the effects of individual predictor variables on the species establishment. In the next chapter, an adaptive prediction model tackling the above issues is presented.

Chapter 4

Local probability based adaptive model (LPAM) for risk evaluation and knowledge discovery

As stated earlier, the vast majority of predictive distribution models are static. As a result, these models cannot operate in dynamic environments and they cannot improve their performance over time. In this chapter, an adaptive model based on local probability named the local probability adaptive model (LPAM) for risk prediction is presented. LPAM is based on ECM and DENFIS and contains a local probability module. This dynamic model delivers knowledge about the contributions of each predictor variable to the response variable. In other words, the knowledge extracted from LPAM explains the influence of each predictor variable on the modelled output.

First, the LPAM building blocks are explained. Then, we propose a method used for the visualization of the extracted knowledge. The model has been applied on two benchmark problems and the results of those experiments are given at the end of this chapter. Finally, the features of the proposed model are summarised. LPAM was first published in (Soltic *et al.*, 2003) and further evaluated in (Soltic *et al.*, 2004a & 2004b).

4.1 Introduction

The black-box nature of MLP networks is without a doubt an important issue when these networks are used for building predictive models and eco-climatic knowledge discovery. Novel approaches that can overcome this issue are required. The proposed model, named LPAM, has many desirable features that are unavailable in MLP networks:

1. An LPAM has a knowledge extraction facility delivering the knowledge learnt by the networks in the form of IF-THEN rules. When a LPAM is used as an insect distribution model the extracted knowledge explains how habitat characteristics influence the insect's distributions.
2. An LPAM facilitates incremental learning. It can learn new data without the need to be retrained on the old data samples. This characteristic makes the

LPAM suitable for on-line applications and for applications where a complete set of training data is not given in advance and some new training data samples might become available in future.

3. An LPAM is based on local probability and therefore is a logical approach for modelling the habitat of invasive insects over smaller geographical areas rather than on a global scale. Thus, the information about insect locations is better conserved in the estimations. A LPAM builds a set of adaptive local models through the clustering of data samples and by employing inductive reasoning. Each model represents a cluster of locations with similar characteristics.

An LPAM includes three modules or components (Fig. 4.1):

1. A clustering module,
2. A probability evaluation module, and
3. An inference module.

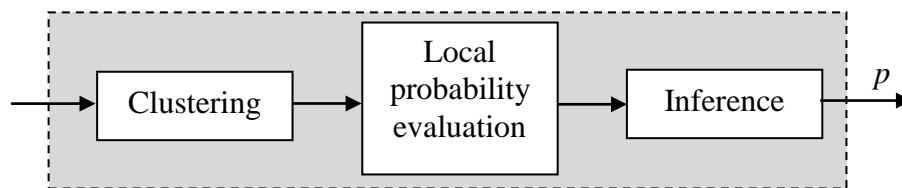


Fig. 4.1. Block diagram of an LPAM. The clustering and inference stages are based on ECM and DENFIS, respectively. The local probability evaluation stage lowers the model's sensitivity to noise.

The clustering and inference modules employ ECM and DENFIS, respectively. As a result, LPAM are able to operate both on-line and off-line. The clustering and probability evaluation modules make the predictive model less sensitive to noise. Together they minimize the influence of the noisy data on the performance of the model. They also transform the binary input vectors representing locations into vectors with continuous output (cluster probabilities). The output of the LPAM is confined to values between 0 and 1, therefore a probability threshold (θ) must be used to dichotomise these output values into either 0 or 1. The output values $p \geq \theta$ are designated as Class 1 and $p < \theta$ are Class 0. When LPAM are used for building distribution models these continuous values represent the distributions at various locations with similar climate characteristics. The cluster probabilities are used to train the inference stage and consequently to obtain the establishment potential predictions at

test locations as well as to form the IF-THEN rules explaining what the model has learnt. We show later that the extracted rules can be used to determine the influence of each predictor variable on the distribution of the pest in a specific area defined by climatic variables. The end result is a flexible model with accurate predictive performance and the ability to identify the species-habitat relationships (Soltic *et al.*, 2004a & 2004b; Soltic *et al.*, 2003). The building blocks of LPAM and their role in the proposed approach are explained in the next section.

4.2 The proposed model

4.2.1 Problem definition

Given a domain data set: $D = \{S_1, S_2, \dots, S_n, Y\}$ where $S_i(i=1, \dots, n)$ are data samples of D and $Y = y_1, y_2, \dots, y_n$ is the vector under estimation. Suppose $S = x_1, x_2, \dots, x_l$. The target is to predict Y in terms of S by modelling an estimation function f , where $Y = f(S)$. When used to model the risk of insect invasions the model fits response surfaces as a function of predictors in the environmental space $E = \{S_1, S_2, \dots, S_n\}$ and then uses the spatial pattern predictor surfaces to predict the responses in the geographical space $G = \{g_1, g_2, \dots, g_k\}$, where the data samples are of type $g_i = (\text{latitude}_i, \text{longitude}_i)$.

4.2.2 ECM Clustering

The clustering module was introduced to allow the LPAM to deal with noisy data sets. The data samples are clustered and each cluster is given a local probability value p_i based on the ratio of class one samples to class two samples within the cluster. The eco-climatic data for an insect may contain many false-absence samples. Examples of false-absences include sites where the insect was not found and therefore the location is considered unsuitable for its establishment when in reality the insect did not have a chance to establish at these locations due to other factors. Replacing class labels with p_i values lowers the influence of the incorrect data on the training of the LPAM.

The clustering stage utilizes an evolving clustering module called ECM. ECM is a fast, one-pass algorithm for dynamic clustering of input stream data, where there is no prior knowledge about the optimum number of clusters (Kasabov, 2002; Kasabov & Song, 2002). Each cluster is described by a centre C_{cj} and radius R_{cj} . This algorithm is a distance-based clustering method where the cluster centres (called prototypes) are determined on-line such that the maximum distance d_{max} between an input sample s_i and

the closest prototype(s) cannot be larger than some threshold value D_{thr} . Therefore, the number of clusters created with this algorithm depends on the characteristics of data samples being clustered. However, the number of created clusters is user-controlled by a suitable value for D_{thr} . Furthermore, D_{thr} can be adjusted during the on-line clustering process (Kasabov, 2002). This algorithm ensures that the distance from each cluster centre to the furthest sample in the cluster is less than D_{thr} . ECM partitions a data set D into ζ clusters, where $\zeta > 1$:

$$D = \bigcup_{i=1}^n s_i = C_1, C_2, \dots, C_\zeta \quad (4.1)$$

The ECM algorithm is very straight forward. One sample at a time is taken from the training data set and its similarity, measured as the normalized Euclidean distance d_{ij} , to the already formed cluster centres C_c is evaluated.

Let X_i be the nearest sample to C_{cm} . If the distance d_{im} is less than this cluster's R_{cm} (i.e., $d_{im} < R_{cm}$) X_i belongs to this cluster and R_{cm} and C_{cm} stay unchanged (Fig. 4.2.a). However, if d_{im} is greater than $2 \times D_{thr}$ (i.e., $d_{im} > 2 \times D_{thr}$) X_i does not belong to any cluster and a new cluster is created with the centre $C_{ci} = X_i$ and radius $R_{ci} = 0$ (Fig. 4.2.b). Otherwise (i.e., $d_{im} \leq 2 \times D_{thr}$), the sample belongs to the cluster and the cluster's R_{cm} and C_{cm} are updated (Fig. 4.2.c). The centre C_{cmNew} is moved along the line connecting X_i and C_{cmOld} so that $d_{imNew} = R_{cmNew}$:

$$R_{cmNew} = \frac{d_{imOld} + R_{cmOld}}{2} \quad (4.2)$$

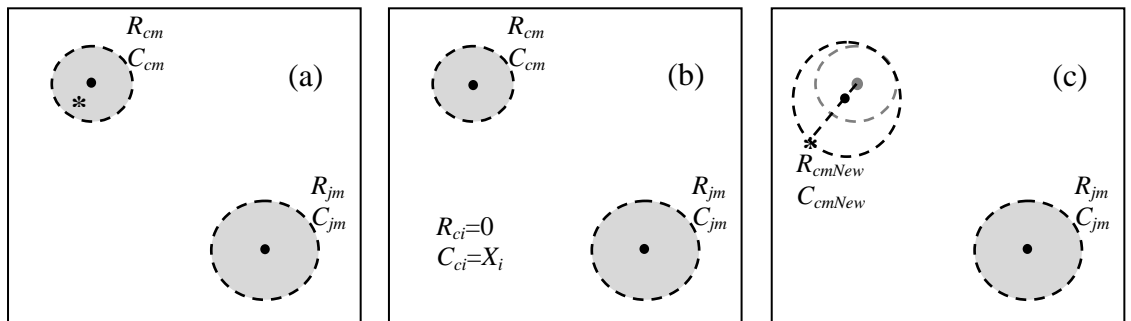


Fig. 4.2. ECM clustering (Kasabov & Song, 2002). The current sample X_i is represented by ‘*’. (a) $d_{im} < R_{cm}$, neither the cluster (C_{cm}, R_{cm}) is updated nor a new cluster is created, (b) $d_{im} > 2D_{thr}$, a new cluster is created ($C_{ci} = X_i$ and $R_{ci} = 0$), (c) $d_{im} \leq 2D_{thr}$, X_i belongs to the C_m cluster and C_{cm} and R_{cm} are updated.

4.2.3 Local probability evaluation

The clusters found using ECM are further analysed with a probability evaluation unit. If $\{C_1, C_2, \dots, C_\zeta\}$ are the clusters from the clustering module. For each cluster $C_i \in \{C_1, C_2, \dots, C_\zeta\}$ the mean vector of predictor variables (features) is calculated using:

$$M_i^c = \frac{\sum_{j=1}^{|C_i|} S_j}{|C_i|} \quad (4.3)$$

where $i = 1, \dots, \zeta$ and $S_j = x_{j1}, x_{j2}, \dots, x_{jl}$ are features of the X_j sample.

The local probability for the samples from the cluster C_i is given by:

$$p_i^c(y|x_1, x_2, \dots, x_l) = \frac{\sum_{j=1}^{|C_i|} \sum_{a=1}^m \prod_{b=1}^l p(y_a|x_b)}{|C_i|} \quad (4.4)$$

According to Bayesian theory (Neter, Wasserman & Kutner, 1990) $p(y_a|x_1, \dots, x_l) = \prod p(y_a|x_b)$ and therefore (4.4) can be reformulated as

$$p_i^c(y|x_1, x_2, \dots, x_l) = \frac{\sum_{j=1}^{|C_i|} \sum_{a=1}^m p(y_a|x_1, x_2, \dots, x_l)}{|C_i|} \quad (4.5)$$

In the special case when $m = 1$, (4.5) can be further simplified:

$$p_i^c(y|x_1, x_2, \dots, x_l) = \frac{\sum_{j=1}^{|C_i|} p(y_a|x_1, x_2, \dots, x_l)}{|C_i|} \quad (4.6)$$

Finally, each cluster $C_i \in \{C_1, C_2, \dots, C_\zeta\}$ is represented by a pair (M_i^c, p_i^c) where M_i^c and p_i^c are obtained from (4.3) and (4.6). A new dataset $D_M = \{M_1^c, M_2^c, \dots, M_n^c, P^c\}$ is used to train the inference module.

The probability evaluation module reduces the influence of noisy data on the prediction accuracy of the LPAM. The clusters containing only the locations where the pest is present have $p_i^c = 1$. The clusters without any location where the pest is present have $p_i^c = 0$. However, most clusters are made up of various numbers of pest present and pest absent locations. More present than absent locations result in a greater p_i^c ($p_i^c > 0.5$) and

vice versa. Calculating the establishment potential for a whole cluster ensures that small variations of data do not affect the result.

4.2.4 Knowledge discovery in DENFIS

DENFIS is a fuzzy inference system introduced by Kasabov and Song (2002). This system employs the first-order Takagi-Sugano fuzzy rules (Kasabov, 2002). DENFIS is an ECOS model (watts, 2009). Its structure evolves through on-line, incremental, hybrid (unsupervised and supervised) learning. DENFIS accommodates new input data, features and classes through local element tuning and calculates the output through a dynamic fuzzy inference. New fuzzy rules are created and updated during the learning process. The rules can be extracted at any time during (and after) the learning stage. This feature is very desirable for predictive ecological modelling because it delivers an explanation of how a species' population is influenced by climate variables in the form of easy to read IF-THEN rules.

DENFIS has been used on real-world financial data (Huang, Pasquier and Quek, 2008), for time series modelling and prediction (Kasabov, 2002), for software reliability estimation (Kiran & Ravi, 2007), software development cost estimation (Kumar *et al.*, 2008) and for the prediction of a primary transformer's waveform (Kasabov, Venkov & Minchev, 2003).

One of the disadvantages of DENFIS is that it requires larger data sets than other networks, such as the MLP network. Also, error rates depend on the characteristics of the data samples. Namely, during training, DENFIS partitions the problem space into a number of regions. If test data does not fall into those regions the system produces lower prediction accuracy. Despite its disadvantages DENFIS is a particularly promising tool for building predictive ecological models because it can deliver fuzzy rules created during the training stage:

$$\begin{aligned}
 &\text{IF } x_1 = R_{11} \text{ AND } x_2 = R_{12} \text{ AND } \dots \text{ AND } x_l = R_{1l} \\
 &\quad \text{THEN } y = f_1(x_1, x_2, \dots, x_l) \\
 &\quad \dots \\
 &\text{IF } x_1 = R_{m1} \text{ AND } x_2 = R_{m2} \text{ AND } \dots \text{ AND } x_l = m_{1l} \\
 &\quad \text{THEN } y = f_m(x_1, x_2, \dots, x_l)
 \end{aligned}$$

where f_i are linear functions in the form $y = \beta_0 + \beta_1 x_1 + \dots + \beta_l x_l$ and m is the number of local models. In Section 4.3 these rules are detailed and their graphical representation proposed.

4.2.5 Model implementation

The proposed model can be implemented according to the following steps:

Step 1: Partition training data from D into ξ clusters using ECM.

Step 2: Given that $\{C_1, C_2, \dots, C_\xi\}$ are clusters from the clustering module. Represent each cluster by a pair (M_i^c, p_i^c) . The new dataset $D_M = \{M_1^c, M_2^c, \dots, M_\xi^c, P^c\}$ is then used to train the inference module.

Step 3: Perform regression in D_M . Performing regression in D_M allows the model to estimate probability without losing the key information in the clusters.

Step 4: Risk estimation in the D domain.

Step 5: Predict the response in the geographical space G . This step is optional. It is used when a LPAM models species distributions. For this purpose, the predictive maps are drawn to visualise possible hotspots. In this work, the map contours were created using the biharmonic spline interpolation method with 1° grid spacing (MATLAB).

The MATLAB code for the LPAM is given in Appendix A.

4.3 Local knowledge extraction with the use of LPAM and its visualization

As stated earlier, LPAM deliver knowledge in the form of IF-THEN rules. Consider an LPAM model with three predictor variables (x_1, x_2, x_3) and two extracted rules (Rule 1, Rule 2) as shown in Table 4.1. The proposed visualization is given in Fig. 4.3.

Rule 1 states that

IF the value of x_1 is in the area defined by a Gaussian function with the mean value c_{11} and the standard deviation σ_{11}

AND the value of x_2 is in the area defined by a Gaussian function with the mean value c_{12} and the standard deviation σ_{12}

AND the value of x_3 is in the area defined by a Gaussian function with the mean value c_{13} and the standard deviation σ_{13}

THEN the response is calculated by the following formula

$$y = a_{10} + a_{11} x_1 + a_{12} x_2 + a_{13} x_3,$$

where a_{11} , a_{12} and a_{13} are regression coefficients in Rule 1.

Rule 2 is characterised with its own set of Gaussian functions and regression lines and states that

IF the value of x_1 is in the area defined by a Gaussian function with the mean value c_{21} and the standard deviation σ_{21}

AND the value of x_2 is in the area defined by a Gaussian function with the mean value c_{22} and the standard deviation σ_{22}

AND the value of x_3 is in the area defined by a Gaussian function with the mean value c_{23} and the standard deviation σ_{23}

THEN the response is calculated by the following formula

$$y = a_{20} + a_{21} x_1 + a_{22} x_2 + a_{23} x_3,$$

where a_{21} , a_{22} and a_{23} are regression coefficients in Rule 2.

In all the extracted rules, a_{i0} is the predicted value of y when all variables x_i are equal to zero.

TABLE 4.1

Example of two rules extracted by the LPAM

	c	σ	$y = a_{r0} + a_{r1} x_1 + a_{r2} x_2 + a_{r3} x_3$
Rule 1:			
x_1	c_{11}	σ_{11}	$a_{10} + a_{11} x_1 + a_{12} x_2 + a_{13} x_3$
x_2	c_{12}	σ_{12}	
x_3	c_{13}	σ_{13}	
Rule 2:			
x_1	c_{21}	σ_{21}	$a_{20} + a_{21} x_1 + a_{22} x_2 + a_{23} x_3$
x_2	c_{22}	σ_{22}	
x_3	c_{23}	σ_{23}	

We propose here that the centres of the Gaussian curves and coefficients of the regression lines can be used to explain the influence of each predictor variable on the output. Fig. 4.3 shows the visualization of the rules from Table 4.1. The red circles illustrate the Rule 1 contributions and the blue circles illustrate the Rule 2 contributions. In this example Rule 1 describes the data samples with high predictor values. The contributions of x_1 and x_3 are positive and a decrease in x_2 causes an increase in the response. The main contributor is x_3 ($a_{13} > a_{11} > |a_{12}|$). Rule 2 describes the data samples with low and moderate predictor values. All regression coefficients are positive and the main contributor is x_1 ($a_{21} > a_{22} > a_{23}$).

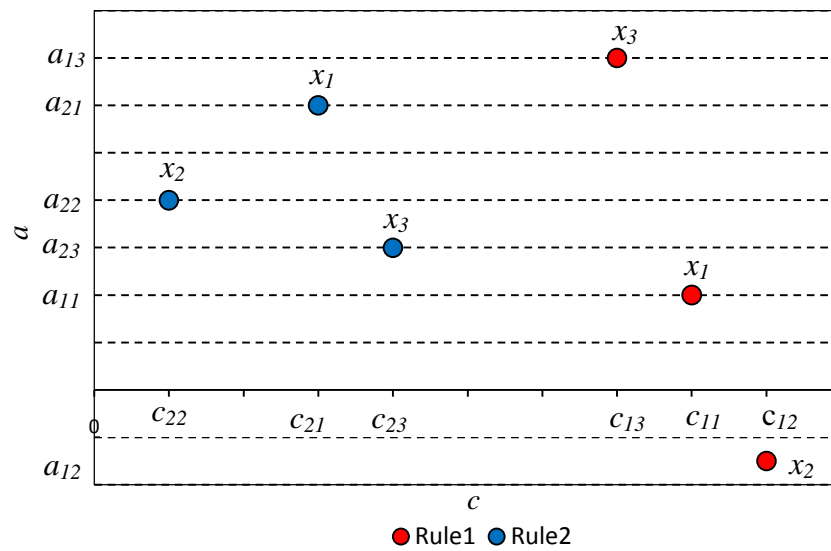


Fig. 4.3. Contributions of predictor variables in two fuzzy rules. Each rule explains the influence of three variables, x_1 , x_2 and x_3 . Rule 1: higher $x_2 \Rightarrow$ lower response (a_{12} is negative), the main contributor is x_3 (a_{13}). Rule 2: all positive contributors, the main contributor is x_1 (a_{21}).

4.4 Validation of the LPAM using benchmark datasets

The proposed model was initially validated using two very popular benchmark datasets from the UCI Machine Learning Repository database, *i.e.* the iris plant dataset and the wine recognition dataset (Asuncion & Newman, 2007). Both datasets contain 3 classes, where each class, C_i , refers to a type of iris plant or a type of wine, respectively. The LPAM is designed to be used on two-class problems and therefore three subsets of the original datasets were prepared, where Subset 1 comprised the C_0 and C_1 samples, Subset 2 contained C_1 and C_2 samples and the C_1 and C_2 samples were in Subset 3. In all experiments the subsets were shuffled before leave-one-out cross validation was performed:

- Step 1: Let $i = 1$.
- Step 2: Remove data sample X_i from the original dataset D .
- Step 3: Train on reduced dataset.
- Step 4: Evaluate the performance using X_i .
- Step 5: Return X_i back into the original dataset D .
- Step 6: Let $i = i + 1$
- Step 7: IF there are more samples THEN repeat Steps 2 to 6 ELSE
- Step 8: Finish the modelling.

Because the output of the LPAM is confined to values between 0 and 1, a probability threshold (θ) must be used to dichotomise these output values into either 0 (Class 0) or 1 (Class 1). As stated earlier, the output values below θ are assigned a '0' and the output values above θ are assigned a '1'. Here, the results for $\theta = 0.5$ are given, although a better accuracy might be achieved at some other, usually higher, threshold θ .

4.4.1 Validation on the iris dataset

The iris dataset is one of the most popular benchmark datasets. The set contains 3 classes of 50 samples each, where Class 0 refers to the iris Setosa flower, Class 1 refers to the iris Versicolour flower and Class 2 refers to the iris Virginica flower. Each sample is characterised by 4 features: sepal length (S_l), sepal width (S_w), petal length (P_l) and petal width (P_w), all given in cm, and it is given one of three class labels (0, 1 or 2). Table 4.2 and Fig. 4.4 detail the dataset's characteristics showing average values (μ) and standard deviations (σ) of the features in this dataset. While the values of S_l and S_w are similar in all 3 flowers, iris Setosa plants have much lower P_l and P_w values than the other two plants.

The LPAM perfectly classified the iris Setosa and iris Versicolour (Class 0 and Class 1) samples and the iris Setosa and iris Virginica (Class 0 and Class 2) samples with an accuracy of 100%. It had the accuracy of 98% with θ equal to 0.5 when classifying the iris Versicolour and iris Virginica (Class 1 and Class 2) samples. However, the accuracy of classifying these two flowers was 100% when θ was set to 0.6, 0.7, 0.8 and 0.9. Overall, the LPAM was successful on the iris benchmark problem. A global model built

using a generalised linear model resulted in the accuracy of 87.5% ($\theta = 0.5$) and 100% when θ was set to 0.6, 0.7, 0.8 and 0.9

TABLE 4.2

Details of the iris dataset. Averages (μ) and standard deviations (σ) of the features in the iris dataset are expressed in centimetres and rounded to 2 decimal places (2.d.p.)

Class		S_l	S_w	P_l	P_w
0	μ	5.01	3.42	1.46	0.24
	σ	0.35	0.38	0.17	0.11
1	μ	5.94	2.77	4.26	1.33
	σ	0.52	0.31	0.47	0.20
2	μ	6.59	2.97	5.55	2.03
	σ	0.64	0.32	0.55	0.27

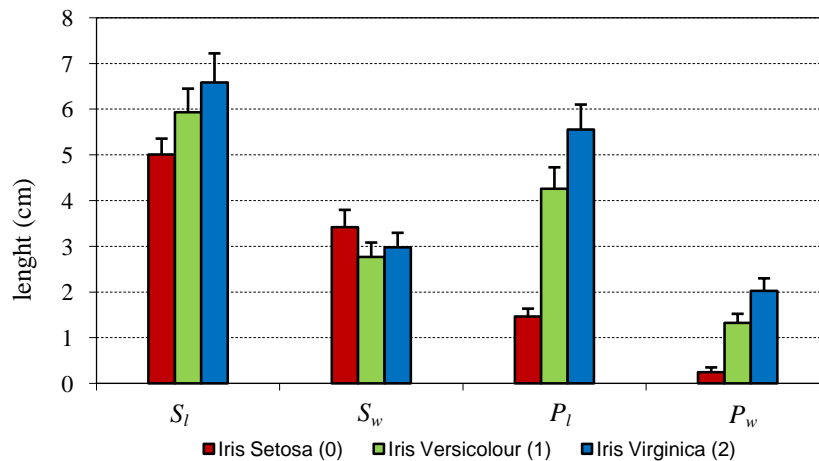


Fig. 4.4. The iris dataset - average values and standard deviations of the features of 50 samples in each of the three different classes, each class representing one type of iris plant.

Due to the low number of features this dataset is useful for validating the knowledge discovering capabilities of the LPAM. Two rules were extracted for the iris Versicolour and iris Virginica (Class 1 and Class 2) flowers (Table 4.3 and Fig. 4.5). Rule 1 describes iris flowers with smaller sepals and petals and Rule 2 describes iris flowers with larger petals and sepals. Based on regression coefficient values it can be seen that the petal features ($a_{1P_l} = 1.36$, $a_{1P_w} = 0.98$, $a_{2P_l} = 1.65$ and $a_{2P_w} = 0.80$) are more important in the classification of irises than the sepal features ($a_{1S_l} = -0.55$, $a_{1S_w} = -0.60$,

$a_{2S_l} = -0.84$ and $a_{2S_w} = -0.35$). The LPAM found that an increase in sepal features decreases the overall response (all a_{iS_j} were negative) of the LPAM model.

To assess the validity of the rules delivered by the LPAM, we plotted the Class 1 and Class 2 samples in two 2D spaces (Fig. 4.6). The rules favoured the petal features over the sepal features when deciding if a flower is an iris Versicolour plant (Class 1) or an iris Virginica plant (Class 2). By examining the Fig. 4.6 plots it can be seen that the samples are easier to distinguish in the petal space (Fig. 4.6.b) than in the sepal space (Fig. 4.6.a). There is a considerably bigger overlap of samples in the 2D sepal space than in the 2D petal space. Therefore, the LPAM rules that give bigger values to the petal regression coefficients than to the sepal ones are correct.

TABLE 4.3

Two rules extracted by the LPAM for the iris Versicolour and iris Virginica dataset.

	c	σ	$y = a_{r0} + a_{rS_l} S_l + a_{rS_w} S_w + a_{rP_l} P_l + a_{rP_w} P_w$
Rule 1:			
S_l	0.32	0.21	
S_w	0.05	0.28	$0.89 - 0.55 S_l - 0.60 S_w + 1.36 P_l + 0.98 P_w$
P_l	0.26	0.23	
P_w	0.25	0.24	
Rule 2:			
S_l	0.57	0.18	
S_w	0.57	0.18	$0.84 - 0.84 S_l - 0.35 S_w + 1.65 P_l + 0.80 P_w$
P_l	0.64	0.18	
P_w	0.64	0.18	

4.4.2 Validation on the wine dataset

The wine dataset is also a very popular benchmark dataset from the UCI Machine Learning Repository database. This dataset comprises samples of 3 types of wine (Class 0, Class 1 and Class3), where each sample is described by 13 features: alcohol (Al), malic acid (Ma), ash (A), alcalinity of ash (Aa), magnesium (Mg), total phenols (Tp), flavanoids (F), nonflavanoid phenols (Np), proanthocyanins (Pa), colour intensity (Ci), hue (H), OD280/OD315 of diluted wines (OD) and proline (P). The average values (μ) and standard deviations (σ) of all features in each of the three different wines are shown

in Table 4.4. Note that this dataset has a different number of data samples in the different classes. There are 59 Class 0 data samples, 71 Class 1 data samples and 48 Class 3 data samples.

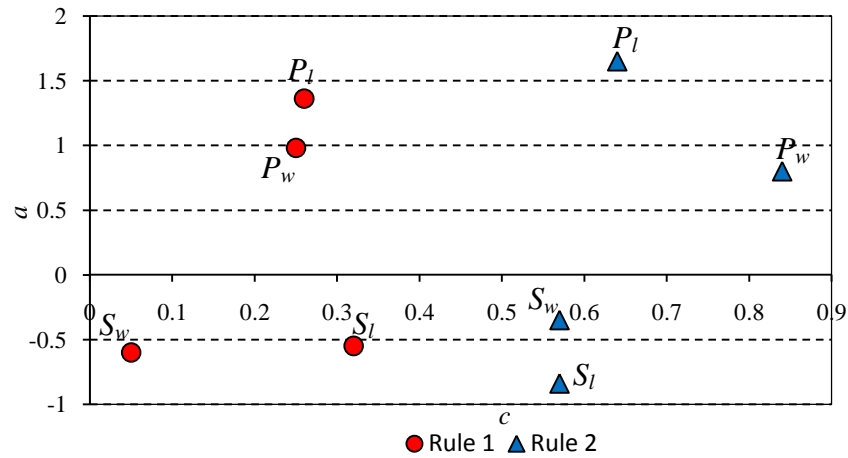


Fig. 4.5. The contributions of predictor variables in the two fuzzy rules for the iris Versicolour and iris Virginica dataset. In both rules P_l and P_w are more influential than S_l and S_w .

Again, the LPAM was successful in perfectly classifying Class 0 and Class 1 wines and Class 0 and Class 2 wines with an accuracy of 100%, but it had the accuracy of 94% ($\theta = 0.5$) when classifying Class 1 and Class 2 wine samples. Nevertheless, the accuracy of classifying these two wines was 100% for $\theta = 0.6, 0.7, 0.8$ and 0.9 . The global model built using a generalised linear model had an accuracy of 87.5 on the same set of data.

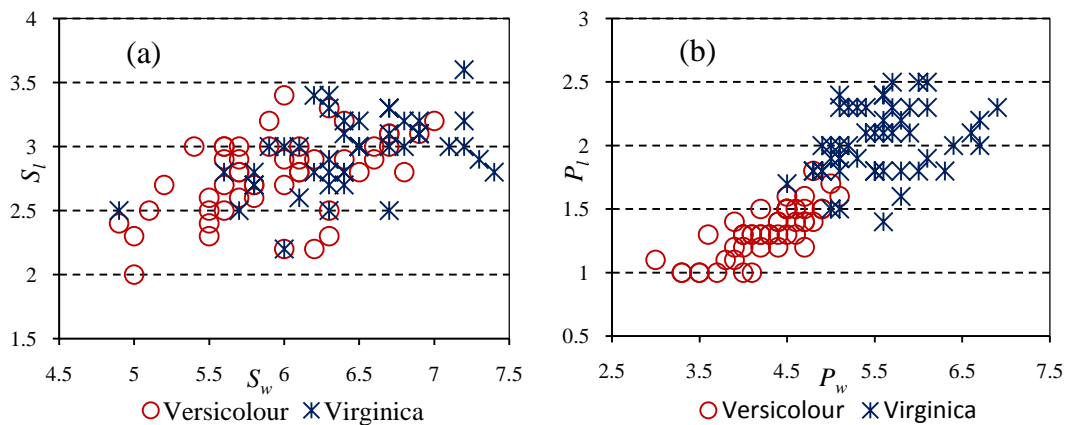


Fig. 4.6. Two features for the iris Versicolour and iris Virginica plotted in a 2D space. (a) 2D sepal space, S_l vs. S_w (b) 2D petal space, P_l vs. P_w .

Because there are 13 features in the wine dataset it is harder to examine the rules extracted about this dataset than that the rules that were extracted about the iris flowers.

We present here what the LPAM had discovered about the Class 0 and Class 1 wines (Table 4.5 and Fig. 4.7). To make the visualization of the rules more clear Rules 1 and 2 are shown in separate scatter charts. It is interesting to see that in both rules the same set of features are the most influential positive contributors, *i.e.* alcalinity of ash (*Aa*) with $a_{1Aa} = 0.96$ and $a_{2Aa} = 0.79$, proanthocyanins (*Pa*) with $a_{1Pa} = 0.33$ and $a_{2Pa} = 1$ and total phenols (*Tp*) with $a_{1Tp} = 0.40$ and $a_{2Tp} = 1.12$. This does not mean that the influences of other features are less important. On the contrary, these features are essential for the classification. They contribute to the recognition of samples. For example, in Rule 2 *F* is the most influential negative contributor with $a_{2F} = -2.4$. The same feature has a very small influence in Rule 1 ($a_{1F} = -0.08$). In both rules, the sign of the contributions of all features but magnesium (*Mg*) and alcohol (*Al*) are the same, though the values of their contributions are different. While *Mg* has a positive contribution in Rule 1 ($a_{1Mg} = 0.06$) its contribution is negative in Rule 2 ($a_{2Mg} = -0.34$). There are also features with the same contributions. Here, the contributions of *Ma* are exactly the same in the both rules ($a_{1Ma} = a_{2Ma} = -0.37$).

TABLE 4.4

Details of the wine dataset. Averages (μ) and standard deviations (σ) of the features in the wine dataset are rounded to 2.d.p.

	Class 0		Class 1		Class 2	
	μ	σ	μ	σ	μ	σ
<i>Al</i>	13.74	0.46	12.28	0.54	13.15	0.53
<i>Ma</i>	2.01	0.69	1.93	1.02	3.33	1.09
<i>A</i>	2.46	0.23	2.24	0.32	2.44	0.18
<i>Aa</i>	17.04	2.55	20.24	3.35	21.42	2.26
<i>Mg</i>	106.34	10.50	94.55	16.75	99.31	10.89
<i>Tp</i>	2.84	0.34	2.26	0.55	1.68	0.36
<i>F</i>	2.98	0.4	2.08	0.71	0.78	0.29
<i>Np</i>	0.29	0.07	0.36	0.12	0.45	0.12
<i>Pa</i>	1.90	0.41	1.63	0.60	1.15	0.41
<i>Ci</i>	5.53	1.24	3.09	0.92	7.40	2.31
<i>H</i>	1.06	0.12	1.06	0.20	0.68	0.11
<i>OD</i>	3.16	0.36	2.79	0.50	1.68	0.27
<i>P</i>	1115.71	221.52	519.51	157.21	629.90	115.10

TABLE 4.5

Two rules extracted by the LPAM for the Class 0 and Class 1 wines.

	c	σ	$y = a_{r0} + a_{rAl} Al + a_{rMa} Ma + \dots + a_{rP} P$
Rule 1:			
<i>Al</i>	0.79	0.23	
<i>Ma</i>	0.27	0.24	
<i>A</i>	0.51	0.21	
<i>Aa</i>	0.26	0.21	
<i>Mg</i>	0.38	0.20	2.29 - 0.25 <i>Al</i> - 0.37 <i>Ma</i> - 0.66 <i>A</i> + 0.96 <i>Aa</i> + 0.06 <i>Mg</i> +
<i>Tp</i>	0.80	0.18	0.4 <i>Tp</i> - 0.08 <i>F</i> + 0.01 <i>Np</i> + 0.33 <i>Pa</i> - 0.1 <i>Ci</i> - 0.15 <i>H</i> -
<i>F</i>	0.52	0.19	0.85 <i>OD</i> - 0.8 <i>P</i>
<i>Np</i>	0.30	0.19	
<i>Pa</i>	0.53	0.22	
<i>Ci</i>	0.78	0.22	
<i>H</i>	0.44	0.22	
<i>OD</i>	0.68	0.19	
<i>P</i>	0.75	0.22	
Rule 2:			
<i>Al</i>	0.12	0.20	
<i>Ma</i>	0.27	0.17	
<i>A</i>	0.53	0.19	
<i>Aa</i>	0.48	0.20	
<i>Mg</i>	0.21	0.20	2.03 + 0.06 <i>Al</i> - 0.37 <i>Ma</i> - 0.56 <i>A</i> + 0.79 <i>Aa</i> - 0.34 <i>Mg</i> +
<i>Tp</i>	0.40	0.16	1.12 <i>Tp</i> - 2.40 <i>F</i> + 0.11 <i>Np</i> + <i>Pa</i> - 0.18 <i>Ci</i> - 0.09 <i>H</i> -
<i>F</i>	0.24	0.19	0.41 <i>OD</i> - 0.71 <i>P</i>
<i>Np</i>	0.71	0.22	
<i>Pa</i>	0.33	0.17	
<i>Ci</i>	0.13	0.20	
<i>H</i>	0.95	0.29	
<i>OD</i>	0.36	0.19	
<i>P</i>	0.17	0.20	

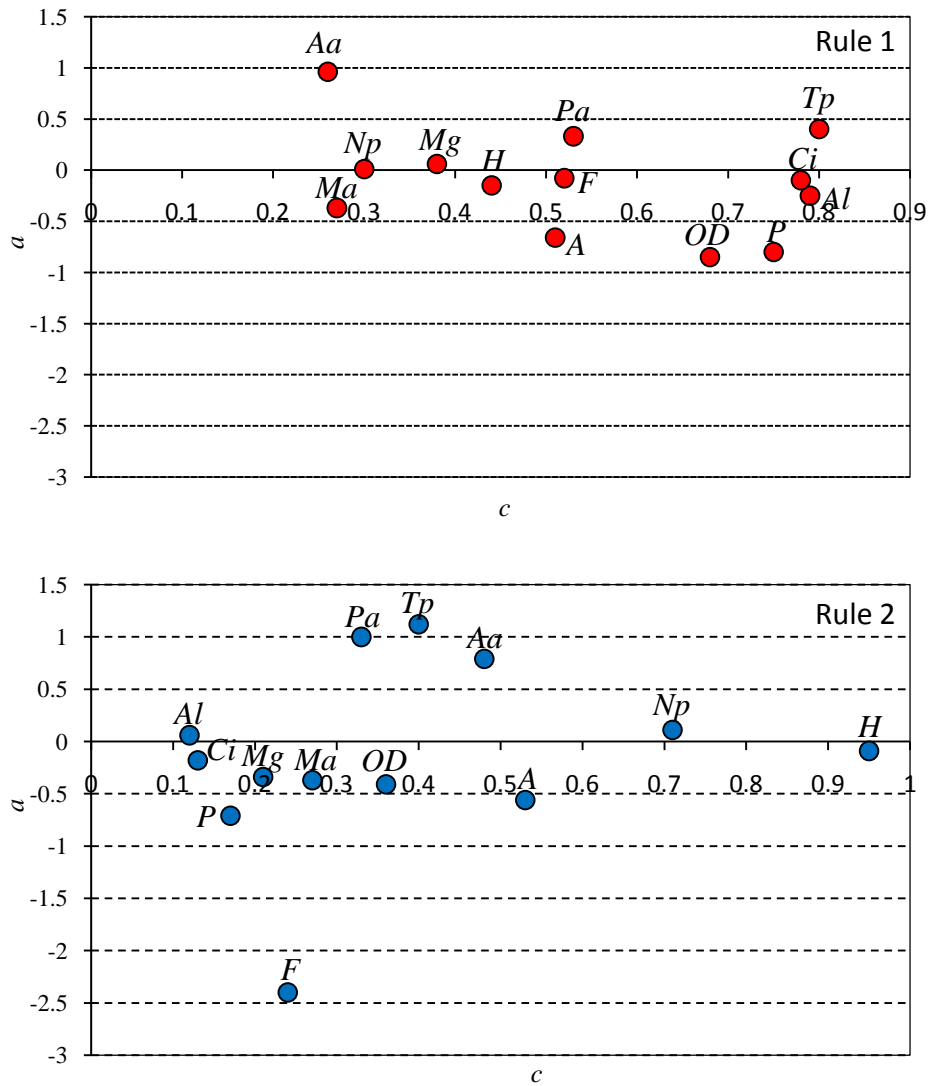


Fig. 4.7. The contributions of predictor variables in the two fuzzy rules for the Class 0 and Class 1 wines.

4.4.3 Comment on the knowledge discovery

Using the benchmark datasets to test the performance of a new model and particularly to assess the quality of the accumulated knowledge is useful before the model is applied on more complex real-world datasets. In this thesis a well understood benchmark dataset, *i.e.* the iris dataset, whose features can be examined in a 2D data space has been selected to initially test the proposed LPAM model. This evaluation approach revealed that the LPAM can indeed correctly learn and explain the data patterns in the iris datasets.

The local learning utilised by the LPAM resulted in two local models and the LPAM assembled knowledge in two rules. The response functions, y_{Rule1} and y_{Rule2} , were fine-tuned to capture local patterns in two classes of iris flowers.

$$y_{\text{Rule1}} = 0.89 - 0.55 S_l - 0.60 S_w + 1.36 P_l + 0.98 P_w$$

$$y_{\text{Rule2}} = 0.84 - 0.84 S_l - 0.35 S_w + 1.65 P_l + 0.80 P_w$$

Unfortunately, in many applications the influence of data characteristics on the modelled process is less obvious (*e.g.* a multi-dimensional datasets). For that reason, the LPAM was validated on a more complex dataset, *i.e.* the benchmark wine dataset where 13 features describe two brands of wine. Again, the LPAM created two local models, each favouring a different set of wine characteristics:

$$y_{\text{Rule1}} = 2.29 - 0.25 Al - 0.37 Ma - 0.66 A + 0.96 Aa + 0.06 Mg + 0.4 Tp - 0.08 F + 0.01 Np + 0.33 Pa - 0.1 Ci - 0.15 H - 0.85 OD - 0.8 P$$

$$y_{\text{Rule2}} = 2.03 + 0.06 Al - 0.37 Ma - 0.56 A + 0.79 Aa - 0.34 Mg + 1.12 Tp - 2.40 F + 0.11 Np + Pa - 0.18 Ci - 0.09 H - 0.41 OD - 0.71 P$$

We show later in this thesis that in datasets that exhibit bigger variations between samples belonging to a class, the LPAM approach builds more than two local models. This not only has potential to increase the modelling accuracy but also to deliver more precise knowledge. The LPAM partitions the problem space into clusters of similar patterns and explores patterns in each of clusters to acquire knowledge in local areas. We believe that a set of local models have a more powerful explanatory capability than a single global model. Even when no significant difference in terms of accuracy between two modelling approaches is obtained, the knowledge extracted from local experts is more detailed than one extracted from a one for all model. In the following chapter we show that the LPAM could be successfully used for risk evaluation in a more complex modelling environment, *i.e.* modelling the global distributions of pest insects.

4.5 Conclusion

In this chapter, an adaptive model based on local probability, named LPAM, for risk prediction was introduced. The model can be used to estimate risks, while also allowing us to determine, in a transparent way, the influence of each predictor variable on the

modelled output. The LPAM can also be used for classification by applying a threshold to the output value. In addition, a simple technique for visualising the predictor-response relationships based on the rules extracted during the training of an LPAM is proposed.

Two main features of the LPAM must be highlighted. First, it is an adaptive model as it evolves to accommodate new data samples as they become available. This feature has already been recognised in the ecological modelling community as necessary for modelling of effects of climate change on ecological processes. Second, it has a knowledge extraction facility that allows an easy extraction and identification of contributing variables. The proposed approach identifies a number of local clusters in a set of input data samples, each with unique predictor-response relationships. Adopting the LPAM can lead to acquiring knowledge that is impossible to extract through the use of traditional neural network models. However, to achieve this two LPAM parameters must be optimised, the number of local models ζ and the threshold value D_{thr} . In our work, these parameters were manually tuned and therefore there is an opportunity for future work to look at automating this process.

The LPAM was evaluated on two benchmark problems from the UCI Machine Learning Repository database. The accuracy of the LPAM on both benchmark problems was 100%. The first dataset, the iris dataset, was chosen because it has a small number of features. As a result, a thorough evaluation of the knowledge discovery capabilities of the LPAM was possible. We proved that the LPAM was successful both in terms of classification accuracy and knowledge discovery. The second dataset, the wine dataset, was employed to evaluate the performance of the LPAM on sets with an unequal number of samples per class as well as to see how the model performs on high-dimensional sets. Again, the LPAM was successful on this dataset. In the next chapter, we present the results of using the LPAM for assessing the establishment potential of a pest insect and validate the obtained results and compare our work to two more recent works where traditional neural networks were used on similar datasets.

Chapter 5

Modelling and prediction of establishment of the insect *P. citri* using LPAM

As discussed in Chapter 3, MLP networks are the most popular neural networks for building species distribution models. Unfortunately, MLP-based models cannot easily provide an interpretation of the effect of the individual modelling variables on the species establishment nor can they adapt their structure to environmental and data changes. To see if we can get around these problems in this chapter we apply the LPAM for assessing the establishment potential of a pest insect and compare the results to those of two more recent studies where traditional neural networks (MLP and SOM) were used on similar datasets.

Acknowledging that data is important to the performance of an ecological model a thorough analysis of the characteristics of the data used in the modelling is conducted and the results of this analysis are presented. The details of these results are given together with an explanation of the discoveries that were made by using the LPAM for the case study data. Finally, for the first time, a predictive map is produced and presented to facilitate the visual inspection of the distributions of the studied insect.

5.1 Introduction

An LPAM was used to develop a forecasting model to predict the establishment potential of the citrus mealybug, *Planococcus citri* (Risso). Unfortunately, currently there is little known about the response of this insect to influential environmental variables. This interpretation would be very useful to researchers who study insect-environment relationships and for those working on pest management.

It must be noted that this pest species is very widespread. Consequently, it has been challenging to forecast its dynamics. Yet, as will be shown in this thesis, the LPAM delivered valuable results that could not be obtained using traditional analysis tools usually employed for solving ecological studies problems. For example, with the LPAM, we explain how the climatic characteristics at various locations influence the

insect's potential distribution. The LPAM also provides a mechanism to reduce the influence of false data on the modelling accuracy. Overall, the experimental results show that the LPAM is able to learn to predict the establishment potential of insect species from eco-climatic data with the accuracy similar to the accuracy obtained in other studies.

5.2 Data characteristics

For this study, the meteorological data for more than 6000 worldwide geographic areas where the *Planococcus citri* (Risso) has been recorded as either present (P, 1) (223 locations) or is considered absent (A, 0) was assembled from published sources (Peacock, 2005). This insect is extremely widespread (Waterhouse & Sands, 2001), and so data is represented from all continents. The *P. citri* is recorded in the dataset as present in New Zealand (CABI, 2004) but the New Zealand biosecurity community consider this insect absent since the 1980s. In an experiment studying the establishment potential of the 106 pest species with potential to establish in New Zealand *P. citri* was found to have the highest establishment potential (Worner & Gevrey, 2006).

The influence of various climate variables on the establishment of *P. citri* was studied by Peacock, Worner and Sedcole in 2006. They used discriminate analysis to determine the climatic preferences of 35 insects including *P. citri* and found that the presence of different species is influenced by a different set of climatic variables. Out of 34 climate variables, soil moisture, rain, temperature and evapotranspiration were found to influence the establishment potential of insects more than any other climate characteristics which were studied. However, which variables represent the limiting factor differs from species to species and more often than not the true limiting factors are not known. In another study annual actual evapotranspiration was used with other climatic factors to predict reptile and amphibian species richness (Rodríguez, Belmontes & Hawkins, 2005) and to predict the distribution of various insects (Rodríguez & Gorla, 2004; Bayoh, Thomas & Lindsay, 2001). An aridity factor indicating the dryness of an area together with the temperature and some other climatic characteristics were used to model the distribution of a moth species (Chefaoui & Lobo, 2008) and melon thrips (Dentener, Whiting & Connolly, 2002). Another study found that temperature is one of the most important climatic variables influencing the distribution of pest insects (Zhang & Zhang, 2008; Ulrichs & Hopper, 2007; Watts & Worner, 2008). In our experiments we used a mixture of climate variables shown in Table 5.1.

TABLE 5.1Climate variables used to model the establishment potential of the insect *P. citri*.

Variable name (unit)	Symbol	Range	Average
Maximum summer temperature (°C)	T_{Smax}	2.5 – 36.6	24.8
Minimum winter temperature (°C)	T_{Wmin}	-35.2 – 27.2	11.1
Mean total rainfall (mm)	R_{mean}	1 – 4331	1090.6
Annual actual evapotranspiration (mm)	AAE	1 – 1903	691.5
Moisture index (mm)	MI	0 – 8.1	1.1

To investigate the dataset the exploratory analyses explained in Chapter 3 Section 3.4 was conducted. First, to avoid the influence of a large number of locations where a species is absent, the number of locations where this pest insect is recorded absent was kept similar to the number of locations where this pest insect is recorded as present. Second, the spatial distribution of the locations was checked to identify possible geographical outliers. Third, the insect's response to the chosen predictor variables was investigated. Fig. 3.5 (Chapter 3) shows the distribution of *P. citri* over the T_{Smax} range. Fig. 5.1 (a) – (d) shows the distributions over the other climatic variables' ranges. From these figures it is possible to deduce that the value of the present (n_p) to absent (n_a) ratio r increases with an increase in temperatures, particularly with T_{Wmin} , and with an increase in R_{mean} and AAE . Also, there is a noticeable lack of sampled locations which have a drier climate (*i.e.* higher MI values). Finally, a correlation analysis on the predictor variables was carried out (Table 5.2). The highest correlation was between R_{mean} and AAE ($r = 0.84$). This higher correlation was not a problem for our method because evolving systems such as the LPAM tolerate learning from correlated variables (Kasabov, 2002). As expected, correlations between MI and both temperature attributes were negative, as higher temperatures will decrease the MI towards increasing dryness. The influence of T_{Wmin} on MI is lower ($r = -0.08$) than the influence of T_{Smax} ($r = -0.45$).

TABLE 5.2Correlation matrix of the predictor variables for the insect *P. citri*.

R_{mean}	0.84	0.71	0.07	0.44
0.84	AAE	0.39	0.26	0.59
0.71	0.39	MI	-0.45	-0.08
0.07	0.26	-0.45	T_{Smax}	0.58
0.44	0.59	-0.08	0.58	T_{Wmin}

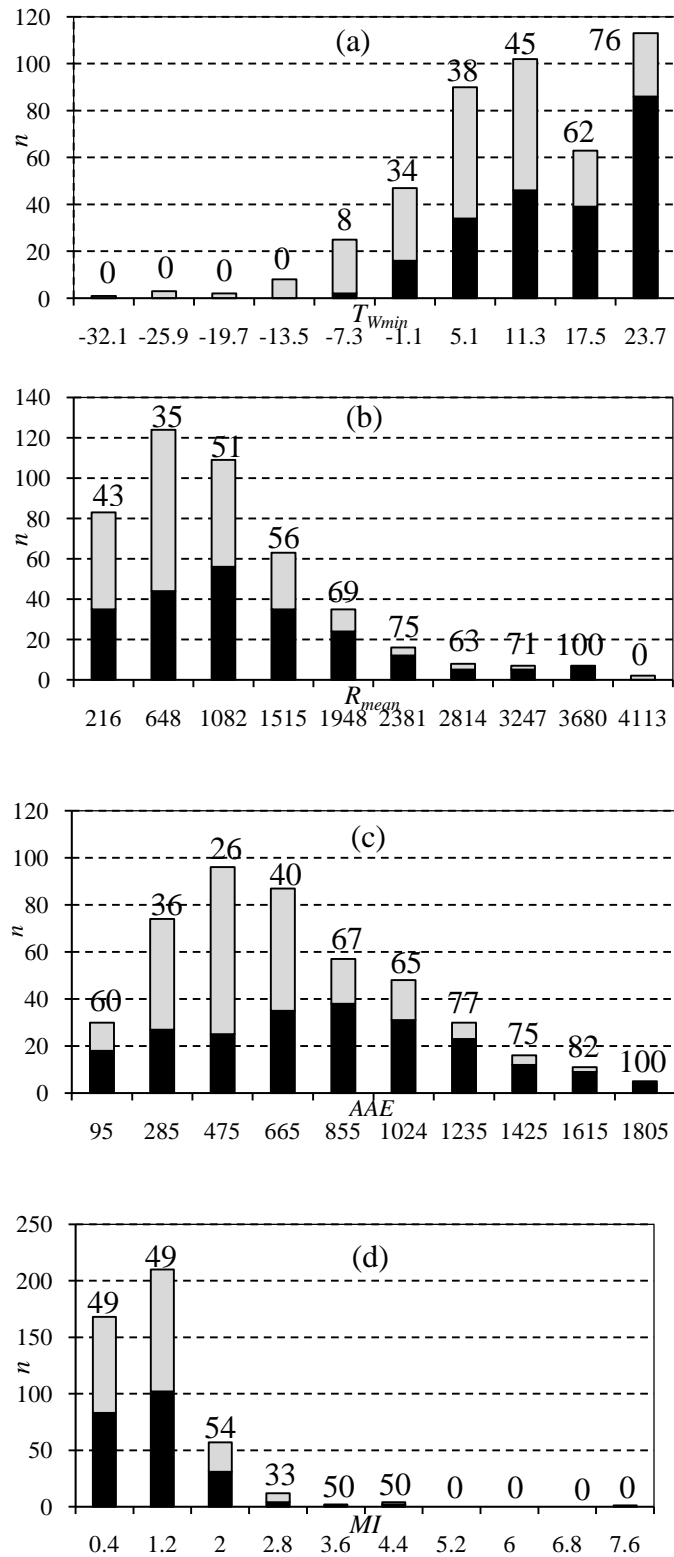


Fig. 5.1. Distributions of the insect *P. citri* over a range of T_{Wmin} (a), R_{mean} (b), AAE (c) and MI (d). Each bar represents the distribution of all locations (n), where the dark areas represent the number of plots occupied by the insect (n_p). The number written on top of each bar shows the percentage of all sites occupied by the insect ($r = n_p \div n$ %). The distribution over the T_{Smax} is given in Chapter 3 Fig. 3.5.

Next, the results of our modelling are presented. First, we present the results of the cluster analysis. This is followed by the explanation of the rules discovered by the LPAM. Then, we present the predictive map for the modelled pest insect. Finally, we compare our results to those achieved when MLP and SOM were used.

5.3 Clustering information

The results of the ECM clustering are summarised in Table 5.3. The cluster size varies from one hundred locations to only one location. The r column shows the number of locations at which the insect is present against the total number of locations for a given cluster, *i.e.* $r = n_p \div n$. Cluster 4 is the largest cluster with 100 locations. It contains locations in Central and South America; Central, East and Southern Africa; South, East and Southeast Asia and North and East Australia. The locations in this cluster have the third highest ratio ($r = 0.73$) between present and absent locations (*i.e.* the *P. citri* is present at 73 and absent from 27 locations). The highest ratio ($r = 0.85$) is in Cluster 8, which includes 13 locations, mainly from Southeast Asia. The smallest clusters (containing one location) are Cluster 15 (Switzerland), Cluster 18 (Chile), Cluster 19 (Canada) and Cluster 20 (Guinea). The *P. citri* is absent from all of these four clusters.

It is interesting to observe that the insect is also absent from the two larger clusters, Cluster 11 (12 locations) and Cluster 13 (24 locations). The table also includes the mean values of the attributes for all clusters. Higher r are obtained for clusters with higher average values for the temperature parameters T_{Smax} and T_{wmin} , which supports the observation that the insect prefers warmer sites. Nine New Zealand locations can be found in four clusters, Clusters 1, 3, 5 and 13 (Table 5.4). The ratio r is equal to 0.45, 0.28, 0.5 and 0 respectively. There are seven New Zealand locations (78%) in clusters where $r \leq 0.28$. The remaining two locations belong to the clusters where $r = 0.45$ and $r = 0.5$.

Overall, the insect is more frequently present (*i.e.* there is a high ratio r) in Clusters 8, 7, 4, etc, which do not contain any NZ locations, and less often present (*i.e.* there is a low r) in the clusters which include NZ locations. Thus, the *P. citri* is more strongly associated with sites that are not clustered or associated with New Zealand sites.

TABLE 5.3

Details of the ECM cluster analysis.

Cluster	Samples	r	R_{mean}	AAE	MI	T_{Smax}	T_{Wmin}
1	91	0.45	1102.6	715.4	1.34	24.3	7.1
2	76	0.51	328.8	260.8	0.34	27.6	11.1
3	98	0.28	720.01	500.3	1.03	20.3	5.6
4	100	0.73	1407	1026.5	1.08	27.2	20.4
5	6	0.50	2509.5	651.3	3.77	17.4	6.3
6	61	0.21	643.7	471.7	0.96	22.5	-2.7
7	38	0.79	2105.8	1355.2	1.35	28.0	23.2
8	13	0.85	3591.2	1681.3	2.03	28.1	25.6
9	56	0.64	914.3	717.6	0.72	28.0	18.7
10	18	0.39	225.17	216.9	0.11	33.4	18.5
11	12	0	385.25	304.3	0.71	19.6	-15.7
12	4	0.5	2223.3	764.7	2.90	24.5	4.3
13	24	0	1124	556.2	1.86	17.3	1.1
14	25	0.68	1728.5	855.8	1.79	22.9	17.7
15	1	0	2744	338	8.12	5.8	-8.9
16	10	0.70	2730.6	1190.1	1.86	27.6	22.4
17	2	0.50	1151	338	3.49	5.6	-10
18	1	0	398	349	0.72	10.8	1.7
19	1	0	112	104	0.31	7.2	-35.2
20	1	0	4003	943	2.51	27.6	25

TABLE 5.4

Clustering results for New Zealand. There are seven New Zealand locations (78%) in clusters where $r \leq 0.28$. The *P. citri* is more strongly associated with sites that are not clustered or associated with New Zealand sites.

Cluster	NZ	r
1	1	0.45
3	4	0.28
5	1	0.50
13	3	0

5.4 Establishment and climatic variables

The trained LPAM was used to establish the relationship between the response variable (presence/absence) and the set of predictor (climate) variables (Table 5.1). The climate variables used to model the establishment potential of the insect *P. citri* were suggested by an ecological expert (Worner, personal communication). During the learning phase its DENFIS module created three climatic environmental clusters (environmental sub-envelopes), each representing locations with unique climatic characteristics. Locations are represented by three fuzzy rules (Table 5.5). For example, Rule 1 states that IF the value of R_{mean} is in the area defined by a Gaussian function with the mean of 0.29 (*i.e.* $c_{1R_{mean}} = 0.29$) and the standard deviation of 0.29 (*i.e.* $\sigma_{1R_{mean}} = 0.29$) AND AAE is in the area defined by a Gaussian function with $c_{1AAE} = 0.41$ and $\sigma_{1AAE} = 0.17$ AND MI is in the area defined by a Gaussian function with $c_{1MI} = 0.19$ and $\sigma_{1MI} = 0.16$ AND T_{Smax} is in the area defined by a Gaussian function $c_{1T_{Smax}} = 0.69$ AND $\sigma_{1T_{Smax}} = 0.32$ and T_{Wmin} is in the area defined by a Gaussian function with $c_{1T_{Wmin}} = 0.91$ and $\sigma_{1T_{Wmin}} = 6.13$ THEN the response is calculated using the following formulae $y = 0.61 - 1.47 R_{mean} + 1.46 AAE + 1.91 MI + 0.44 T_{Smax} + 0.28 T_{Wmin}$.

The contributions of the predictors in each rule are shown in Fig. 5.2. It is possible to deduce from these graphs the extent to which the predictors influence the response. As explained in Chapter 4 Section 4.3, the centres of Gaussian functions (c_{ri}) and the coefficients of the regression lines (a_{ri}) are used explain the influence of each climatic variable on the establishment of the modelled pest insect.

Rule 1 describes the locations with a high minimum winter temperature ($c_{1T_{Wmin}} = 0.91$); a moderate maximum summer temperature ($c_{1T_{Smax}} = 0.69$) and annual actual evapotranspiration ($c_{1AAE} = 0.41$); dry soil ($c_{1MI} = 0.19$) and a low total mean rainfall ($c_{1R_{mean}} = 0.29$). MI ($a_{1MI} = 1.91$) has the greatest effect. Also, a decrease in R_{mean} causes an increase in the response.

The locations described by Rule 2 have a low mean for the annual actual evapotranspiration ($c_{2AAE} = 0.18$) and maximum summer temperature ($c_{2T_{Smax}} = 0.05$) values; a moderate mean for rainfall ($c_{2R_{mean}} = 0.66$) and winter temperatures ($c_{2T_{Wmin}} = 0.44$) values; but they have very wet soil ($c_{2MI} = 0.95$). AAE ($a_{2AAE} = 8.26$) has the greatest effect for this group of locations, and a decrease in winter and summer temperature causes an increase in the response.

All five variables in Rule 3 have the low or very low mean value ($c_{3R_{mean}} = 0.18$, $c_{3AAE} = 0.03$, $c_{3MI} = 0.06$, $c_{3T_{Smax}} = 0.11$, $c_{3T_{Wmin}} = 0.04$); low $c_{3R_{mean}}$ and c_{3AAE} values indicate overall dry conditions, and low $c_{3T_{Smax}}$ and $c_{3T_{Wmin}}$ values indicate low temperatures. It is interesting to see that the influence of the predictor variables on the response is high; especially the influence of the rainfall variable ($a_{3R_{mean}} = -13.93$, *i.e.* less rain results in a higher response), annual actual evapotranspiration ($a_{3AAE} = 11.63$), and *MI* ($a_{3MI} = 8.21$). *AAE* is one of the most influential predictors in all three rules ($a_{AAE} = 1.46$, $a_{2AAE} = 8.26$ and $a_{3AAE} = 11.63$).

TABLE 5.5
Knowledge extracted by the LPAM.

	c	σ	$y = a_{r0} + a_{rR_{mean}} R_{mean} + a_{rAAE} AAE + a_{rMI} MI + a_{rT_{Smax}} T_{Smax} + a_{rT_{Wmin}} T_{Wmin}$
Rule 1:			
R_{mean}	0.29	0.29	
AAE	0.41	0.17	$0.61 - 1.47 R_{mean} + 1.46 AAE + 1.91 MI + 0.44 T_{Smax} +$
MI	0.19	0.16	$0.28 T_{Wmin}$
T_{Smax}	0.69	0.32	
T_{Wmin}	0.91	6.13	
Rule 2:			
R_{mean}	0.66	0.25	
AAE	0.18	0.20	$1.52 + 2.01 R_{mean} + 8.26 AAE - 2.30 MI - 3.77 T_{Smax} -$
MI	0.95	0.29	$2.07 T_{Wmin}$
T_{Smax}	0.05	0.20	
T_{Wmin}	0.44	0.19	
Rule 3:			
R_{mean}	0.18	3.37	
AAE	0.03	0.05	$0.65 - 13.93 R_{mean} + 11.62 AAE + 8.21 MI - 0.51 T_{Smax}$
MI	0.06	0.17	$- 1.60 T_{Wmin}$
T_{Smax}	0.11	0.21	
T_{Wmin}	0.04	0.06	

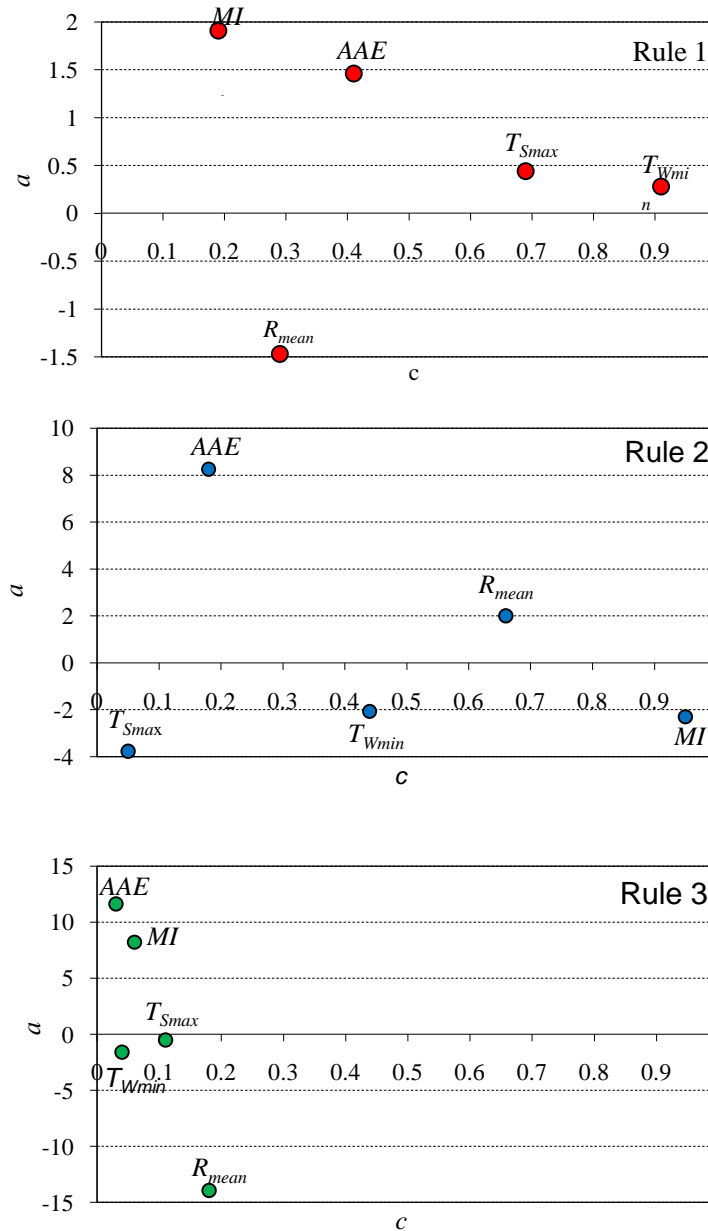


Fig. 5.2. Contributions of predictor variables in three different fuzzy rules. Each rule explains the influence of five climate characteristics on the establishment of the pest insect. Rule 1: higher rainfall \Rightarrow lower response, the main positive contributors are $a_{IMI} = 1.91$ and $a_{IAAE} = 1.46$. Rule 2: higher MI , T_{Smax} and $T_{Wmin} \Rightarrow$ lower response, the main positive contributor is $a_{2AAE} = 8.26$. Rule 3: all contributors have low mean values, higher $R_{mean} \Rightarrow$ lower response, the main positive contributors are $a_{3AAE} = 11.62$ and $a_{3MI} = 8.21$.

The trained LPAM can be used to estimate the insect's establishment potential for each location from the original data set D . Table 5.6 summarizes the result for 20 selected locations.

TABLE 5.6

The results for 20 selected locations. The first 10 locations were chosen because they were given the highest establishment potential estimates. The second 10 locations were given the lowest overall estimates.

Location	(Lat, Long)	Risk index	P or A
Australia, Kununurra	(-15.4, 128.5)	1	0
Australia, Tyalgum	(-28.4, 153.2)	0.95	0
Sardinia, Sassari	(40.7, 8.6)	0.95	1
Malta,	(35.5, 14.3)	0.95	1
Brazil, Rio Grande do Sul	(-30, -53)	0.94	1
Mozambique, Mungari	(-17.2, 33.5)	0.94	0
Galapagos,	(0, -91)	0.94	1
Peru, Lima	(-12.1, -77)	0.91	1
UK	(53, -2)	0.91	0
Burkina Faso, Upper Volta	(12, -1)	0.91	1
Argentina, Misiones	(-27, -55)	0.42	1
Madagascar, Ankavandra	(-18.8, 45.3)	0.39	0
Turkmenistan	(39, 59)	0.38	1
USA, Tustin	(33.7, -117.8)	0.33	0
South Africa, Matroosberg	(-33.4, 19.8)	0.31	0
USA, Pearl	(32.7, -103.4)	0.29	0
China, Datong	(40.1, 113.33)	0.29	0
New Zealand, Eyrewell forest	(-43.4, 172.3)	0.27	0
USA, Winslow/Mun. AZ	(35.02, -110.73)	0.07	0
Saudi Arabia, Hafr al Batin	(28.33, 46.12)	0	0

The first 10 locations were chosen because they were found to have the highest risk of establishment of the pest insect. The second 10 locations were given the overall lowest establishment potential estimates. In other words, it is unlikely that the pest insect will establish at these 10 locations. Each location is described by a pair of geographic coordinates (latitude, longitude), which are given in the second column. The model predictions are tabulated in the ‘*Risk index*’ column. For the purpose of comparison, the ‘*P or A*’ column (present = 1, absent = 0) shows the known (recorded) establishment status of the pest insect. The highest establishment was predicted for two locations from Australia (-15.4, 128.5) and (-28.4, 153.2). Even though these two locations do not

currently have this pest insect the model suggests that they should be treated as areas where this pest insect could potentially establish upon its introduction. Again, we must bear in mind that the status of this pest at these two Australian locations, as recorded in the modelling data, may be incorrect, as it is possible that the insect could thrive there if introduced.

5.5 Risk map

To complete the modelling, a predictive map showing a general trend of establishment potential for *P. citri* in different areas of the world was compiled (Fig. 5.3). The areas were classified into 6 ranges with the establishment potential of 0, 0.2, 0.4, 0.6, 0.8 and 1, where the lower values represent a lower risk of insect establishment, and the higher values represent possible hot spots (biharmonic spline surfaces were created using the interpolation method with 1° spacing). Results plotted in Fig. 5.3 are very similar to the spatial distribution of the insect *P. citri* shown in Chapter 3 Fig. 3.4. The highest establishment potential (0.8 or 80%) was predicted for the Caribbean, north of South America, Oceania, and South, Southeast and East Asia, where the insect has already established. Most of New Zealand is in the range of 0.4 (40%). These results indicate that caution should be exercised when dealing with produce coming into New Zealand from the countries where the insect has been established.

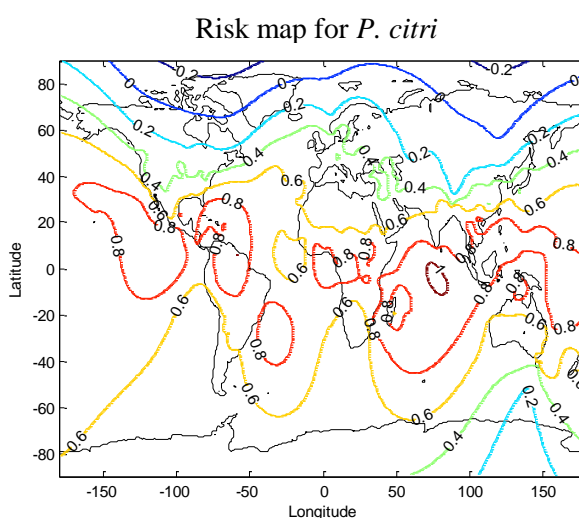


Fig. 5.3. The establishment probability map for the insect *P. citri* showing distributions predicted by the LPAM. Higher values represent the possible hot spots – the locations where the climate is suitable for the pest insect establishment. The map contours were created using the biharmonic spline interpolation method with 1° grid spacing available in MATLAB.

5.6 Influence of the threshold parameter

As stated earlier, The LPAM can be used for classification. In that case, the threshold parameter θ is used to convert the continuous output values generated by the LPAM model into binary presence or absence predictions. Changing θ changes all values in the confusion matrix affecting the performance of the models. Given a threshold value θ and a location g_i , a response greater than or equal to θ (i.e. where the ‘Risk index_{*i*}’ $\geq \theta$) means that the pest is likely to establish at this location ($y_i = 1$ or presence), while a response where the ‘Risk index_{*i*}’ $< \theta$ means the location’s climate is not favourable for the insect’s establishment ($y_i = 0$ or absence). Typically, θ is equal to 0.5. If y_i is equal to the recorded presence/absence value, then the prediction is correct.

The proposed model was evaluated over 7 probability thresholds, from 0.3 to 0.9 in steps of 0.1, at which the presence of the insect might be accepted (Fig. 5.4). The accuracy a , defined in (3.3), was assessed over the range $\theta \in [0.3, 0.9]$. As shown in Fig 5.4, the model’s accuracy in predicting the presence of the insect increases as θ increases, exhibiting values greater or equal to 80% for $\theta = 0.8$ and $\theta = 0.9$. For $\theta = 0.7$ $a = 78.6\%$.

The accuracy of the LPAM was also measured using Cohen’s Kappa statistic (3.1) at $\theta = 0.5$. As stated earlier in Chapter 3 Cohen’s Kappa statistic (κ) is often used for assessing the accuracy of predictive ecological models. The confusion matrix is given in Fig. 5.5 where $sen = 0.57$, $spe = 0.75$ and $\kappa = 0.33$ indicate that the model has difficulty modelling this species.

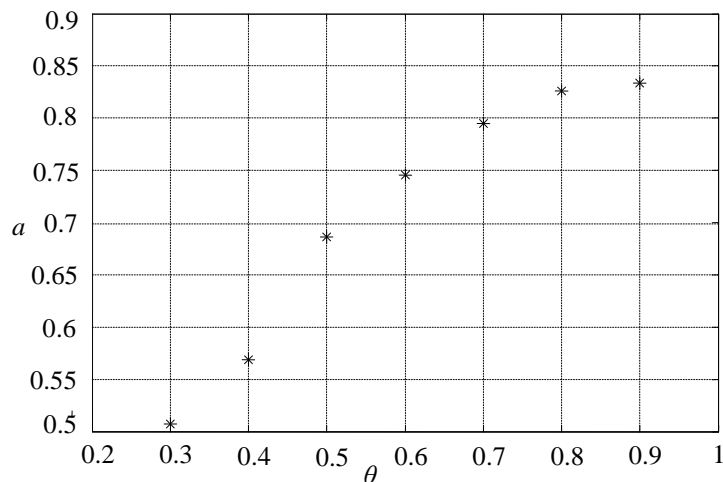


Fig. 5.4. The accuracy a of the predictive distribution model as a function of the threshold parameter θ . The accuracy is above 80% for $\theta = 0.8$ or 0.9 .

A number of factors could have influenced the performance of the model. First, it is possible that the climatic variables are poor predictors of the *P. citri*'s distribution patterns at the locations assessed here. Second, the results might be influenced by the way the data was collected. Particularly, the number of false absences was too high. Having the *spe* greater than the *sen* indicates that the model is biased to the absence data which is known to be very noisy. Third, there are suggestions that the distribution of this particular pest insect is particularly hard to model. This is further explored in the following section. Nevertheless, building predictive habitat models is a complex task and low κ values are not uncommon (Willems *et al.*, 2008; Watts & Worner, 2008; Eyre *et al.*, 2005).

	R_P	R_A
P_P	128	57
P_A	95	174

Fig. 5.5. Confusion matrix. R_P and R_A are the number of recorded presences and absences and P_P and P_A are the number of predicted presences and absences. 128 locations are correctly predicted as presences and 174 locations as absences.

5.7 Informative comparison

The proposed model and the results of the case study were published in 2003 and 2004 (Soltic *et al.*, 2004a & 2004b; Soltic *et al.*, 2003). Since then two similar studies involving the insect *P. citri* and its establishment potential have been conducted by Sue Worner and her team at Lincoln University, Canterbury, New Zealand where the traditional neural networks, *i.e.* MLP and SOM networks, were used as modelling tools (Watts & Worner, 2008; Gevrey *et al.*, 2006). In both studies, the establishment potentials of 844 insect pest species including *P. citri* were assessed. Despite their datasets and objectives not being exactly the same as the ones in this thesis, their work is the most similar currently published study available for comparison with this thesis. By looking at their work and how the results compare to our study, we can evaluate the significance and the benefits of the results achieved in this case study.

In (Watts & Worner, 2008) MLP networks were used to determine the relative contribution of various climatic variables to the establishment potential of 844 insect pest species at 495 geographic regions. The finding from their work that rain and

temperature variables were the predominant climatic variables influencing the distributions of pest insects supports the selection of predictor variables used in our case study. Furthermore, Watts and Worner concluded that *P. citri*'s complex behaviour is very hard to model. In their study, the performance of the model for *P. citri* was the lowest compared to the performance of the models built for all other species. While the accuracy of the model predicting the establishment potential of *P. citri* in this thesis is similar to theirs, the approach proposed here has additional benefits over the approaches using the MLP networks. Firstly, the training process of the proposed model consists of only one epoch compared to the 750 training epochs reported in (Watts & Worner, 2008). Secondly, our proposed approach does not suffer from catastrophic forgetting or the black-box syndrome. Clearly, there are advantages of the LPAM compared to the MLP. Thirdly, in the proposed approach there is no need for post-training descriptor variable analysis because the contribution analysis is done at the same time the network is trained. Fourthly, the knowledge extracted from the trained LPAM network is transparent.

In (Gevrey *et al.*, 2006) the establishment potential of 844 species, including *P. citri*, were analysed through the means of a SOM. The objective was to estimate the threat of various insect pest establishments in New Zealand. Whereas the approach in this thesis is based on climate, they modelled an insect's distribution at a particular area as a function of the species assemblages at this area. They ranked the studied species in term of their strength of association with species assemblages. A SOM was employed to find the pest insect specie's assemblages and the weight values of the trained SOM were interpreted as risk estimates. The risk of *P. citri* invasion in New Zealand was found to be 0.93, much higher than the predictions reported in this work and in (Watts & Worner, 2008). The same SOM estimated a low risk of establishment for several species that are already established in New Zealand.

5.8 Conclusion

In this chapter, the results of using the LPAM for the modelling distribution of an invasive pest insect, *Planoccus citri* (Risso), are given. In comparison with two more recently published works, the LPAM approach had a similar accuracy but provided more knowledge about the variables involved. In other words, the LPAM approach builds predictive models that can be used to determine the establishment potential of species, while also allowing us to determine the influence of each exploratory variable

to the distribution of the modelled species. This is a new approach to ecological modelling, where models have been typically built using either the classical statistical methods or traditional neural networks.

Predicting the establishment potential of a species is challenging, especially when the species is a widespread pest insect such as the one used in this study. The quality of the modelling data is often questionable since the data includes information about the species' absences along with the species' presences. The results presented in this work were compared to similar work by Sue Worner and her team. Their more recent results support the results obtained in this work. They favour the usage of traditional neural networks for a pest insects risk assessment. While the accuracies achieved in this work and their work are similar, our approach has advantages over the other methods which are discussed in the text above. The two main advantages must be highlighted. First, the LPAM is an evolving model ready to accommodate new data samples as they become available. This feature has already been recognised in the ecological modelling community as necessary for successful modelling. Second, the LPAM has a local knowledge extraction facility that allows for easy extraction and identification of contributing variables to the establishment potential and distribution of species in the form of IF-THEN rules. These rules are easily extracted from the trained model and they are intended to support expert knowledge.

In this case study, the LPAM identified three unique climatic regions, each with unique climate-insect relationships:

1. Dry sites with warm winters and moderate summer temperature where the soil dryness is the most influential factor and where the sites with more rain are less likely to be invaded by the insect;
2. Wet sites with lower summer temperatures and moderate winter temperatures with annual actual evapotranspiration having the highest influence and where decreases in summer and winter temperatures increase the risk of establishment;
3. Dry and cool regions where the establishment is positively influenced by annual actual evapotranspiration and soil dryness and reduced by the amount of rainfall.

Furthermore, an establishment probability map for the insect based on the modelling results was generated. To our knowledge this was the first map of its type created for this pest insect. For the *P. citri* the LPAM has given New Zealand the establishment probability of around 40%. This result generally agrees with the fact that the insect used

to be present in New Zealand and therefore it is likely that this insect would establish a viable population again if it was given the chance. The highest likelihood of establishment was found for locations where the insect has already been established validating the proposed approach.

This case study has shown how the LPAM can be used to predict the establishment of species and determine the role of various explanatory variables in a pest insect's distributions. Although, the approach was tested on a case study involving one pest insect there are no reasons to believe that the approach will not work for other species and other set of exploratory variables.

Finally, it is clear that the approach must be tested further and evaluated by experts from biosecurity, species biology and entomology. Adopting the LPAM might lead to acquiring knowledge that is impossible to extract through the use of the traditional neural network models.

Chapter 6

Transductive approach for 'personalised' modelling and knowledge discovery with case studies from biosecurity

This chapter reports on the applicability of transductive modelling techniques for the estimation of the establishment potentials of pest insects. The focus is on the case studies where the presence and absence of species are described by a set of climatic attributes. To study this, four modelling approaches, including the LPAM, based on inductive and transductive reasoning, were used on three different real-world insect datasets. The models were compared and their effectiveness in predicting and interpreting the pest insect distributions are discussed.

6.1 Introduction

A vast majority of predictive models in ecological modelling apply inductive inference where a model is created from all training data and then is applied to the available test data (or new data). According to Vapnik (1998), building a model based on all available training data might be unnecessary. He proposes the use of a paradigm, referred to as transductive reasoning and transductive inference, where the focus of the modelling is on one single data sample. Accordingly, the predictive models are built on the basis of the knowledge of one particular data sample at the time.

Transductive methods were found to perform better than inductive methods for datasets with high and medium noise levels (Schwaighofer & Tresp, 2002) and for small and moderate datasets (Derbeko, El-Yaniv & Meir, 2003; Blum & Chawla, 2001). Acknowledging that real-world eco-climatic data, such as the data used in this thesis, is noisy and sparse, transduction rather than induction might produce more accurate species-habitat distribution models. Therefore, the suitability of using transductive reasoning in modelling the establishment potential of pest insects has been investigated in this thesis by building personalized models to model the distribution of three insects: *Aspidiella hartii* (Cockerell), *Geococcus coffeae* and *Xyleborus perforans*. These insects

are not currently present in New Zealand, but they are frequently found on fresh produce coming into New Zealand from various overseas locations where the insects are established (S. P. Worner, personal communication, 2003).

Four approaches were used to build predictive habitat models for each insect:

- Model 1 is based on a classic statistical method (linear regression) and inductive reasoning. Therefore, each Model 1 is a global model built from all available training data.
- Model 2 is a collection of local models based on DENFIS and built using inductive reasoning. These local models collectively model the distribution of a pest insect.
- Model 3 is the LPAM, the new model introduced in Chapter 4. This model is also a collection of local models.
- Model 4 is a model based on transductive modelling and therefore it is a collection of ‘personalized’ models, where each model is built for a specific locality. These models were built using weighted KNN method described in Chapter 2 Section 2.4 by (2.3) and (2.4).

To summarise, the objectives of this case study were to:

1. Apply inductive and transductive modelling techniques to model the relationships between an insect and its habitat;
2. Evaluate the performance of those models on different real-world insect data sets; and
3. Assess the knowledge delivered by the models.

The models were assessed in terms of their accuracy (measured in terms of the Cohen’s Kappa statistics κ) and explanatory capability. The results achieved in this case study have been partially published in (Soltic & Peacock, 2006a & 2006b).

6.2 Data characteristics

Three bio-climatic distribution data sets (Peacock, 2005) were used to assess the transductive and inductive approaches. In Chapter 3, Table 3.3 gives a summary of the number of samples in each dataset, together with the number of sites where each insect is recorded present or absent. Data for the insect *A. hartii* was compiled from 78 worldwide sites where the insect has been recorded as either present (36 sites) or where it is considered absent (42 sites). Data for the insect *G. coffeae* was compiled from 75 worldwide sites and data for the insect *X. perforans* from 191 worldwide sites. *G.*

coffea had been recorded as present at 38 sites and absent from 37 sites, while *X. perforans* had been recorded present at 92 sites and absent from 99 sites. Each site is labelled either as a location where the particular pest insect is present (1), or absent (0). The presence and absence of these insects are related to three climatic variables: maximum summer air temperature (T_{Smax}), minimum winter air temperature (T_{Wmin}) and annual potential evapotranspiration (AAE) (Table 6.1). Note that three climatic variables approximately cover the same range of values in all three datasets, though the lowest T_{Smax} values in the *X. perforans* dataset and the lowest T_{Wmin} values in the *G. coffea* dataset are significantly lower than those in other two sets.

TABLE 6.1

Range of three climate variables used to model the establishment potential of the following insects: *A. hartii*, *G. coffea* and *X. perforans*.

Symbol	<i>A. hartii</i>	<i>G. coffea</i>	<i>X. perforans</i>
T_{Smax} (°C)	13.3 – 33.8	16.6 – 38.3	8.6 – 35.6
T_{Wmin} (°C)	-21.9 – 27.2	-9.9 – 27	-28.9 – 27.2
AAE (mm)	447 – 2376	534 – 3081	441– 3081

Note that the number of sampled locations within each climatic range and datasets differs significantly (Figs. 6.1, 6.2 and 6.3). Again, the number written on top of each bar in the graphs shows the percentage of all sites occupied by the insect ($r = n_p \div n \%$). Fig. 6.1 shows the distribution of the insect *A. hartii* over the ranges of T_{Smax} (Fig. 6.1a), T_{Wmin} (Fig. 6.1b) and AAE (Fig. 6.1c). This insect prefers higher summer and winter temperatures. It occurs more often than not at the warmer locations where $T_{Smax} > 25^\circ\text{C}$ and $T_{Wmin} > 13^\circ\text{C}$. There are 29 locations in the 6 lower summer temperature ranges where the pest insect is either not found or very rarely found, 32 locations in the 7 lower winter temperature ranges and 37 locations in the 4 lower AAE ranges. Overall this pest insect prefers locations with higher temperature and annual evapotranspiration ($AAE > 1290\text{mm}$) values.

The insect *G. coffea* prefers locations with relatively high but not very hot summer temperature and warmer winter temperatures, but moderate to high annual evapotranspiration (Fig. 6.2). This pest insect was found more often at locations where $24^\circ\text{C} < T_{Smax} < 30.8^\circ\text{C}$, $T_{Wmin} > 16^\circ\text{C}$ and $1000\text{mm} < AAE < 1986\text{mm}$. Two thirds of the

sampled locations are in the T_{Smax} ranges not preferred by this pest insect. The number of present and absent locations in the T_{Wmin} and AAE ranges is more balanced.

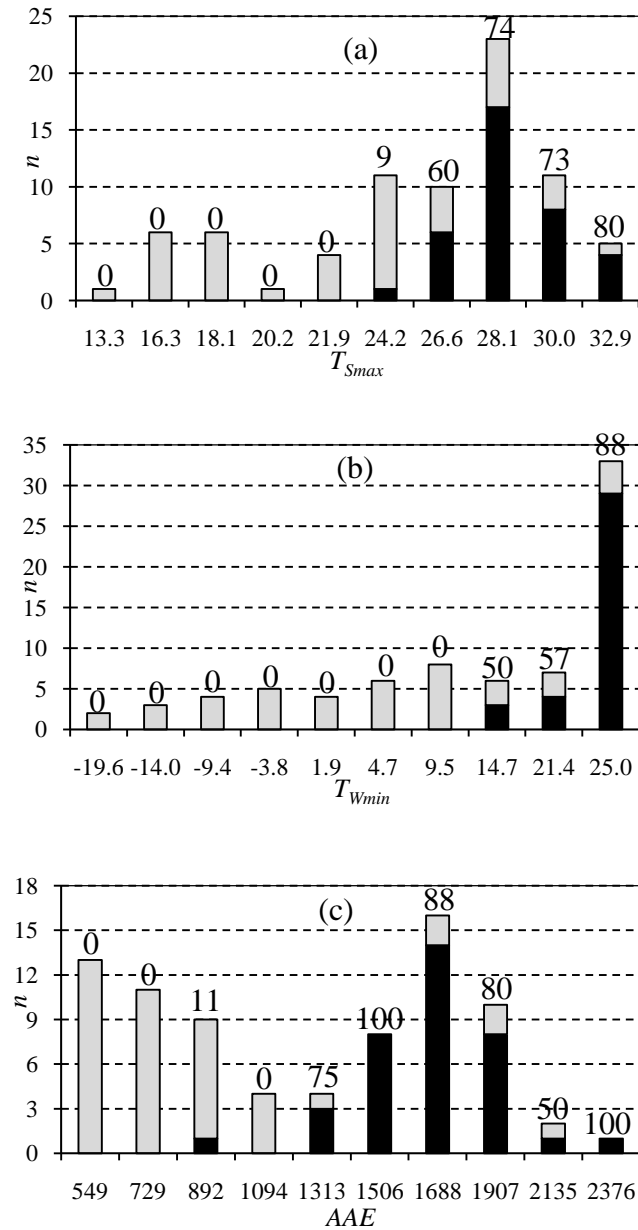


Fig. 6.1. Distributions of the insect *A. hartii* over a range of T_{Smax} (a), T_{Wmin} (b) and AAE (c). Each bar represents the distribution of all locations, where the dark areas represent the number of locations where the insect is present. The number written on top of each bar shows the percentage of all sites occupied by the insect ($r = n_p \div n \%$).

Fig. 6.3 shows the distribution of the insect *X. perforans* over the ranges of T_{Smax} (Fig. 6.3a), T_{Wmin} (Fig. 6.3b) and AAE (Fig. 6.3c). This insect was more often found at sites with higher summer temperatures ($24.8^\circ\text{C} < T_{Smax} < 32.9^\circ\text{C}$), warmer winter temperatures where $T_{Wmin} > 16.1^\circ\text{C}$ and moderate annual evapotranspiration ($241\text{mm} < \text{AAE} < 2004\text{mm}$). The pest insect is present less frequently at the locations where the

annual evapotranspiration is greater than 2040mm. The number of samples in the ranges preferred by this pest insect and in the ranges where the insect pest is mostly absent is balanced across all three climatic variables.

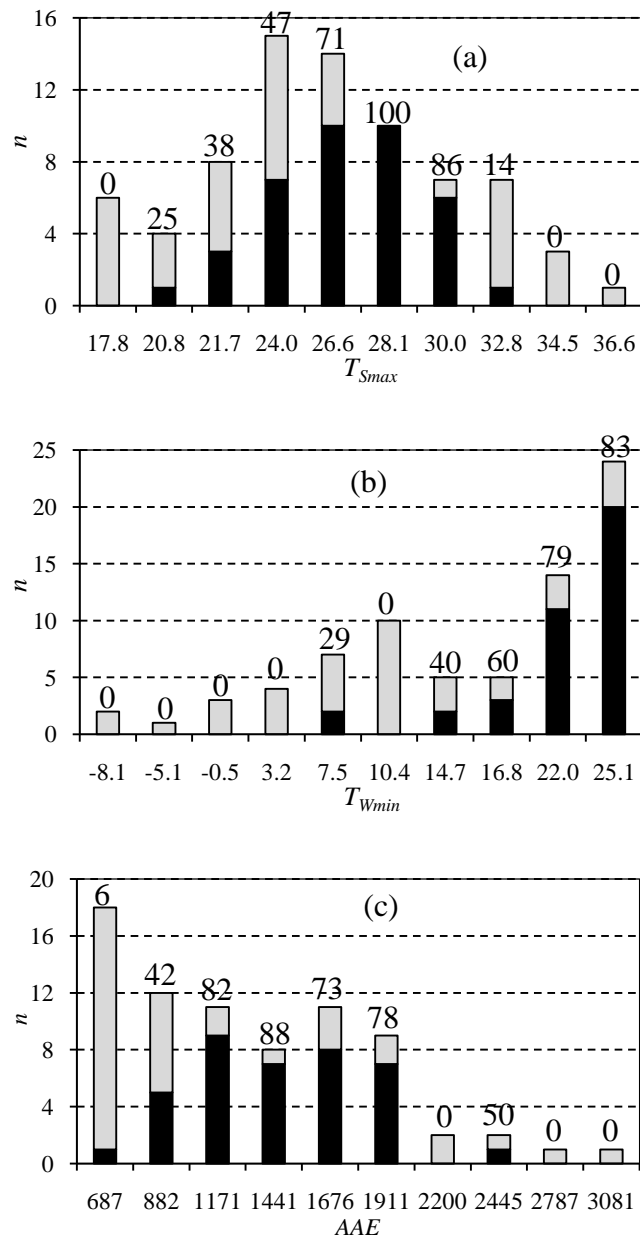


Fig. 6.2. Distributions of the insect *G. coffeae* over a range of T_{Smax} (a), T_{Wmin} (b) and AAE (c). Each bar represents the distribution of all locations, where the dark areas represent the number of locations where the insect is present. The number written on top of each bar shows the percentage of all sites occupied by the insect ($r = n_p \div n \%$).

Finally, it must be noted that the species-habitat relationships of the pest insects used in this thesis are explained solely on the basis of the data samples from the available datasets and therefore the explanation completely depends on the sampling method employed to collect these data samples.

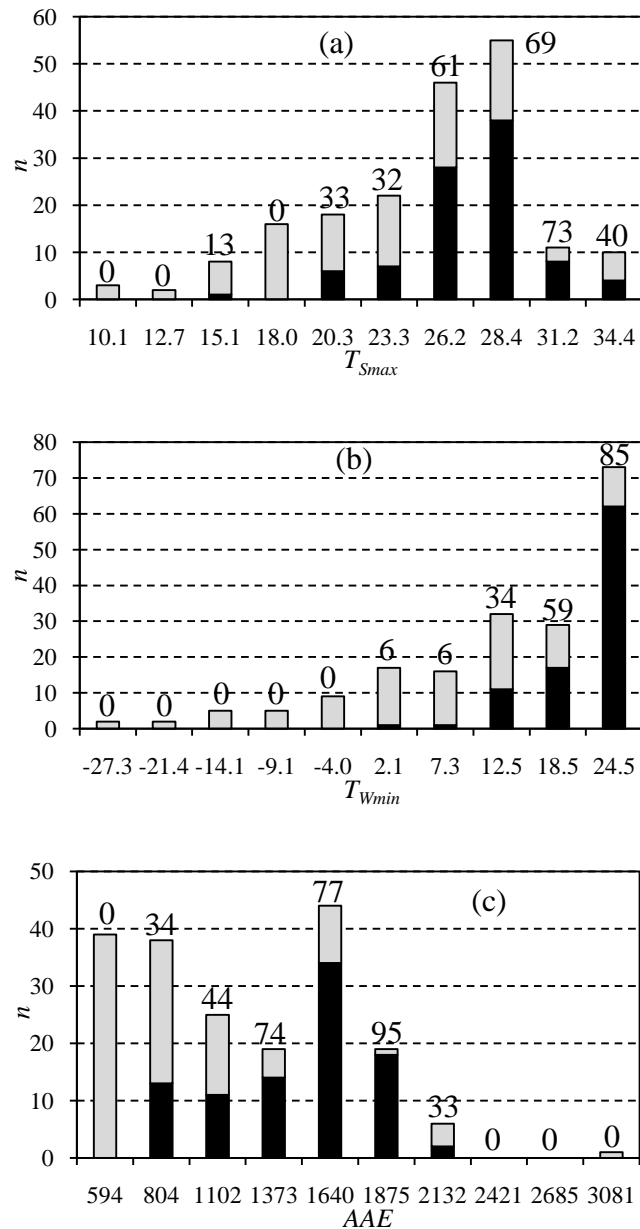


Fig. 6.3. Distributions of the insect *X. perforans* over a range of T_{Smax} (a), T_{Wmin} (b) and AAE (c). Each bar represents the distribution of all locations, where the dark areas represent the number of locations where the insect is present. The number written on top of each bar shows the percentage of all sites occupied by the insect ($r = n_p \div n \%$).

6.3 Discussion of results

6.3.1 Model evaluation

We assess the models' discrimination capabilities with four threshold dependent measures: sensitivity (*sen*), specificity (*spe*), Cohen's kappa (κ) and overall accuracy (*a*) as explained in Chapter 3 Section 3.3.3. Leave-one-out cross validation was performed on all three pest insect datasets as explained in Chapter 4 section 4.4. Additionally,

effects of threshold variation on the performance of each model over the range $\theta \in [0.3, 0.9]$ were plotted and evaluated in terms of the *sen*, *spe*, κ and *a*. Since the accuracy of the transductive models depends on the number of nearest samples (n_n) used to build the particular model, the effect of the size of the neighbourhood was explored by evaluating Model 4's performance with different n_n . Only the performances of the best transductive models were compared against the performances of other models.

6.3.2 Evaluation of the *A. hartii* distribution models

To begin with, the graphs shown in Figs. 6.4, 6.5, 6.6 and 6.7 are examined. In these graphs the effect of threshold variation on the performance of each of the four models is illustrated by plotting the sensitivity (*sen*), specificity (*spe*), Cohen's kappa (κ) and overall accuracy (*a*) against the threshold θ . As expected the overall accuracy (*a*) values are between the *sen* and *spe* values. All four models generate predictions that have a value within the range of [0, 1]. These predictions are dichotomised into presences (1) and absences (0) using a threshold value θ . Recall, that predictions (*P*) above or equal to θ are assigned a 1 (presence), while if $P < \theta$ we assign a 0 (absence). When the θ is altered, all threshold dependent measures are affected. To examine in detail the models' behaviour, the performance variables are plotted over the range of threshold values $0.3 < \theta < 0.9$ in steps of 0.1, yielding seven plotted points for each model or 28 points per figure in total.

Fig. 6.4 shows the effect of different thresholds on the performance of Model 1 (the global model). The accuracy *a* is in the range of [0.53 0.90] with the maximum at $\theta_{Optimal} = 0.5$ and $\theta_{Optimal} = 0.6$. At higher thresholds (*i.e.* $\theta = 0.8$ and $\theta = 0.9$) the κ shows 'poor' or 'no agreement' between observed and predicted outcome values. The highest κ_{max} ('very good') was achieved at thresholds $\theta = 0.5$ and $\theta = 0.6$. The highest sensitivity ($sen_{max} = 0.85$) of this model was at $\theta = 0.6$. Both *spe* and *sen* deteriorated down to $sen = 0.33$ and $spe = 0.53$ at $\theta = 0.9$. This deterioration pulled *a* from its maximum value $a_{max} = 0.9$ down to its minimum value $a_{min} = 0.53$ at $\theta = 0.9$. At $\theta = 0.5$ (the typically used θ), $a = 90\%$ and $\kappa = 0.80$ ('very good').

Fig. 6.5 shows the results obtained with Model 2. This approach built the predictive model that exhibits 'very poor' and 'poor' κ at low θ threshold values, *i.e.* $\theta \leq 0.5$. The κ value of 0.77 ('very good') was achieved at $\theta = 0.7$ and then κ exhibits a sharp decline at the higher thresholds, from $\kappa = 0.77$ at $\theta = 0.7$ to $\kappa = 0.47$ at $\theta = 0.9$. The overall accuracy *a* was in the range of [0.53 0.88] with the maximum at $\theta_{Optimal} = 0.7$.

Sensitivity sen was in the range of [0.49 0.82]. This model predicted absence with a very high accuracy, namely the spe was in the range of [0.80 1]. At the typically used threshold $\theta = 0.5$, the accuracy was relatively low $a = 64\%$ and $\kappa = 0.32$ ('poor').

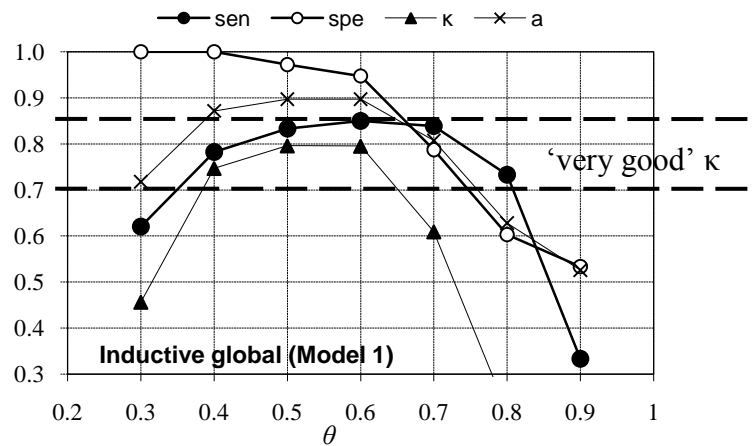


Fig. 6.4. The effect of θ on the performance of the global model built for *A. hartii* expressed in terms of sensitivity (sen), specificity (spe), Cohen's kappa (κ) and overall accuracy (a). The highest $\kappa_{max} = 0.80$ ('very good') and $a_{max} = 90\%$ are at $\theta_{Optimal} = 0.6$ & 0.5 . Note that a θ of 0.5 is typically used.

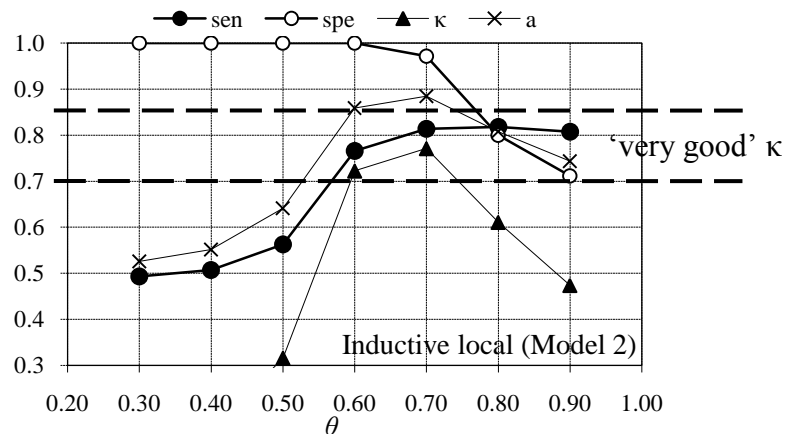


Fig. 6.5. The effect of θ on the performance of the local model built for *A. hartii* expressed in terms of sensitivity (sen), specificity (spe), kappa (κ) and overall accuracy (a). The highest $\kappa_{max} = 0.77$ ('very good') and $a_{max} = 88\%$ are at $\theta_{Optimal} = 0.7$. At $\theta = 0.6$ and $\theta = 0.7$ all measures but the κ values are above 0.7 .

The graphs for Model 3, the LPAM, are shown in Fig. 6.6. It is interesting to see that this model performed better at the lower threshold values ($\theta = 0.3$ and $\theta = 0.4$) and that the κ values decreased for $\theta \geq 0.4$. Its maximum accuracy was 88% . At $\theta = 0.5$, the accuracy was satisfactory $a = 85\%$ and $\kappa = 0.69$ ('good').

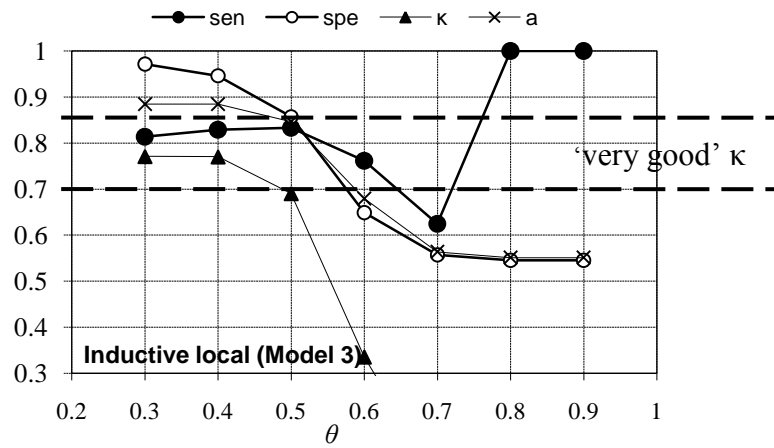


Fig. 6.6. The effect of θ on the performance of the LPAM for *A. hartii* expressed in terms of sensitivity (*sen*), specificity (*spe*), kappa (κ) and overall accuracy (*a*). The highest $\kappa_{max} = 0.77$ ('very good') and $a_{max} = 88\%$ are at $\theta_{Optimal} = 0.3$ & 0.4 . At $\theta = 0.5$, $a = 85\%$.

The graph for Model 4 (Fig. 6.7) shows smaller variations for all four performance indicators at lower threshold values ($\theta \leq 0.7$). In this range the *sen*, *spe*, κ and *a* values were high (*sen* = 0.88, *spe* = 0.97, κ = 0.85 and *a* = 0.92, $\theta = 0.7$), particularly the κ was 'excellent'. However, when $\theta > 0.7$ *spe*, κ and *a* sharply decreased. In contrast, *sen* firstly decreased to *sen* = 0.85 at $\theta = 0.8$ and then bounced back to *sen* = 0.88 at $\theta = 0.9$. At $\theta = 0.5$, $a = 91\%$ and $\kappa = 0.82$ ('very good').

Due to the importance κ has in assessing the performance of predictive ecological distribution models, κ values were plotted at the different probability thresholds θ (Fig. 6.8). The performance of Model 4 (the individual model), when measured in terms of κ , was higher than the performance of the other models. In particular, its κ values at $\theta = 0.5$, 0.6 and 0.7 were 0.82, 0.85 and 0.85, respectively.

Each model achieved a maximum accuracy (a_{max}) at different θ designated by $\theta_{Optimal}$ (Table 6.2). Model 4 was found to be the most accurate for modelling the distribution of *A. hartii*, followed by Model 1, while Model 2 and Model 3 (the local models) were the least accurate.

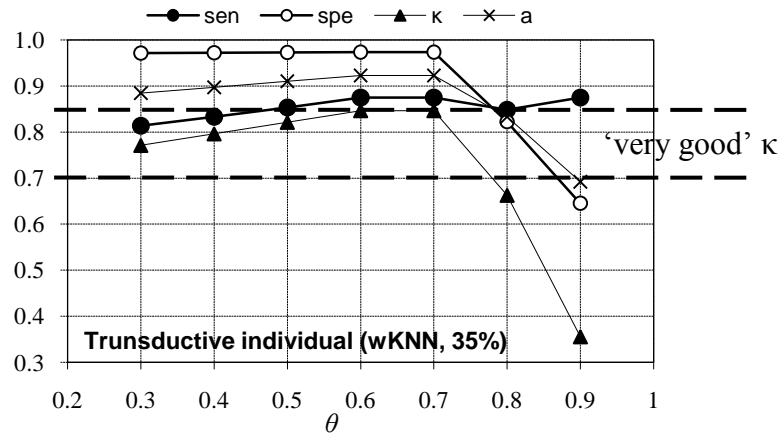


Fig. 6.7. The effect of θ on the performance of individual models built for *A. hartii* expressed in terms of sensitivity (*sen*), specificity (*spe*), kappa (κ) and overall accuracy (*a*). The highest $\kappa_{max} = 0.85$ ('excellent') and $a_{max} = 92\%$ are at $\theta_{Optimal} = 0.6$ & 0.7 .

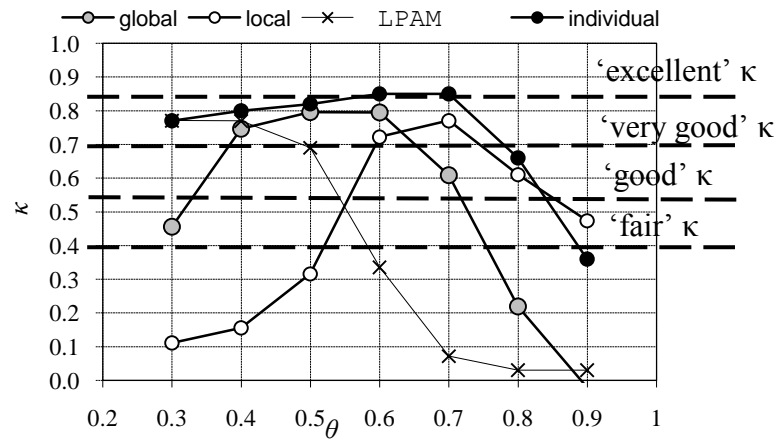


Fig. 6.8. The κ values for the studied models. The performance of transductive model, when measured in terms of κ , is better than the performance of all other models.

TABLE 6.2

$\theta_{Optimal}$ for which each model performed the best when performance was measured in terms of overall accuracy *a*.

Model	$\theta_{Optimal}$	$a_{max}(\%)$
1	0.5, 0.6	90
2	0.7	88
3	0.4	88
4	0.6, 0.7	92

6.3.3 Evaluation of the *G. coffeae* distribution models

The models based on inductive and transductive reasoning were also used for forecasting the establishment potential of *G. coffeae*. The performance of Model 1 is examined in Fig. 6.9. The overall accuracy a was in the range of [0.55 0.88]. The highest $\kappa_{max} = 0.76$ ('very good') and $a_{max} = 88\%$ were achieved at an optimal threshold $\theta_{Optimal}$ equal to 0.8. This model performed better at the higher threshold values ($\theta \geq 0.6$) than at the lower, the accuracy was $a = 67\%$ and $\kappa = 0.33$ ('poor') at $\theta = 0.5$.

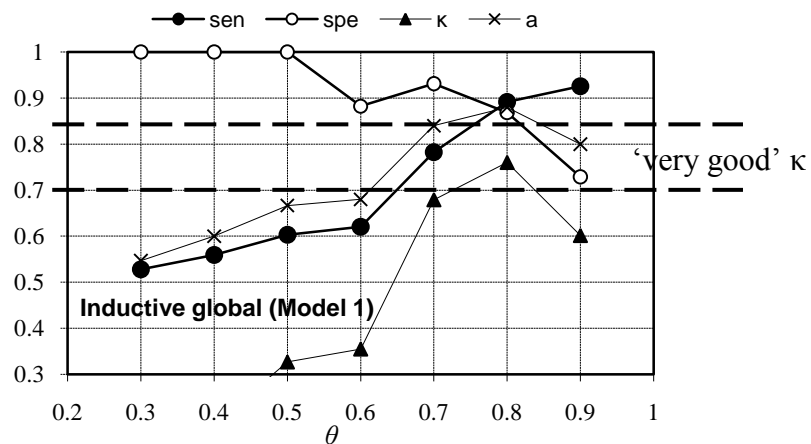


Fig. 6.9. The effect of θ on the performance of the global model built for *G. coffeae* expressed in terms of sensitivity (*sen*), specificity (*spe*), Cohen's kappa (κ) and overall accuracy (*a*). The highest $\kappa_{max} = 0.76$ ('very good') and $a_{max} = 88\%$ are at $\theta_{Optimal} = 0.8$.

Model 2 also gave better results at the higher thresholds (Fig. 6.10). Its κ_{max} and a_{max} were higher than Model 1's. The overall accuracy a was in the range of [0.56 0.91]. At a θ equal to 0.7, the κ_{max} is 0.81 and a_{max} is 91%. At $\theta = 0.5$ the overall accuracy a was slightly higher ($a = 68\%$) than the Model 1's accuracy. At $\theta = 0.5$, the κ value was 0.35 ('poor').

Graphs for Model 3 and Model 4 are given in Fig. 6.11 and Fig. 6.12, respectively. Model 3 had the best performance at $\theta = 0.8$ ($\kappa_{max} = 0.71$ and $a_{max} = 85\%$). Its overall accuracy a was in the range of [0.56 0.85]. At $\theta = 0.5$, the accuracy values ($a = 63\%$ and $\kappa = 0.25$). Model 4 performed better at the lower threshold values ($\theta \leq 0.6$). Its overall accuracy a was in the range of [0.49 0.76]. The highest $\kappa_{max} = 0.52$ ('fair') and $a_{max} = 76\%$ were achieved at thresholds $\theta = 0.4$ & 0.5. Model 4's accuracy at $\theta = 0.5$ was the highest of all studied models.

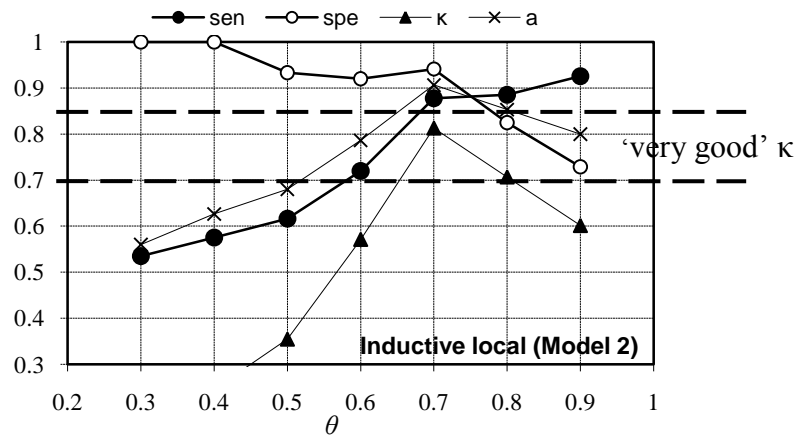


Fig. 6.10. The effect of θ on the performance of the local model built for *G. coffeae* expressed in terms of sensitivity (*sen*), specificity (*spe*), kappa (κ) and overall accuracy (*a*). The highest $\kappa_{max} = 0.81$ ('very good') and $a_{max} = 91\%$ are at $\theta_{Optimal} = 0.7$.

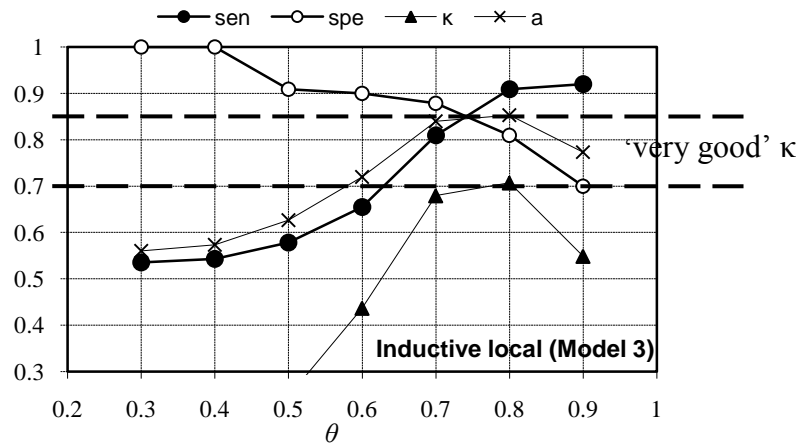


Fig. 6.11. The effect of θ on the performance of the LPAM built for *G. coffeae* expressed in terms of sensitivity (*sen*), specificity (*spe*), kappa (κ) and overall accuracy (*a*). The highest $\kappa_{max} = 0.71$ ('very good') and $a_{max} = 85\%$ are at $\theta_{Optimal} = 0.8$.

Fig. 6.13 reveals that the κ of the transductive model had quite different behaviour than the κ of the other models. While the κ of the other models was higher at larger θ , the κ of the transductive model decreased at the higher values of θ . In particular, at $\theta = 0.5$ the accuracy of Model 4 was the highest in terms of κ . But then at $\theta = 0.7$ this changed and Model 4 became the worst performing model. Overall, all the models had more problems with the predicting establishment potential for *G. coffeae* than for *A. hartii*.

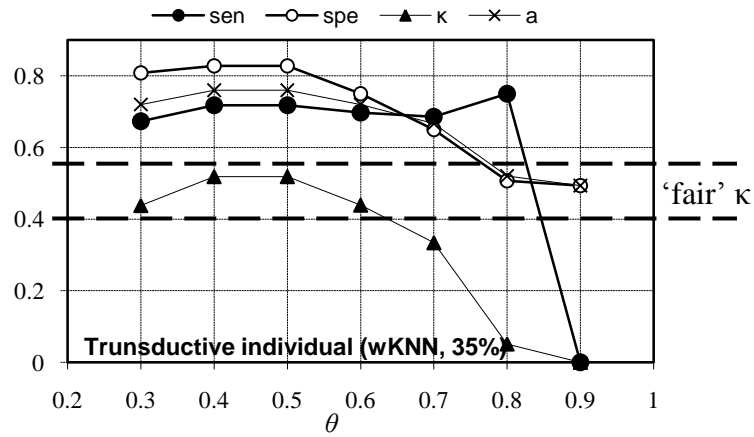


Fig. 6.12. The effect of θ on the performance of individual models built for *G. coffeae* expressed in terms of sensitivity (*sen*), specificity (*spe*), kappa (κ) and overall accuracy (*a*). The highest $\kappa_{max} = 0.52$ ('fair') is at thresholds $\theta = 0.4$ & 0.5 , $a_{max} = 76\%$ is at $\theta_{Optimal} = 0.5$.

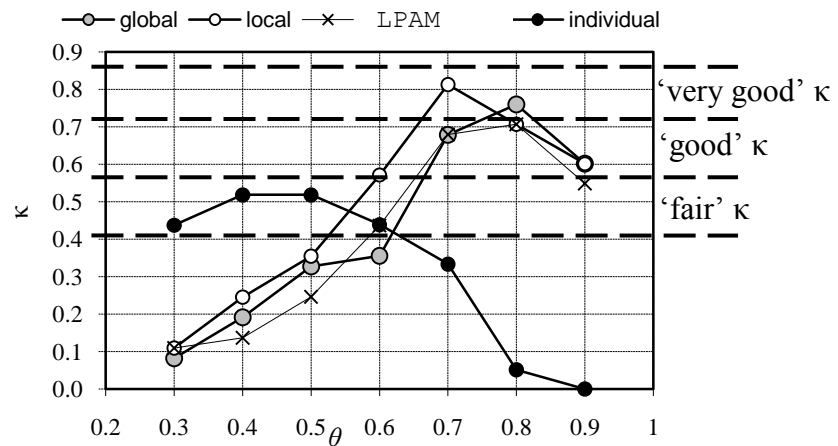


Fig. 6.13. The κ values for the studied models. The performance of the transductive model exhibits a different trend than performances of the other three models.

6.3.4 Evaluation of the *X. perforans* distribution models

All four distribution models for *X. perforans* were less accurate, in terms of κ , than the models built for the other two species. This can be seen exploring plots in Figs. 6.14, 6.15, 6.16, and 6.17. Fig. 6.14 shows plots for Model 1. The overall accuracy *a* of this model is in the range of [0.51 0.81]. The best κ_{max} and a_{max} values are equal to 0.62 ('good') and 81%, respectively, while at $\theta = 0.5$ the accuracy was considerably lower ($\kappa = 0.46$, $a = 73\%$). Model 2's *a* is in the range of [0.51 0.81] (Fig. 6.15). The maximum values were achieved at $\theta_{Optimal} = 0.5$ (i.e. $a_{max} = 81\%$ and $\kappa_{max} = 0.61$ ('good')). Model 3 (Fig. 6.16) had $a_{max} = 77\%$ and the *a* values were in the range of [0.50 0.77]. Model 4

(Fig. 6.17) had the same maximum accuracy $a_{max} = 77\%$ but slightly different κ ($\kappa_{maxModel3} = 0.55$, $\kappa_{maxModel4} = 0.54$). Its overall accuracy a is in the range of $[0.52 \ 0.77]$. Both models achieved a ‘fair’ accuracy at $\theta = 0.5$ ($a_{Model3} = 75\%$ and $\kappa_{Model3} = 0.50$, $a_{Model4} = 77\%$ and $\kappa_{Model4} = 0.54$).

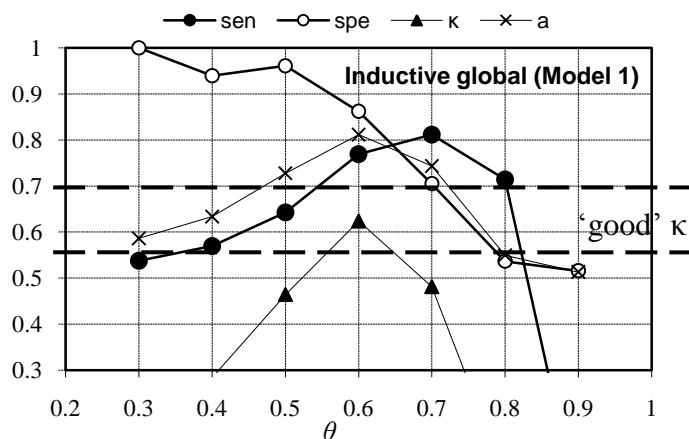


Fig. 6.14. The effect of θ on the performance of the global model built for *X. perforans* expressed in terms of sensitivity (*sen*), specificity (*spe*), Cohen’s kappa (κ) and overall accuracy (*a*). The highest $\kappa_{max} = 0.62$ (‘good’) and $a_{max} = 81\%$ are at $\theta_{Optimal} = 0.6$.

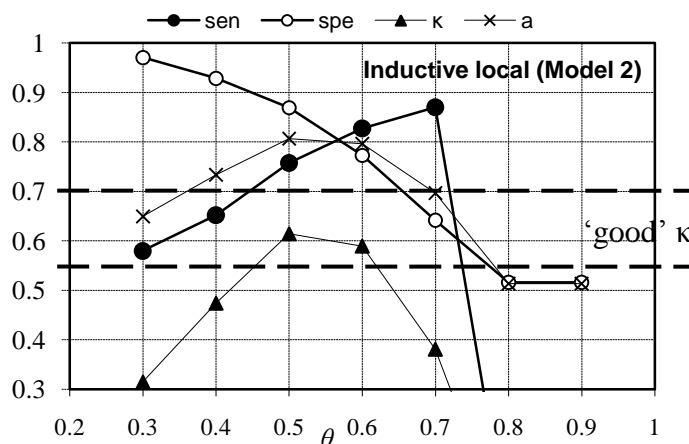


Fig. 6.15. The effect of θ on the performance of the local model built for *X. perforans* expressed in terms of sensitivity (*sen*), specificity (*spe*), kappa (κ) and overall accuracy (*a*). The highest $\kappa_{max} = 0.61$ (‘good’) and $a_{max} = 81\%$ are at $\theta_{Optimal} = 0.5$.

None of the models had κ in the ‘very good’ range (Fig. 6.18). However, all κ plots show the same behaviour. They are lower at the lower θ values and increase as θ increases, reaching a maximum at either $\theta = 0.5$ or $\theta = 0.6$ and then sharply decrease. When $\theta = 0.5$ Model 2 is the best performing model, followed by Model 4, Model 3 and Model 1. However, at $\theta = 0.6$ the order changes. Model 1 becomes the best performing,

followed by Model 2, Model 3 and Model 4. Model 4 was the best performing model for $\theta \geq 0.7$.

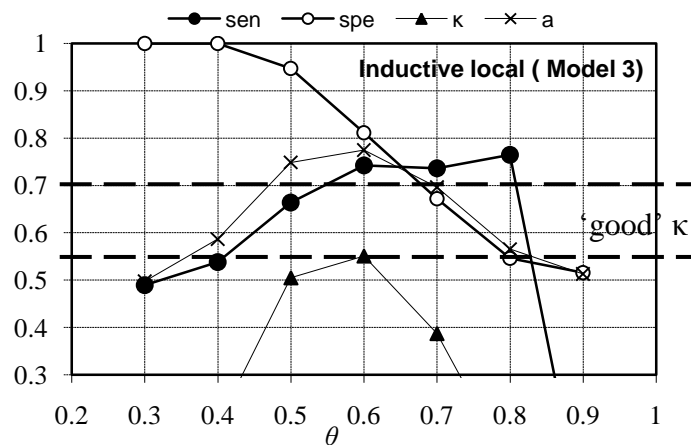


Fig. 6.16. The effect of θ on the performance of the LPAM built for *X. perforans* expressed in terms of sensitivity (*sen*), specificity (*spe*), kappa (κ) and overall accuracy (*a*). The highest $\kappa_{max} = 0.55$ ('good') and $a_{max} = 77\%$, are at $\theta_{Optimal} = 0.6$.

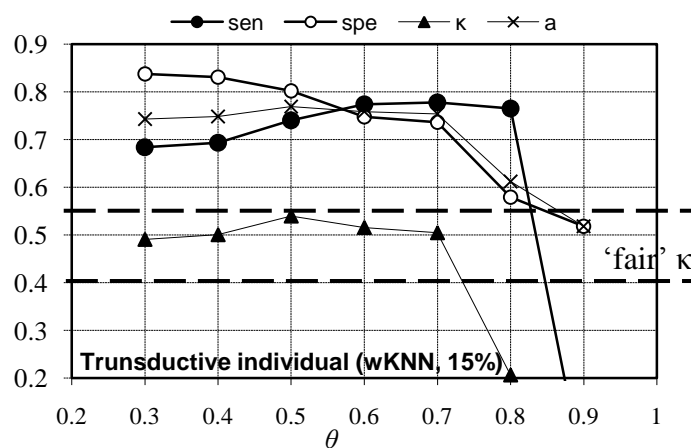


Fig. 6.17. The effect of θ on the performance of individual models built for *X. perforans* expressed in terms of sensitivity (*sen*), specificity (*spe*), kappa (κ) and overall accuracy (*a*). The highest $\kappa_{max} = 0.54$ ('fair') and $a_{max} = 77\%$, are at $\theta_{Optimal} = 0.5$. Note: 15% samples have been used to create the model.

6.3.5 Comparative analysis of the four approaches when used for predicting the establishment potential of three different pest insects

The bar charts in Fig. 6.19 summarise the maximum accuracy of the four models when used for assessing the establishment potential of *A. hartii* (AH), *G. coffeae* (GC) and *X. perforans* (XP). The three sets of bars on the left, one for each insect, show the models'

accuracy in terms of a while the three sets on the right show the accuracy in the terms of κ . Recall that the maximum values were achieved at different threshold values ($\theta_{Optimal}$). Clearly, the threshold value θ must be optimised to attain the best results.

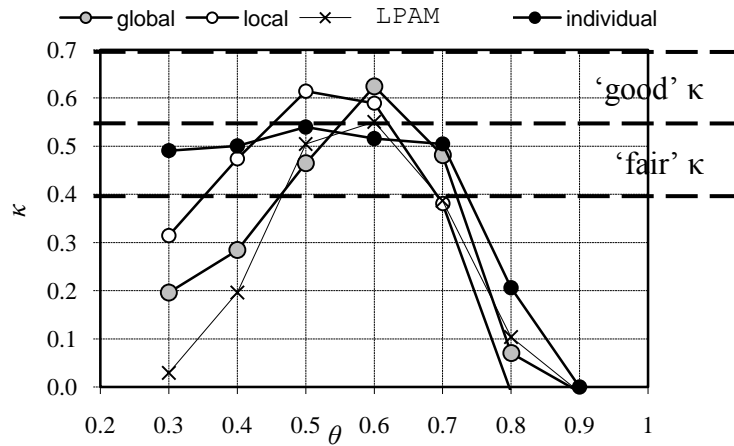


Fig. 6.18. The κ values for the studied models. None of the models achieved ‘very good’ κ values.

While transductive reasoning was beneficial for building predictive model for *A. hartii*, the same approach delivered less accurate predictions for the other two species. The model based on DENFIS was the most accurate for *G. coffeae* and the global model was winner for *X. perforans*. As mentioned earlier, all the models had problems modelling the distribution of *X. perforans*. The importance of θ on modelling accuracy must be highlighted and it is clear that the comparative analysis would have had different results if the models were assessed using different threshold values.

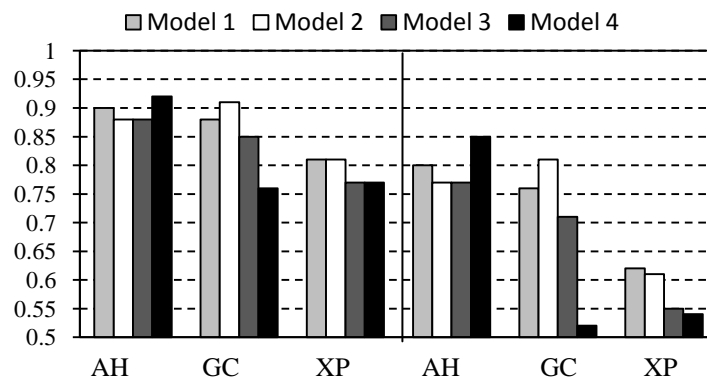


Fig. 6.19. The maximum accuracies of the studied models in terms of a and κ values. Note that the maximum values were achieved at different $\theta_{Optimal}$.

6.3.6 Knowledge discovery for different modelling techniques

Predictive models can contribute to our understanding of the factors that control patterns of species distribution. In order to investigate the knowledge discovered through the use of the different modelling techniques, the global, local and ‘personalised’ models were used to assess the suitability of Bombay (India) and Wellington (New Zealand) for establishment by the insect *A. hartii* and to detail the insect-habitat relationships. This insect is known to be present in Bombay, while it is considered absent in Wellington. For this purpose, these two locations were removed from the original data set D and the remainder of the data was used to create the distribution models. It can be seen in Table 6.3 that all models predicted a low risk of these insects establishing at the New Zealand location and high risk of establishment at Bombay. Hence, the models correctly predicted the establishment potential at both locations.

Assessing the quality of the knowledge discovered by the use of different modelling techniques is even less straightforward. Proving that one model better describes species-habitat relationships than the other models is often impossible, especially when there is little or no available information about the species’ habitat preferences. The true relationships are generally impossible to obtain for insect species for the reasons explained in Chapter 3 Section 3.3. One solution would be to release a pest insect into a new location to see if it would survive. This approach is risky and not used very often. Or, a controlled environment could be built where the climate variables would be controlled and their influences on the pest observed. This approach is either expensive or impractical, or both. Therefore, due to the lack of knowledge on how the studied pest insects are influenced by their surrounding climate, we can only make suggestions. These suggestions must be further studies by biosecurity scientists and biologists.

First we examine the knowledge discovered using the global model. According to Model 1, the global model in this study, the establishment potential of the insect *A. hartii* increases with AAE and T_{Wmin} , but decreases with T_{Smax} . The influence of AAE is negligible ($a_{AAE} = 0.00057$) compared to the influences of summer ($a_{TSmax} = -0.02$) and winter ($a_{TWmin} = 0.01$) air temperatures:

$$y_{MI} = 0.12 + 0.00057 AAE - 0.02 T_{Smax} + 0.01 T_{Wmin}.$$

The negative relationship between the species and T_{Smax} is questionable because according to Fig. 6.1(a), the percentage of all sites occupied by this pest insect increases at the locations with higher T_{Smax} values.

TABLE 6.3

The prediction results obtained using the different models for Bombay, where the species is established (1) and Wellington where the species has not yet established (0).

Model	Bombay	Wellington
1	0.767	0.259
2	0.810	0.113
3	0.765	0.039
4	0.973	0

Local modelling, using Model 2, discovered two clusters of locations (two local models, LM₁ and LM₂) both with a positive relationship between the insect's distribution and T_{Wmin} . Intuitively, having two models explaining the pest insect's worldwide distribution is a more sensible approach than explaining all the locations with only one model. Fig. 6.20 shows the contributions (based on the regression coefficients) of selected predictors in each of the two rules. LM₁ is applicable for warmer locations with average *AAE* values. At those locations, the likelihood of the insect's establishment has a positive relationship with all three climatic variables, with T_{Wmin} being the main contributor:

$$y_{LM1} = 0.14 + 0.26 AAE + 0.08 T_{Smax} + 1.57 T_{Wmin}.$$

LM₂ describes the establishment potential at cooler locations with lower *AAE* values, at which the likelihood of the insect's establishment increases with warmer T_{Wmin} , but decreases slightly with T_{Smax} and *AAE*. The contribution of T_{Wmin} is an order higher than the negative contributions of the other two climatic variables:

$$y_{LM2} = 1.00 - 0.03 AAE - 0.05 T_{Smax} + 0.21 T_{Wmin}.$$

As said earlier, the performance of the 'personalised' model (Model 4) is dependent on the number of neighbouring samples k used during the modelling stage, a parameter that is not known in advance and must be optimised. If the number of samples is too low the model can be overfitting and if the number is too high we can get a poorly performing

model. The influence of k on the predictions of Model 4 is shown in Fig. 6.21, where the likelihood of the insect establishment at two locations (Bombay and Wellington) is plotted against the number of neighbouring samples (k) used during the modelling. Each plot includes the range of k values investigated, from 5% to 45%, in steps of 5%, where k is expressed as a percentage of the total number of samples available for modelling. Note that there are nine points for each locality (Bombay (●) and Wellington (○)). Although, the predicted establishment potential at Bombay varies with k ($P_{min} = 0.84$, $P_{max} = 0.97$), these variations are not significant. The establishment potential of this species is found to be 0 at Wellington for all k from $k = 5\%$ to $k = 40\%$ and 0.05 for $k = 45\%$.

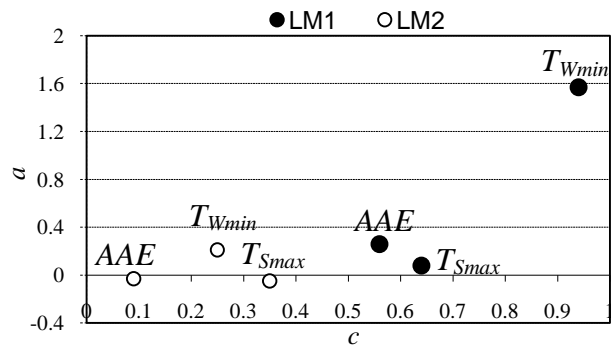


Fig. 6.20. The contributions of predictor variables in two different fuzzy rules. Each rule explains the influence of three climate characteristics on the establishment potential of *A. hartii*. LM₁: higher T_{Smax} , T_{Wmin} and $AAE \Rightarrow$ higher response, the main positive contributor is $a_{T_{Wmin}} = 1.57$. LM₂: higher $T_{Wmin} \Rightarrow$ higher response, the main positive contributor is $a_{T_{Wmin}} = 0.21$.

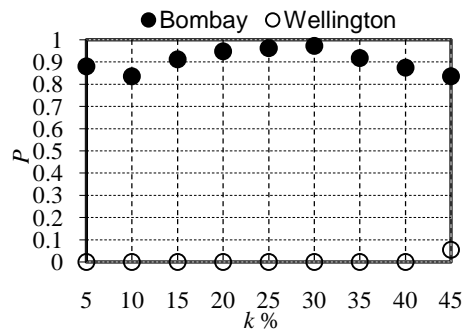


Fig. 6.21. The influence of the number of neighbouring samples k on the accuracy of predicting establishment potential of *A. hartii* using transductive reasoning.

6.3.7 Comment on the knowledge discovery using the ‘personalised’ modelling

The one disadvantage of the transductive approach used in this thesis is that it delivers the predictions for pest distributions but with no clear explanation of how these predictions were made. For each data sample $X_i \in D$ a ‘personalised’ prediction y_i is calculated as a weighted average of the output values y_k of k most similar data samples to X_i , X_k form a sub-training dataset D_{TRi} for X_i . The similarity between data samples is calculated in their feature space using (6.1), where d is the dimensionality of data samples, *i.e.* the number of features, and x_{if} and x_{kf} are values of the f -th feature of the i -th and k -th data samples.

$$\|X_i - X_k\| = \sqrt{\sum_{f=1}^d (x_{if} - x_{kf})^2} \quad (6.1)$$

The calculation of y_i is rather simple. Given a problem space shown in Fig. 6.22, y_i is calculated as

$$y_i = \frac{\sum_{k=1}^{12} w_k y_k}{\sum_{k=1}^{12} y_k} \quad (6.2)$$

where weight $w_1 > w_2 > \dots > w_{12}$ represent the importance of the X_k samples to the output value of the X_i sample.

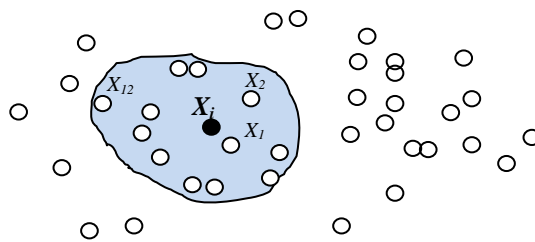


Fig. 6.22. A hypothetical problem space D . The output value for X_i is calculated as a weighted average of the output values of the X_1, X_2, \dots, X_{12} data samples.

Using (6.2) ‘personalised’ rankings of the importance for each X_k sample can be derived for any new X_i . However, no explicit knowledge of the variable importance is available in these ‘personalised’ models. This is an obvious disadvantage of using (6.2).

However, once a D_{TRi} sub-training dataset is identified a learning system capable of knowledge discovery and representation can be applied to derive a piece of ‘personalised’ knowledge, M_{ij} , for each new X_i (Fig. 6.23). Having this knowledge, the invasive predisposition of specific geographical regions, X_i , could be evaluated.

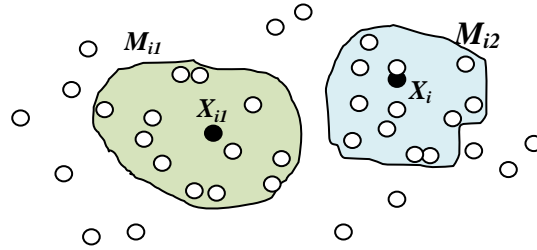


Fig. 6.23. A piece of ‘personalised’ knowledge – a ‘personalised’ model M_{ij} is created for each new data sample.

6.4 Conclusion

The purpose of this investigation was to assess the applicability of transductive reasoning for forecasting the establishment potential of pest insects into new locations. For this purpose four types of predictive models were used to predict the distributions of three pest insects that are of potential economic importance to New Zealand. Model 1 was a global model, Model 2 and 3 were local models and Model 4 was built using (2.3) and (2.4) given in Chapter 2 and transductive reasoning. The models were assessed using leave-one-out cross validation and Cohen’s kappa over the range of threshold values, from 0.3 to 0.9. The models’ ability to discover knowledge from data was also discussed. The assessment of the results was based on the important assumption that the modelling data sets are the real representation of the pest insects’ distributions under the climatic conditions.

Solely on the criteria of correctly predicting the presence or absence of *A. hartii*, Model 4 performed better than all the other models. Its $\kappa = 0.85$ (‘excellent’) is the highest of all the models’ κ values in the $\theta < 0.8$ range. The same model predicted the distribution of *G. coffeae* with the highest accuracy ($\kappa = 0.52$, ‘fair’ κ) for $\theta = 0.5$, but its performance deteriorated at higher thresholds. Other models have κ in the ‘poor’ range at the same θ . However, Model 4 was the worst performing model at thresholds $\theta \geq 0.7$. We also found that all the models had with problems predicting the distribution of *X. perforans*. The highest $\kappa_{max} = 0.62$ (‘good’) was achieved at $\theta_{Optimal} = 0.6$ by Model 1. Model 4’s highest $\kappa_{max} = 0.54$ (‘fair’) at $\theta_{Optimal} = 0.6$.

Overall the performance of all the models was very similar and it is hard to suggest one technique that is better than the others. If ecologists are interested in studying the influence of the contributions of each explanatory variable to the output response then the models based on DENFIS (Model 2 and 3) are preferable due to their capability to deliver a set of rules describing species-habitat relationships.

Unfortunately, in ecological modelling the ‘true’ model is often not available to assess the validity of a model. Furthermore, eco-climatic data is prone to noise and interactions are complex and nonlinear. Therefore, a number of different techniques must be used, their results carefully compared and their validity evaluated in the predefined context of their applications. Due to the huge number of species and diversities of possible habitats, the models that are appropriate for one species may be totally inappropriate for others, and comparing results across the discipline can potentially lead to misleading conclusions.

Chapter 7

Spiking neural networks for evolving connectionist systems – a review

Many real-world applications, including applications in ecology and bio-security, require data mining in the temporal domain. It is not easy to explore temporal patterns using neural networks with traditional rate-coded neurons. Therefore, a different approach to temporal data analysis must be considered. A more realistic and biologically plausible network of neurons can be used. Chapter 7 deals with the third generation of artificial neurons, *i.e.* spiking neurons. Firstly, it provides the advantages of using spiking neurons rather than the more familiar rate coded traditional neurons.

Secondly, this chapter provides an introduction to the main characteristics and dynamics of biological neurons, along with a survey of some popular computational models. The survey begins by looking at the Hodgkin and Huxley's description of the influences of conductance of three ion channels on the spike activity of the giant axon of squid. Following this there is a review of threshold models, which includes a review of integrate-and-fire models. The survey of computational models finishes with the Izhikevich's model, which combines the biological plausibility of the Hodgkin-Huxley model with the computational efficiency of the integrate-and-fire models.

Thirdly, we discuss the encoding of analogue values. The real life modelling data is often obtained through various sensors that measure steady-state values and those values must be represented internally for spiking neurons. In other words, the continuous values must be changed to temporal spike patterns.

Overall, this chapter gives the support and background to the contributions presented in Chapters 8 and 9 where an evolving spiking neural network is proposed and then used for taste recognition and an FPGA implementation of this same network is introduced and compared to its software implementation.

7.1 Why to use spiking neurons and spiking neural network models?

Spiking neural networks (SNN) are comprised of spiking neurons that are more biologically plausible than their predecessors. Although all artificial neural networks mimic the human brain with very simplified models of biological neurons, the spiking neurons differ greatly from their predecessors in the way that they convey information between them. The spiking neurons communicate using pulses or spikes, *i.e.* discrete voltage events within a continuous time period, which is similar to real neurons. The spiking neurons receive individual spikes from their afferent (pre-synaptic) neurons and generate spikes by themselves. Because these spikes have a very similar form (amplitude and duration) their shapes are considered insignificant in the transfer of information and only the firing times, *i.e.* timing of the spikes, are thought important for information processing. Contrary, the traditional neurons, which we discuss in Chapter 2, use rate codes, continuous signals with amplitudes that typically fall within a range of 0 and 1, where higher amplitudes correspond to higher average firing rates.

For decades, rate coding has been used as the coding scheme for encoding input data for artificial neural networks. The use of rate codes is in line with the findings that cortical neurons transmit information as rate codes in ensembles of 50-100 neurons (Shadlen & Newsome, 1998). However, there is experimental evidence that suggests that neurons generate high resolution temporal patterns (Wagner *et al.*, 2005; Tetko & Villa, 2001; Prut *et al.*, 1998; Buračas *et al.*, 1998; Softky, 1995; Abeles *et al.*, 1993) and therefore representing them with rather simple rate codes might be too simplistic. Furthermore, some researchers argue that rate coding is too slow to model speeds with which sensory systems can operate (Gautrais & Thorpe, 1998). As a result, pulse coding was found to be more appropriate than the slower rate coding for modelling fast processing modalities (Thorpe, Delorme & Van Rullen, 2001). Particularly, the importance of the first post-stimulus spike for cortical coding of stimulus has been highlighted (Furukawa, Xu & Middlebrooks, 2000; Oram & Perrett, 1992). An experiment on the sensory cortex of rats found that the majority of the information was contained in the timing of the first spike, and the remaining information was encoded within the spike patterns (Peterson, Panzers & Diamond, 2002; Peterson, Panzari & Diamond, 2001). Others are also in favour of temporal coding that relies on the timing of a single spike (Bothe, Poutré & Kok, 2002; Maass, 1997a & 1997b; Gerstner *et al.*, 1996). While neural coding is still a topic of debate, more and more evidence has been collected that shows

that brain processing must rely, to some extent, on the time structures of the spike patterns. A survey of recent experimental discoveries on coding in neural systems can be found in (Van Rullen, Guyonneau & Thorpe, 2005) and (Bothe, 2004). Given that the exact encoding schemes utilised by the human brain are unknown, a number of encoding schemes need to be employed to represent the different aspects of a stimulus variable (Van Rullen, Guyonneau & Thorpe, 2005).

It has been recognised that spike trains can carry more information than rate codes (McClurkin *et al.*, 1991; Stein, 1967). Furthermore, networks of spiking neurons have been proven to be more computationally powerful for the same number of neurons than the networks built from traditional neurons (Maass, 1997b). Sejnowski (1995) questioned if it is time to start using a new neural code. Many researchers agreed that it is. However, while the ability to learn temporally encoded patterns is one advantage of spiking neurons, particularly in tasks where the time component is available, they are typically computationally more complex than their predecessors.

There are two distinctive groups of applications of networks of spiking neurons:

1. SNN are used for modelling brain functions with the purpose of better understanding and reproducing the spike processing in the brain (Vogelstein *et al.*, 2007; Iannella & Kindermann, 2005, Kasabov, Benuskova & Wysoski, 2005, Knoblauch, 2005),
2. SNN are used as a tool for information processing, particularly for image processing (Wysoski, Benuskova & Kasabov, 2006; Thorpe, Delorme & Van Rullen, 2001; Delorme *et al.*, 1999; Van Rullen *et al.*, 1998), olfactory processing (Ambard & Martinez, 2006; Allen, Abdel-Aty-Zohdy & Ewing, 2005; Brody & Hopfield, 2003; Allen *et al.*, 2002; White & Kauer, 1999), speech recognition (Wysoski, Benuskova & Kasabov, 2007b; Verstraeten *et al.*, 2005) and in robotics (Alnajjar & Murase, 2006; Kubota & Sasaki, 2004; Nielsen & Lund, 2003). There are examples of SNN being applied to benchmark datasets to perform supervised classification, (Belatreche, Maguire & McGinnity, 2007; Wu *et al.*, 2006; Booi & Nguyen, 2005), function approximation (Wu *et al.*, 2006) and time series prediction (Sohn, Zhang & Kaang, 1999).

The work in this thesis belongs to the second group of applications of SNN, that is, we use spiking neurons for the purpose of modelling human taste recognition. As will be discussed later on in Chapter 8, the gustatory system is one of the less understood

human sensory systems. Therefore, an attempt to build a biologically plausible model for this system would be naive. Instead, a very simple model of taste recognition was build to show that spiking neurons are a promising new solution for building approximations of the taste recognition system. Below is a brief introduction to various artificial models of biological neurons.

7.2 Spiking neural models: a review

7.2.1 A brief introduction to natural neurons and neural networks

In order to understand how spiking neural models simplify the biological neuron the physiology of a generic cortex neuron needs to be explained. Fig. 7.1 shows an approximation of the structure of a tiny portion of the mammalian cortex. Each neuron consists of three functional parts; namely a number of dendrites, a soma and an axon. Given that the soma is a processing unit, its associated axon is the output where its own spikes are released to other neurons and the dendrites are the inputs where the incoming spikes are received. The junction ‘points’, shown as black circles, denote synapses, *i.e.* the area where an axon and a dendrite meet.

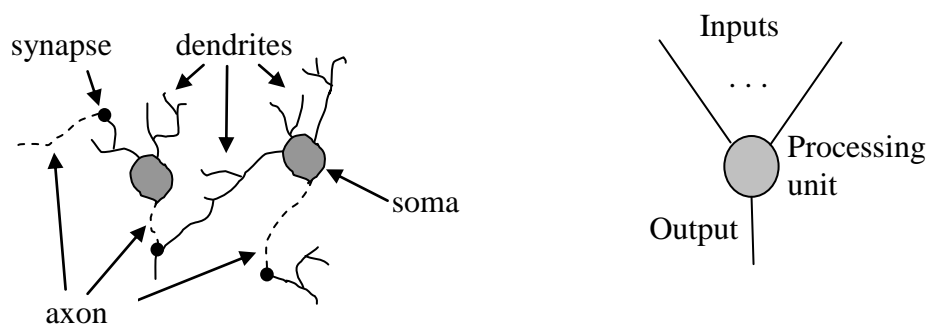


Fig. 7.1. On the left, an approximation of the structure of a tiny portion of the mammalian cortex is shown. See the text for the description of each functional part. In reality, the neurons are numerous and very closely packed. On the right, the schematic diagram of the soma as a processing unit with a number of inputs and one output is given.

The synapse is a highly complex part of the neuron. It is responsible for passing the spikes (Fig. 7.2) that originated at a pre-synaptic neuron (the source of a spike) to a post-synaptic neuron (the destination of a spike). The synapses are influenced by the transmitted spikes, resulting in an increase or decrease in their synaptic strength (Zucker

& Regehr, 2002; Abbott & Nelson, 2000; Kistler & van Hemmen, 1999). As a result, the amplitudes of the incoming spikes are modulated accordingly. The neurons are also dynamic. Their membrane potentials change under the influence of incoming pre-synaptic spikes. The arrival of a spike causes a chain of bio-chemical reactions which result in a change in the membrane potential (PSP , post-synaptic potential) of the post-synaptic neuron (Gerstner & Kistler, 2002). The change is positive ($EPSP$, excitatory post-synaptic potential) if the spike comes via an excitatory synapse or negative ($IPSP$, inhibitory post-synaptic potential) if the synapse is inhibitory (Fig. 7.3). The time it takes for a spike to arrive from a pre-synaptic neuron j is denoted by t_j^f . As the spikes arrive from the pre-synaptic neurons, the post-synaptic potentials of a neuron add up and if the total value exceeds some threshold value PSP_θ the post-synaptic neuron itself generates a spike. This spike is propagated along the axon to other post-synaptic neurons connected to this firing neuron. Immediately after the post-synaptic spike is generated the neuron enters an absolute refractory period during which no spikes can be generated by the neuron. This is followed by a relative refractory period in which it is harder for the neuron to produce a spike (Vreeken, 2003; Kunkle & Merrigan, 2002; Papoyan, 1997). At the end of the synaptic action the neuron's PSP returns to its resting value (PSP_{rest}) and the neuron is ready to spike again.

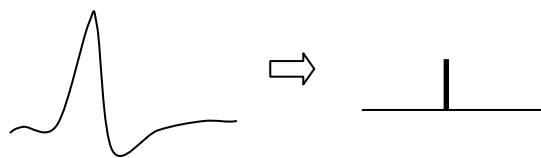


Fig. 7.2. This is a reproduction of the drawing by Gerstner and Kistler (Gerstner & Kistler, pg. 2, 2002) of a neural action potential (or spike) with an amplitude of about 100 mV. The spike's duration is around 1-2 ms. In this thesis the spikes are approximated by a vertical bar, as shown on the right-hand side of this figure.

Fig. 7.4 illustrates a situation where four pre-synaptic spikes arriving via two excitatory synapses cause the membrane potential PSP of the post-synaptic spike to cross the threshold PSP_θ resulting in a post-synaptic spike. Note that the figure is an exaggeration of what actually occurs in reality, *i.e.* four spikes, each causing a voltage increment in the range of 1 mV are not enough to push the membrane potential of the post-synaptic neuron above its PSP_θ value, which is typically at about 20 to 30 mV above PSP_{rest} . Clearly, 20-40 pre-synaptic spikes must arrive via the excitatory synapses in a short period of time in order for a post-synaptic neuron to spike. In addition, the amplitudes

of all voltages in Fig. 7.4 are not in proportion to the actual values measured in biological neurons.

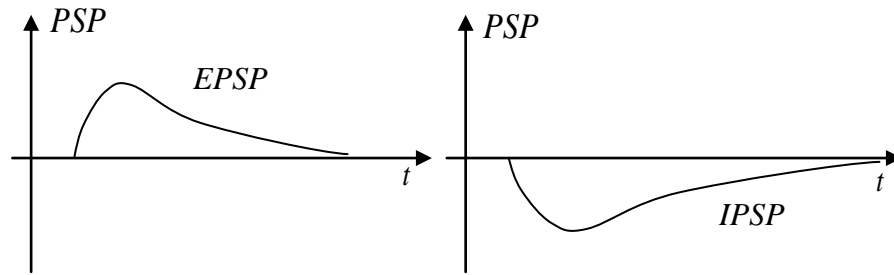


Fig. 7.3. Typical forms of the post-synaptic potential of a neuron as a reaction to a received spike when the neuron is at rest. *EPSP* is caused by a spike arriving via an excitatory synapse and *IPSP* is caused by a spike arriving via an inhibitory synapse.

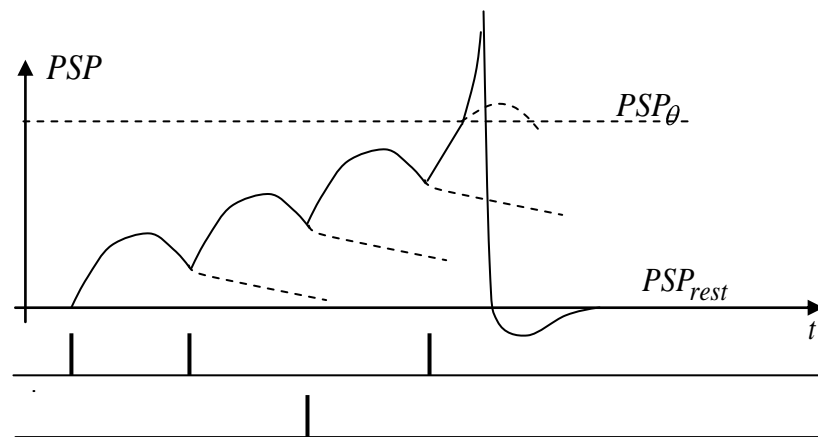


Fig. 7.4. A postsynaptic spike is caused by four pre-synaptic spikes arriving via two excitatory synapses. When the *PSP* crosses PSP_{θ} , the post-synaptic spike is created and propagated along the axon. Note that the figure is an exaggeration of what occurs in reality, *i.e.* four spikes, each causing a voltage increment in the range of 1-2 mV are not enough to push the membrane potential *PSP* of the post-synaptic neuron above PSP_{θ} . In addition, the amplitudes of all voltages are not in proportion to the actual values.

7.2.2 Hodgkin-Huxley model

The Hodgkin-Huxley model is based on the experimental studies by Hodgkin and Huxley. It gives a detailed description of the influences of the conductance of three ion channels on the change of the membrane potential. The model is shown in Fig. 7.5, where the leakage channel is modelled with a parallel resistor-capacitor circuit and the sodium (*Na*) and potassium (*K*) channels are represented by voltage-dependent resistors. The conductances of the channels are represented by G_{Na} , G_k and G_L .

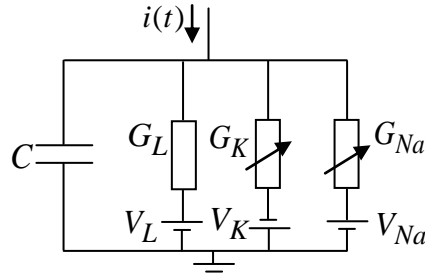


Fig. 7.5. The Hodgkin-Huxley model: schematic diagram (Gerstner, 1999). There are three channels: sodium (*Na*), potassium (*K*) and leakage channel.

Consider a time-dependent input current $i(t)$ being injected into this circuit. Using Kirchoff's current law one can describe the circuit in Fig. 7.5 with the following equation:

$$i(t) = i_C(t) + \sum_{ch} i_{ch}(t) \quad (7.1)$$

where i_C is the capacitor's charging current and $\sum i_{ch}(t)$ is the sum of the ionic currents. The current through an ideal capacitor C (i_C) is proportional to a change of voltage across the capacitor (v_C):

$$i_C(t) = C \frac{dv_C}{dt} \quad (7.2)$$

Substituting (7.2) into (7.1) and reorganising (7.1) yields:

$$C \frac{dv_C}{dt} = i(t) - \sum_{ch} i_{ch}(t) \quad (7.3)$$

The total of three current components was given by Hodgkin and Huxley as:

$$\sum_{ch} i_{ch}(t) = I_{Na} + I_K + I_L = G_{Na} m^3 h (v_C - V_{Na}) + G_K n^4 (v_C - V_K) + G_L (v_C - V_L) \quad (7.4)$$

where G_{Na} , G_K and $G_L = 1/R$ are each the conductance of the sodium, potassium and leakage channels, respectively, V_{Na} , V_K and V_L are constants called reverse potentials, the variables m and n control the *Na* channel and variable h controls the *K* channel. The m , n and h variables are described by three differential equations:

$$\begin{aligned}
\frac{dm}{dt} &= \alpha_m(v_c)(1-m) - \beta_m(v_c)m \\
\frac{dn}{dt} &= \alpha_n(v_c)(1-n) - \beta_n(v_c)n \\
\frac{dh}{dt} &= \alpha_h(v_c)(1-h) - \beta_h(v_c)h
\end{aligned} \tag{7.5}$$

where α and β are empirical functions of v_c . The conductance, reversal potentials and channel controlled variables are empirical parameters. They have been adjusted to fit the data obtained for the squid axon. Although, (7.4) and (7.5) describe the processes involved in spike generation in the giant axon of squid, the model can be adjusted for modelling the spike generation of the soma of the neuron through the addition of further ion channels (Maass, 1999).

Because of its biological relevance the model is commonly used by neuroscientists. The main disadvantage of this model is its complexity which makes the model computationally difficult and therefore not practical for modelling large networks of neurons. It has been shown that this model can be successfully reduced to a single variable threshold-fire model where a spike is created when the membrane potential crosses a threshold value (Kistler, Gerstner & van Hemmen, 1997).

7.2.3 Spike response model

The spike response model (SRM) is a simplified threshold-based model of a biological neuron that is based on the fact that all spikes are very similar in shape, amplitude and duration. Therefore, it is more logical to use the time of a spike concurrency rather than the spike shapes to convey information. This model does not consider the neuron's dynamics at the ion level like in the Hodgkin and Huxley model; instead it uses the membrane potential PSP of a neuron to describe the neuron's dynamics. In this model, a neuron is approximated as a threshold element by two response kernels, one describing the neuron's reaction to an incoming pre-synaptic spike ($\varepsilon(t - t_j^f)$) and one describing the neuron's reaction to its own post-synaptic spike ($\eta(t - t_i^f)$):

$$PSP_i = \sum_{t_i} \eta_i(t - t_i^f) + \sum_j \sum_{t_j^f} w_{ji} \varepsilon_{ji}(t - t_j^f) \tag{7.6}$$

where t_i^f and t_j^f are the firing times of the post-synaptic and pre-synaptic neurons, respectively. The membrane potential of the post-synaptic neuron is equal to 0, *i.e.* PSP_i

$= 0$, before any pre-synaptic spikes arrive. A pre-synaptic spike at t_j^f increases or decreases the post-synaptic neuron's PSP_i by $w_{ji} \times \varepsilon(t - t_j^f)$. This amount is dependent on the synaptic strength of the connection between the pre-synaptic and post-synaptic neurons, represented by w_{ji} . The neuron fires when its PSP_i cross some constant threshold value. The $\eta(t - t_i^f)$ kernel models the neuron's post-synaptic spike and its activity during the refractory period (Fig. 7.4). The typical shapes of the ε kernel are given in Fig. 7.3. The main characteristic of this model is that PSP_i depends on the time since the last output spike as well as on the times of the incoming spikes, *i.e.* t_i^f and t_j^f . In the next subsection, we introduce a special case and the best known realization of spike response models (Gerstner & Kistler, 2002), namely the leaky integrate-and-fire model.

7.2.4 Leaky integrate-and-fire model

The leaky integrate-and-fire model (LIF) is a very popular model of the soma of a neuron. This model is easy to explain through the use of principles from the field of electronics. The soma is modelled by an RC circuit (Fig. 7.6.). The synapses are modelled by a low-pass filter that receives an incoming pre-synaptic spike and transforms it to a current that charges RC . The comparator on the output of the RC circuit is used to model the spike creation. When the voltage across C crosses some threshold value (PSP_θ) the circuit generates a spike.

To satisfy Kirchhoff's current law, $i(t)$ must be equal to sum of all currents through C and R :

$$i(t) = i_R(t) + i_C(t) \quad (7.7)$$

Substituting $i_R(t) = v(t)/R$ and $i_C(t) = C dv/dt$ in (7.7),

$$i(t) = \frac{PSP(t)}{R} + C \frac{dPSP}{dt} \quad (7.8)$$

We arrange (7.8) and introduce the membrane decay time constant of the neuron $\tau = R \times C$.

$$\tau \frac{dPSP}{dt} = -PSP(t) + R i(t) \quad (7.9)$$

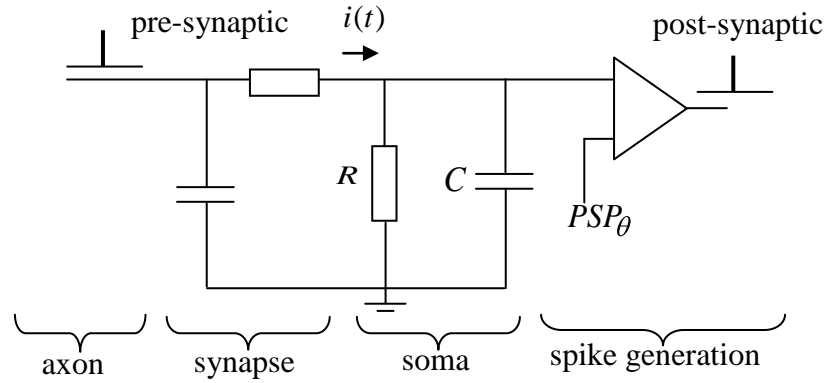


Fig. 7.6. The leaky integrate-and-fire model of a neuron. The resistor R in parallel with a capacitor C models the soma of the neuron. The current $i(t)$ charges C . When the voltage across C exceeds PSP_{θ} the neuron produces a post-synaptic spike.

When PSP exceeds PSP_{θ} the neuron produces a post-synaptic spike. After the spike the PSP is reset to its resting value (PSP_{rest}) and the integration process starts again for every incoming spike. In the absence of pre-synaptic spikes the neuron's PSP gradually decreases (ruled by τ) and after some time reaches PSP_{rest} . The leakage can be removed by setting τ to 0. This transforms a leaky integrate neuron to an integrate-and-fire neuron (IF), a neuron that does not exhibit the gradual decrease of PSP in the absence of incoming spikes.

One of the simplest implementations of IF neurons can be found in (Allen, Abdel-Aty-Zohdy & Ewing, 2004) where the membrane potential is modelled as a counter which is incremented each time a pre-synaptic spikes arrives. When the counter reaches a threshold a postsynaptic spike is generated and the counter is reset to zero. In this model the membrane potential is not continuous, but rather it exhibits steps of constant amplitudes. In the next subsection we introduce the Thorpe's model, where the membrane potential is also increased in steps, but where the steps are not constant and instead depend on the relative order of the firing times of the pre-synaptic neurons.

7.2.5 Thorpe's model

Thorpe's model is a simple version of an integrate-and-fire neuron without leakage in which the membrane potential of a post-synaptic neuron i at time t depends on the firing order of all its pre-synaptic neurons j (Delorme, Perrinet & Thorpe, 2001):

$$PSP_i(t) = w_{ji} \sum_j \text{mod}^{order_j} \quad (7.10)$$

where $\text{mod} \in [0,1]$ is the modulation factor, order_j is the firing order of a pre-synaptic neuron j , $\text{order}_j \in [0, m-1]$ where m is the number of pre-synaptic neurons connected to neuron i , w_{ji} is the synaptic efficacy (weight) of the synapse connecting neuron i and neuron j . The synaptic modification is proportional to the firing order of the spikes received:

$$\Delta w_{ji} = \text{mod}^{\text{order}_j} \quad (7.11)$$

with the same convention as in (7.10). Connections to superior order pre-synaptic neurons are given a higher weight. Higher weights results in stronger connections.

When a spike is received via one of the neuron's synapse, the neuron's potential PSP_i builds up. When the PSP_i reaches a threshold PSP_{θ_i} , neuron i fires a spike. After the spike is fired the PSP_i is set to 0:

$$PSP_i = \begin{cases} PSP_i + P_{ji}, & PSP_i < PSP_{\theta_i} \quad (\text{spike received}) \\ 0, & PSP_i = PSP_{\theta_i} \quad (\text{spike emitted}) \end{cases} \quad (7.12)$$

This model has been proven to be an efficient way of modelling the visual system (Thorpe, Delorme & Van Rullen, 2001) and it has been used to create an audio model (Wysoski, Benushova & Kasabov, 2007b). The suitability of the Thorpe's model as a tool for modelling taste recognition is discussed in Chapter 8.

7.2.6 Izhikevich's model

Integrate-and-fire models are computationally efficient but not very biologically accurate. Izhikevich's model combines the biological plausibility of the Hodgkin-Huxley model with the computation efficacy of the integrate-and-fire models. It is a simple model of cortical neurons which uses two differential equations with four dimensionless parameters a , b , c and d (Izhikevich, 2003):

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I \quad (7.13)$$

$$\frac{du}{dt} = a(bv - u) \quad (7.14)$$

The auxiliary after-spike resetting is given by:

$$IF v \geq 30mV THEN \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \quad (7.15)$$

where v and u are dimensionless variables representing the membrane potential and membrane recovery variables that each account for the activation K^+ and the inactivation Na^+ of ion currents, respectively. Synaptic currents or injected DC-currents are represented by I . The parameter a describes the time scale of u , while b describes the sensitivity of u to v . The parameters c and d describe the after-spike reset value of v caused by fast high-threshold K^+ and slow high-threshold Na^+ and K^+ conductance. The threshold potential PSP_θ is dependent on the history of PSP and it is in the range of $[-55mV, -40mV]$. Despite its simplicity this model has been able to describe different types of neocortical neurons dynamics (regular, fast and low-threshold spiking, intrinsically bursting, chattering, *etc.*) (Izhikevich, 2003).

7.3 Information encoding

Spiking neurons communicate using spikes, so to pass information to a SNN the information must be appropriately encoded. This typically requires the encoding of some analogue values to the temporal domain, *e.g.* a value from a sensor is encoded as a spike pattern. W. Gerstner (1999) divides the encoding techniques used into two basic categories, rate coding and pulse coding.

Rate codes calculate averages, such as:

1. Firing rate averages over time;
2. Average responses of one neuron when stimulated repeatedly with the same input many times;
3. Average activity over a population of neurons.

Gerstner argues that a very popular pulse code based on the time-to-first-spike is consistent with rate coding, *i.e.* if the mean firing rate is high, then the first spike is expected to occur earlier.

The time-to-first-spike coding scheme is visualised in Fig. 7.7 where each neuron spikes only once during some small given period of time (Δt). Given three mean firing rates f_{r1} , f_{r2} and f_{r3} , where $f_{r1} < f_{r3} < f_{r2}$, the time-to-first-spike coding scheme may result in the

spike-firing times set out in Fig. 7.7, where the neuron associated with the highest average rate (f_{r2}) spikes first.

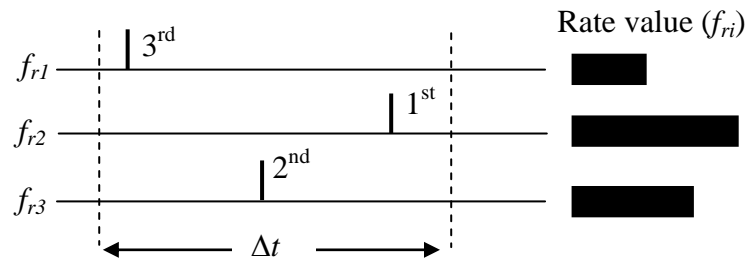


Fig. 7.7. The three average firing rates f_{r1} , f_{r2} and f_{r3} , where $f_{r1} < f_{r3} < f_{r2}$, have been converted to the temporal domain using the time-to-first-spike encoding scheme. The neuron associated with the highest average rate spikes first.

Alternatively, one could use frequency encoding based on probability (Fig. 7.8). Higher values would result in more spikes being produced in some small period of time Δt (Fig. 7.9). A survey of the temporal codes suitable for representation of sensory information can be found in (Cariani, 2004).

```

for all samples in the data set
  for each variable i
    for all time steps
      calculate probability p of sample i
      generate a random number n
      if n is smaller than p
        create a spike at time t
    end
  end
end

```

Fig. 7.8. Pseudo code for frequency encoding based on the probability in which the higher variable values result in more spikes being produced in the same period of time.

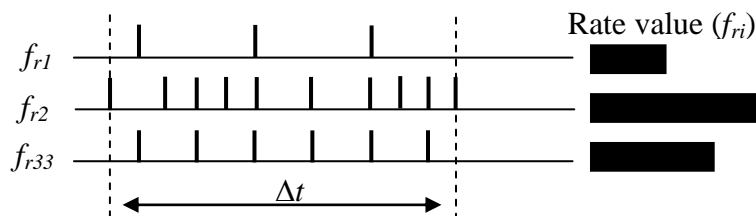


Fig. 7.9. Three average firing rates f_{r1} , f_{r2} and f_{r3} , where $f_{r1} < f_{r3} < f_{r2}$, have been converted to the temporal domain using a frequency encoding scheme. The neuron associated with the highest average rate produces the highest number of spikes.

7.3.1 Rank order coding

Rank order coding (ROC) is based on empirical evidence which proves that the reactions of the visual system to stimuli are very quick. Therefore the encoding in the visual system must be based on the relative timing of spikes rather than on the precise timing of spikes or on average rate codes (Thorpe, Delorme & Van Rullen, 2001). The model has been studied extensively by Thorpe and his team at CERCO, Toulouse (Van Rullen, Guyonneau & Thorpe, 2005; Perrinet *et al.*, 2001; Delorme, Perrinet & Thorpe, 2001; Thorpe, Delorme & Van Rullen, 2001; Van Rullen & Thorpe, 2001; Van Rullen *et al.*, 1998; Gautrais & Thorpe, 1998) and it has proved appropriate for modelling the human visual system. This code is robust to noise and can be used to rapidly transmit a large amount of information (Gautrais & Thorpe, 1998). Their studies also suggested that this coding could be appropriate for encoding information for other sensory modalities. Consequently, this coding scheme has been employed for image processing tasks (Wysoski, Benuskova & Kasabov, 2006; Thorpe, Delorme & Van Rullen, 2001), sound localization (Amin & Fujii, 2004) and speech recognition (Loiselle *et al.*, 2005). It was also used in a preliminary audio model (Wysoski, Benuskova & Kasabov, 2007a).

In ROC a neuron can spike at most once and the earlier spikes are given the highest importance. Consider a vector $v = \{p_1, p_2, p_3\}$ where $p_1 < p_3 < p_2$. Using this encoding scheme p_1 is given rank 2, p_2 is given rank 0 and p_3 is given rank 1 (Fig. 7.10). This results in a new representation of the vector v as $v_{encoded} = \{2, 0, 1\}$.

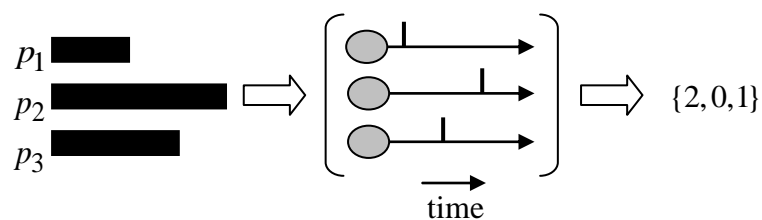


Fig. 7.10. Rank Order Coding. The highest value (p_2) is given the highest rank (0) and the lowest value (p_1) is given the lowest rank (2).

7.3.2 Population coding

Population coding was proposed by Bothe, La Poutré and Kok for encoding continuous input variables in spike-times with a population of neurons (Bothe, 2003; Bothe, La Poutré & Kok, 2002; Bothe, Kok & La Poutré, 2002). Their work extends the network from Natschläger and Ruf (1998) who in turn extended the work carried by Hopfield

where he used time-delay networks to model the features of the olfactory systems (Hopfield, 1995). This model consists of simple integrate-and-fire neurons where each connection between two neurons i and j consists of multiple connections with different delays (Fig. 7.11).

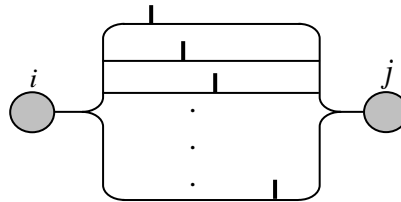


Fig. 7.11. This figure shows how two neurons are connected in the model proposed in (Natschläger & Ruf, 1998). Each connection consists of a set of connections associated with their own delays.

To achieve multiple paths with different delays, each continuous input variable is encoded with a population of m neurons containing overlapping Gaussian receptive fields (Fig. 7.12). Each input variable is encoded into a m -dimensional vector of spike times. Given $m = 6$, six Gaussian values representing the excitations of six new neurons are defined by the intersection points of the vertical line representing a variable v with the six Gaussian curves. Those values are translated into firing times (t^f) where the most stimulated neuron fires first ($t^f = 0$) and the least stimulated neuron fires some time later ($t^f = t_{max}$). Therefore, the range $[v_{min}, v_{max}]$ is translated to firing times in the range $[0, t_{max}]$. Clearly, different sets of receptive fields result in different sets of firing times.

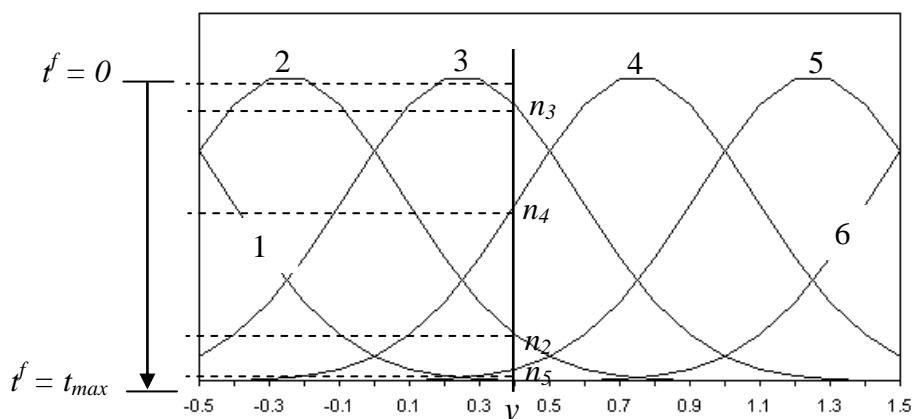


Fig. 7.12. An input variable v is encoded with six Gaussian receptive fields. The highest excited neuron (n_3) is given the shortest firing time, the second most excited neurons (n_4) is given some later firing time and so on. The lowest excited neurons (n_1 and n_6) are given increasing later firing times (t_{max}).

The centres (c_i) and widths (σ) of the receptive fields can be calculated using the following two equations (Bothe, Kok & La Poutré, 2002):

$$c_i = v_{\min} + \frac{(2i-3)}{(m-2)(v_{\max} - v_{\min})} \quad (7.16)$$

$$\sigma = \frac{1}{\beta(m-2)(v_{\max} - v_{\min})} \quad (7.17)$$

where $[v_{\min}, v_{\max}]$ is the range of a variable v , m is the number of Gaussian receptive fields where $m > 2$, and β is a constant optimised for the data set at hand.

In Fig. 7.12 all Gaussian curves have the same width. However, a mixture of broadly and sharply tuned Gaussian receptive fields can be used to increase coding accuracy (Zhang & Sejnowski, 1999). Furthermore, different widths can be used to encode different variables. An optimal combination of broadly and sharply tuned Gaussian receptive fields could be found if one variable is encoded accurately, with narrow Gaussians for this variable, and wider Gaussians for the other variables (Eurich & Wilke, 2000; Zhang & Sejnowski, 1999). Fig. 7.13 shows one possible scenario in which data samples from three classes labelled as Class 1, Class 2 and Class 3 are plotted in a 2D space where p_2 values are plotted against p_1 values. It is apparent that Class 2 and Class 3 samples are very similar, and that they are both quite different to the Class 1 samples. If population coding using receptive fields with constant widths is not sufficient, one possible development is to have a combination of broadly and sharply tuned receptive fields, where the sharply tuned receptive fields are used to increase the sparseness of the Class 2 and Class 3 data samples.

Population encoding increases the dimensionality of the input data. If a data sample is a n dimensional vector and m receptive fields are used, then the encoding scheme results in $n \times m$ input neurons (Fig. 7.14). While this may cause the curse of dimensionality problem in traditional neural networks, this increase of neurons is desirable in the spiking neural networks. Precision can be improved by increasing the number of neurons, and given enough neurons any precision can be achieved (Thorpe, Delorme & Van Rullen, 2001). We explore this concept in Chapter 8.

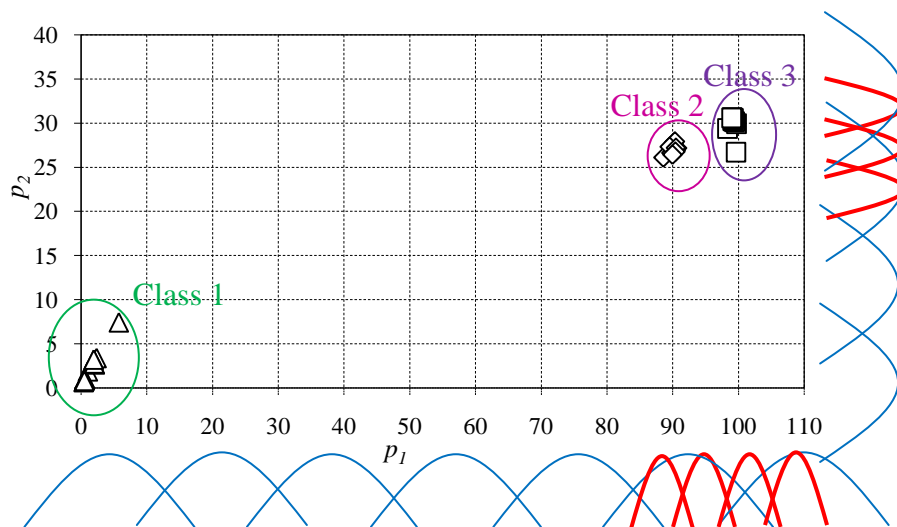


Fig. 7.13. Clusters of samples from three different classes in 2D (p_2 vs. p_1). Broadly and sharply tuned receptive fields might be required to further differentiate the data samples (Soltic, Wysocki & Kasabov, 2008).

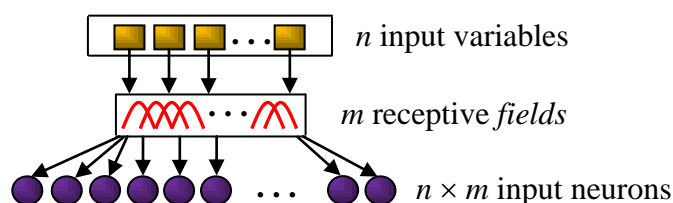


Fig. 7.14. Population encoding increases the dimensionality of input data from n to $n \times m$. This might cause a problem in traditional neural networks but it is desirable when computing with spiking neural networks. Increased dimensionality allows more information to be conveyed and leads to higher system accuracy (Soltic, Wysocki & Kasabov, 2008).

Population encoding increases the sparseness in the original data. A smaller number of input neurons than before the data is encoded are activated. In other words, only a subset of connections conveys spikes. While sparseness is desirable when modelling the visual processing of natural images (Baddeley, 1996; Olshausen & Field, 1996) and in odour representation (Martinez & Hugues, 2004), the main motivation for the use of population coding in Bothe's work was to increase clustering accuracy and the capacity for information transfer (Bothe, Kok & La Poutre, 2002).

7.4 Evolving spiking neural network architecture and training

7.4.1 Evolving spiking neural network architecture

In this thesis the neural networks capable of taste recognition are built using an SNN evolving learning algorithm similar to one proposed by Wysoski *et al.* (Wysoski, Benuskova, and Kasabov, 2006). The networks are hierarchically organized into two layers (L1 and L2) of IF neurons (Fig. 7.15). The latency of the neurons' firing is decided by the order of the incoming spikes. The spikes are propagated from the input layer (L1 neurons) to the output layer (L2 neurons) in a feed-forward manner. All connections between the neurons are excitatory. Each L1 neuron has a single one-to-one connection ($w_{in} = 1$) to its input and one weighted synaptic connection to each of the output neurons via a weight w_{ji} . The w_{ji} values are found during the training of the network by using (7.11). Note that to keep the figure readable the weights are not shown in Fig. 7.15.

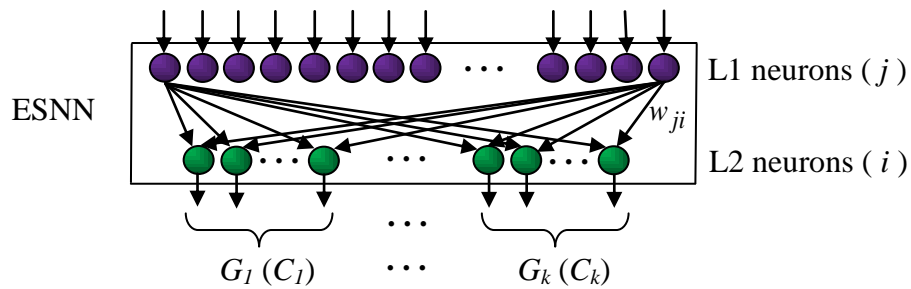


Fig. 7.15. The structure of the evolving spiking neural network (ESNN) comprises L1 and L2 neurons. The L2 neurons are created during the learning stage. Each class C_i is represented with an ensemble of neurons (G_i), and each ensemble is trained to represent one class. For the sake of figure clarity, the weights between the input and the L1 neurons, which are equal to 1, as well as the weights w_{ji} between the L1 and the L2 neurons are not shown. The w_{ji} weights are found during the training of the network.

The L1 neurons represent receptors (sensors) whose values are encoded using the coding scheme proposed in Chapter 8, Section 8.3. The L2 neurons are sensitive to the order of the incoming spikes and can generate at most one spike, each when a pattern is presented to the network. They model the signal recognition process by learning to extract information about the input patterns. L2 neurons are created during the training phase. The L2 layer is initially empty and all L2 neurons are created during the evolving process. An ensemble of L2 neurons (G_i) learns to recognise samples from a class C_i .

The number of L2 neurons differs from one ensemble to another, *i.e.* different classes are learnt by a different number of L2 neurons. The number of L2 neurons created for a class depends on the variation of samples taken for that class. If the samples exhibit large variations, more L2 are created in the G_i . The number of L2 neurons in the ensembles is controlled by a network parameter S_θ . It may seem like the network simply grows to provide a unique path to an output for each input, but that is not the case. One L2 neuron learns a number of input patterns and is able to classify correctly a previously unseen input pattern.

Once the network is trained, it can be used to classify samples. During the classification only the first spikes from the L2 neurons are considered. Given that sample p_{ki} , representing some class (*e.g.* tastant tas_k), is presented to a trained network. This sample p_{ki} has not been used for training of the network and is yet to be classified. Let the neuron $L2_m$, $L2_m \in G_k$, spikes before than any other $L2_i$ neuron from the other ensembles, *i.e.* $L2_i \notin G_k$. The sample p_{ki} is classified as a C_k sample. If two or more neurons from different ensembles spike at the same time the network is unable to classify the sample. However, the classification can take place if multiple neurons from the same ensemble spike at the same time. In the other words, if a number of L2 neurons $\{L2_1, \dots, L2_n\}$ where $\{L2_1, \dots, L2_n\} \subset G_k$ spike at the same time then p_{ki} is classified as C_k (*e.g.* tastant tas_k).

For a given input pattern, the membrane potential PSP_i of an output $L2_i$ is the sum of the postsynaptic potentials PSP_{ji} . The membrane potential PSP_{ji} is caused by a single weighted spike received from $L1_j$ that is connected to $L2_i$:

$$PSP_i = \sum_{j=1}^m PSP_{ji} \quad (7.18)$$

where m is the number of L1 neurons connected to $L2_i$, and PSP_{ji} is given by (7.10). The spikes from the L1 neurons serve as inputs to the L2 neuron. Thus, each spike received via one of $L2_i$'s synapses, builds up the $L2_i$'s membrane potential PSP_i . When PSP_i reaches some critical value PSP_θ , $L2_i$ fires a spike. After $L2_i$ generates a spike its PSP_i is reset to 0 as noted in (7.12).

L2 neurons have dynamic thresholds and their selectiveness can be controlled by adjusting their PSP_θ . The threshold values PSP_θ are found during the training of the

network. The threshold value PSP_{θ_i} of $L2_i$ is set to a proportion of its maximum membrane potential value:

$$PSP_{\theta_i} = c PSP_{i \max} \quad (7.19)$$

where $c \in [0, 1]$ is the constant shared by all L2 neurons.

7.4.2 Evolving spiking neural network training

As stated earlier, the L2 neurons are trained to respond selectively to the presence of a given sample by an incremental learning algorithm proposed in (Wysoski, Benuskova, and Kasabov, 2006). During training, a new neuron, $L2_n$, is created for a training pattern p_i . The newly created neuron $L2_n$ is trained using (7.11) and (7.19). Let the current pattern p_i belong to a class C_c . The Euclidean distances between $L2_n$ to all pre-existing $L2_o \in \{L2/C = C_c\}$ are calculated and used as a similarity measure. If the similarity between $L2_n$ and $L2_o$ is below some threshold value S_θ ($S_\theta > 0$) then $L2_n$ is aggregated with $L2_o$. During aggregation the weights and threshold of $L2_n$ are averaged into the values of $L2_o$, and $L2_n$ is discarded. The network's structure continuously evolves through the creation and merging of neurons based on incoming data.

All training patterns are presented to the network only once, making this a fast algorithm. When some new patterns become available, the network learns them without the need to be trained on the old samples. This type of learning is much faster than the learning used to train MLP networks where the training data must be repeatedly presented to the network during the training. Furthermore, when new data samples are obtained the MLP network must be trained again on both the new and the old samples. The incremental learning employed here allows the network to learn new classes of input data (signals) as they become available.

The learning used to create and train the L2 neurons is a type of local learning. As the neural network is created (and trained) the ensembles (clusters) of L2 neurons are produced. The L2 neurons from an ensemble G_i are trained on the characteristics of only one class, *i.e.* only on the samples p_j belonging to one class C_i (positive examples). The L2 neurons, $L2 \in G_i(C_i)$, learn to respond with earlier spikes when presented with data samples from C_i . Each L2 in G_i could have been updated upon the presentation of more than one sample p_j , but all these samples came from the same class of samples C_i .

7.5 Spiking neural networks for Ecological Decision Support: an unexplored potential

The biggest strength of spiking neural networks is their ability to discriminate between events in time. They are therefore suitable for modelling systems that require an analysis of temporal data. Many applications in ecological modelling require data mining in the temporal domain, but as we discussed in the earlier chapters the modelling of these processes is frequently done using methods that cannot explore temporal patterns.

Lately, the application of arrays of sensors teamed with a pattern recognition system for ecological modelling has been explored. These systems have been used for air quality monitoring (Zampolli *et al.*, 2004; Andò *et al.*, 1998), for detection of volatile gasses in the air (Zanchettin & Ludermir, 2004; Korenman & Kalach, 2003), and for monitoring the quality of potable water (Gardner, *et al.*, 2000). In the past, the analysis of the sensory data was done using a traditional neural network or PCA analysis; both techniques are unable to process temporal signals. However, recently biologically plausible olfactory models based on spiking neurons have been proposed. The examples of this type of modelling can be found in the work done by Raman *et al.* (2006), Martinelli, Amico and Di Natale (2006), Allen, Abdel-Aty-Zohdy and Edwing (2005), Brody and Hopfield (2003), Allen *et al.* (2002) and Rochel *et al.* (2002). Electronic tongues have been also proposed for environmental monitoring. For example, an electronic tongue was used for monitoring the contamination of natural (Gutiérrez *et al.*, 2008) and drinking water (Krantz-Rülcker *et al.*, 2001), where the sensory data was analysed with a traditional neural network and a PCA analysis. While much effort has been put into designing taste sensors with better discrimination characteristics (Gutés *et al.*, 2007), there are no taste recognition models based on SNN.

If SNN be considered are to be considered for environmental modelling researchers need to see that there are new advantages that these networks can offer. One appealing advantage of SNN is that the utilization of spiking neurons allows for a more biologically realistic model. If we can create more biologically realistic models of the sensory systems scientists will be able to build intelligent autonomous robots for tasks that are impractical and harmful for humans to perform. Another advantage of SNN that should be considered is that spiking neurons can process temporal information. The reasons for using SNN are further explored in Chapter 8 where a taste recognition model based on spiking neurons is proposed and evaluated. However, the

meteorological datasets used in Chapters 5 and 6 contain static spatial data and applying SNN to these datasets were not appropriate.

7.6 Conclusion

In this chapter the operation of spiking neurons, the third generation of artificial neurons, has been presented. Even though spiking neurons offer a more realistic description of biological neuron behaviour than their predecessors, they are still oversimplified approximations of biological neurons. Over the past decade a number of spiking neuron models have been proposed. The more detailed and accurate models are usually the ones that are harder to implement. Often a model is chosen due to its simplicity, rather than its biological plausibility. As a result, the simple integrate-and-fire models and their leaky versions have been very popular.

One of the main characteristics of spiking neurons is that they communicate using spike trains. Their dynamics are characterised by the membrane potential that changes under the influence of incoming (pre-synaptic) spikes. The main advantage of spiking neurons is their ability to process temporal data. However, this becomes a burden when designing a system based on spiking neurons for some real-world modelling problems. The real-world data is often obtained through various sensors that measure steady-state values. That data must be transformed to a form that the spiking neurons can handle. There are many coding schemes used for this purpose, some of which are based on the measured responses of the human brain. It is still unclear how the exactly human brain works and therefore it is hard to choose one coding scheme over another.

This chapter gives the background information to support the main contributions presented in Chapter 8 and Chapter 9, where a novel spiking neural network is introduced and employed for taste recognition. The design of the taste recognition model was complicated by the fact that even though the human gustatory system is very important to our wellbeing it is a poorly explored compared to other sensory systems such as the visual and olfactory systems. Also, there are no current spiking neural network models for taste recognition that this work could take inspiration from. The work presented in the following chapters is the first of its kind and the results aim to contribute to the promotion of SNN as a new tool for building taste recognition systems. A detailed explanation of the new model follows in Chapter 8.

Chapter 8

An evolving spiking neural network model for taste recognition (ESNN-PC-TR)

As mentioned before, taste recognition systems can be used for environmental monitoring. The human brain has an amazing ability to recognise hundreds of thousands of different tastes. We ask, “Can we build artificial systems that can achieve that?”. This chapter explores how spiking neurons could be employed for building more biologically plausible and efficient taste recognition systems. It presents a new approach for taste recognition in a simple artificial gustatory model. To start with, the current knowledge on taste perception is reviewed. Several recent discoveries that have contributed to a better understanding of taste perception are then discussed. Furthermore, the benchmark works from this field are surveyed, concentrating on the taste recognition models based on neural networks.

The survey of the benchmark works is followed by a description of a novel spiking neural network model, the evolving spiking neural network with population coding ESNN-PC. The ESNN-PC is based on the ROC integrate-and-fire type of neurons. We apply ESNN-PC for taste recognition and call this model ESNN-PC-TR. To simulate the influence of taste receptor cells on the activity of adjacent cells the population encoding of sensory data is used. As proven in this work, using population encoding makes the proposed model of the gustatory system more biologically plausible and more accurate. We propose and then evaluate a novel knowledge discovery technique for extracting accumulated knowledge from a trained ESNN-PC-TR.

The new modelling approach was implemented in software and hardware. In this chapter the model and its implementation was tested on two real-world taste datasets where the effectiveness of the information encoding and the network’s adaptive properties were explored. The results of this analysis are presented and discussed. The proposed approach is compared with an approach based on MLP that was previously applied to the same modelling problem. The hardware implementation of the model proposed here is discussed in Chapter 9.

8.1 Taste perception

8.1.1 Biological models of taste-coding

It is now widely accepted that there are five different basic tastes, *i.e.* the bitter taste, salt taste, sour taste, sweet taste and umami or savoury taste (Chandrashekar *et al.*, 2006; Margolskee, 1993). In addition, the taste for fats has been suggested as a possible sixth basic taste (Mattes, 2005; Gilbertson, 1998). Taste perception starts in the mouth where chemicals in foods excite taste receptors or taste receptor cells (TRC) that are organised in taste buds which are positioned throughout the mouth cavity. To some degree all tastes can be sensed through parts of the tongue and mouth cavity (Chandrashekar *et al.*, 2006). The taste receptors are connected to the gustatory nucleus of the brain via three types of sensory fibres (Dulac, 2000). Exactly how the taste signals are coded and transmitted to the brain is still arguable. For years, two biological models of taste-coding have been favoured, namely the ‘labeled-line’ model and the ‘cross-fiber’ model (Chandrashekar *et al.*, 2006; Reed, Tanaka & McDaniel, 2006, Bradbury, 2004; Dulac, 2000; Margolskee, 1993). The labeled-line model assumes the existence of dedicated TRC to a taste. In other words, this model states that TRC are specialized sensory cells for detection of only one taste. Each tastant, a chemical that stimulates the sensory cells in a taste bud, stimulates specialized TRC which in turn convey their signals via a dedicated fibre to the brain. In the cross-fiber model, TRC are broadly tuned to more than one taste and a pattern of activity across a number of TRC characterizes the taste sensation.

For years the cross-fibre model has been preferred to the labeled-line model. Broadly tuned TRC have been simulated using an array of non-selective sensors with partially overlapping selectivity where the sensors’ electrical characteristics, for example capacitance, change under the influence of the chemical composition of the tastants (Fig. 8.1). The sensors as a group have different responses to different tastants and the measured responses form the taste signatures of the measured tastants. These arrays of sensors simulating broadly tuned TRC are called electronic tongues (Cortina *et al.*, 2006; Mikhaleva, and Kulapina, 2005; Gallardo *et al.*, 2003; Riul Jr. *et al.*, 2002; de Sousa *et al.*, 2002; Ivarsson *et al.*, 2001; Toko, 2000; Hauptmann *et al.*, 2000). An electronic tongue provides multidimensional data vectors that can be presented for analysis to a pattern recognition system. The electronic tongue and the pattern recognition system represent an artificial gustatory system (Fig. 8.2). Various artificial

models based on measuring the responses of an array of non-selective taste sensors in the presence of various tastants have been proposed. The survey of benchmark works in this field is provided in Section 8.2.1 Table 8.1.

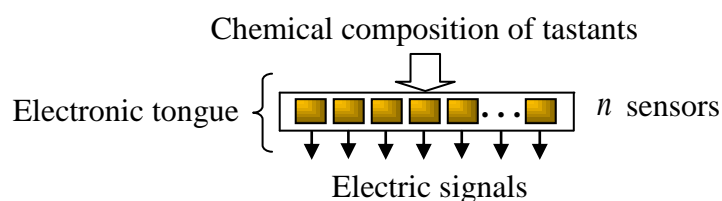


Fig. 8.1. An array of n non-selective taste sensors (represented by ■) converts the chemical composition of tastants into the electrical domain. The sensors are referred to as an electronic tongue (Soltic, Wysocki & Kasabov, 2008).

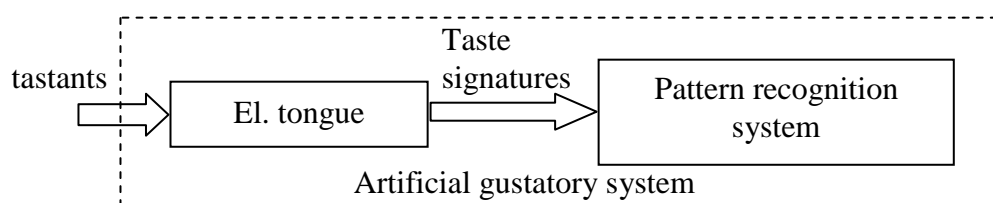


Fig. 8.2. Block diagram of an artificial gustatory system. The artificial gustatory system comprises a sensor array modelling the tongue, *i.e.* the electronic tongue, and a pattern recognition system modelling the taste recognition.

Even though the cross-fibre model has historically been favoured, results from recent studies support the labeled-line model of taste perception for at least the bitter, sour, sweet and umami tastes. Three families of TRC have been identified, each responsive to a different taste modality (Chandrashekar *et al.*, 2006):

- The T1R family (three genes T1R1, T1R2 and T1R3) play an important role in the perception of sweet taste, and a combination of T1R1 and T1R3 forms a broadly tuned umami receptor.
- The T2R family of around 30 TRC are responsive to bitter taste.
- The PKD2L1 TRC are responsive to sour tastes.

Despite the intrinsic differences between the labeled-line and cross-fiber models these models share two common features; they are both unimodal and static (Katz, Nicolelis & Simon, 2002). In keeping with the static nature of the models, Varkevisser *et al.* (2001) measured dynamic taste responses and then rate coded these responses prior to

analysing them by a second generation neural network. Both models are unimodal because they ignore the influences of other sensory modalities on the perception of taste. They assume that taste perception only depends on the taste information processed by the gustatory neurons and is not influenced by any other non-gustatory system, *i.e.* the auditory, olfactory, visual, somatosensory systems.

8.1.2 Dynamics of taste-coding

The suitability of the above mentioned models of taste perception is disputed by empirical evidence that gustatory neural system responses are dynamic. Patricia Di Lorenzo and her team conducted experiments where they observed significant contribution of the spike timing to the discrimination of taste stimuli (Di Lorenzo and Victor, 2003). They performed repetitive measurements of the activities in the nucleus of the solitary tract of the rat while the rats were influenced by four taste stimuli. The stimuli were delivered using a custom-built system allowing the authors to accurately measure action potentials over multiple repetitions of the same experiment, thus overcoming the difficulty of keeping a stimulus flow constant over multiple trials (Katz, 2003). They concluded that the timing of spikes contains information about taste stimuli indicating that temporal coding of taste information must not be ignored. However, Di Lorenzo and Victor found that temporal coding played a larger role in the response transient (initial 2 of 10 seconds of response) than in the remainder of the taste response. Furthermore, some cells showed more temporal rich responses than others, suggesting that taste coding is indeed very complex. Glendinning, Davis and Rai (2006) are also in favour of temporal coding. They measured how the taste system of an insect responds to three bitter tastants and concluded that temporal information is indeed essential for the discrimination of bitter stimuli.

The unimodal nature of the labelled-line and cross-fiber models is also contradicted. Humans are intrinsically aware that their perception of the taste of food and beverages is governed by more than just taste information. Evidence shows that the smell-taste interaction is particularly significant to the perception of taste (Stein and Stoodley, 2006). A smell-taste fusion algorithm has been proposed in (Wide *et al.*, 1998) to provide an overall taste opinion similar to that of human beings. In this smell-taste fusion algorithm, data from electrical smell and tongue sensors was first classified using two ANN, each dedicated to the classification of either smell or taste, and then the results from those networks were fused using the fusion algorithm. This approach

improved the overall discrimination of three types of juice, *i.e.* apple, orange and pineapple. In recent years, a lot of attention has been given to the multimodal property of taste perceptions (Verhagen and Engelen, 2006; Yoshimura *et al.*, 2004; Dulac, 2000; Toko, 2000), and if this continues it will not be long till more integrated systems are proposed.

In summary, the coding of the gustatory system is quite complex. Clearly, the current biological and computational models of taste-coding are not sufficient to describe complicated taste processing. The evidence on the temporal dynamics of taste coding is too compelling to be ignored. However, the knowledge of how exactly the taste information is conveyed from the taste receptors to the brain is rather limited and breakthrough findings on taste coding are still to come. In the next section, a survey and selection of references to artificial taste recognition models are provided. The survey shows that modelling taste recognition is an emerging topic.

8.2 Artificial taste recognition models

8.2.1 Benchmark works

The static and unimodal cross-fiber model of taste perception has been used as the backbone to many artificial taste recognition systems. As shown earlier in Fig. 8.2, models of the artificial gustatory systems consist of an array of taste sensors mimicking the taste receptor cells and a pattern recognition system modelling the neural activities in the brain. The number and type of utilized sensors in the sensor array differ from model to model. In some cases features are extracted from the captured responses in order to lower the number of inputs into the pattern recognition system. It could be said that the most popular sensing techniques are based on (Table 8.1):

- Potentiometry, where the taste is measured as the potential difference between working and reference electrodes,
- Voltammetry, where the current measurements taken at different voltammetric waveform pulses are the measure of taste,
- Conductometry, where the changes in the conductivity of sensors characterise the tastant.

TABLE 8.1

The results of our survey of taste recognition systems, with references.

Tastants	Electronic tongue	Method for analysis	References
Hydrocarbons	frequency	PCA	Hauptmann <i>et al.</i> , 2000
Various beverages	potentiometry	PCA	Toko, 2000
teas, detergents	voltammetry, potentiometry	PCA	Ivarsson <i>et al.</i> , 2001
wines, mineral waters	conductometry	PCA	Riul Jr. <i>et al.</i> , 2002
wines, mineral waters	conductometry	ANN	Riul Jr. <i>et al.</i> , 2004; de
juice	voltammetry	PCA	Sousa <i>et al.</i> , 2002;
ammonium, potassium	potentiometry	ANN	Wide <i>et al.</i> , 1998
phenolic compaunds	voltammetry	ANN	Gallardo <i>et al.</i> , 2003
Nanylphenol homologues	potentiometry	ANN	Gutés <i>et al.</i> , 2005
chlorine, nitrate, bicarbonate	potentiometry	ANN	Mikhaleva & Kulapina, 2005
			Cortina <i>et al.</i> , 2006

The pattern recognition system is usually based on an ANN, although PCA is also popular. It is interesting to observe that ANN have proven highly accurate even in cases where PCA could not provide good discrimination among samples (Riul Jr. *et al.*, 2004). Several examples of using multilayer feed-forward neural networks (MLP) for taste recognition also exist in the literature. As stated earlier, these networks have a fixed structure that is typically optimized by trial and error. Moreover, they learn offline which makes them less flexible and means they have a limited ability to learn new tastes as they become available. The traditional ANN and not the SNN have been used for modelling taste recognition. Even when the dynamic responses were measured, they were rate coded and fed to a traditional neural network. For example, traditional artificial neural networks were used for sorting neural spikes contained in mixed spike trains recorded from the taste organs of an insect (Stitt *et al.*, 1997). Also, in one other experiment, the traditional neural networks were used to analyse the patterns of action potential in a hamster's single taste bud (Varkevisser *et al.*, 2001). The responses were

recorded and rate coded by several features. These features were then presented to a feed forward neural network for pair wise comparison of the tastants.

8.2.2 Taste recognition models based on traditional neural networks and their shortcomings

The traditional artificial neural networks have been used as pattern recognition tools in modelling taste perception. In these systems the inputs into the network are either values obtained from an array of taste sensors or rate coded action potentials recorded from the taste buds of a species. Although, traditional neural networks have been found to be accurate, a number of difficulties associated with such networks could be avoided by using either a different type of traditional neuron model, by employing a different learning algorithm or by employing spiking neurons. For instance, the black-box syndrome found in the traditional neural network models and encountered in (Varkevisser *et al.*, 2001) can be avoided by using a fuzzy neural network and the hand-tuning of the network parameters in (Stitt *et al.*, 1997) could be automated by using genetic algorithms. Furthermore, the problem of finding the appropriate size for the hidden layers (Cortina *et al.*, 2006; Gallardo *et al.*, 2003; de Sousa *et al.*, 2002) can be avoided by using an evolving training algorithm. If modelling needs to be more biologically plausible, then an SNN can be used. Moreover, traditional feed forward networks of artificial neurons are slow to learn. Training these networks consists of repetitively presenting the network with large training data sets. The number of repetitions (epochs) is usually found by trial-and-error. Also, these networks must be retrained on old and new data every time a new data sample is obtained. Both these disadvantages can be avoided by employing an evolving, one-pass learning algorithm, as demonstrated in the SNN model described in the next section.

8.3 ESNN model for taste recognition

The ESNN principles introduced in Chapter 7 Section 7.4 are used here to build a taste recognition model that circumvents most of the issues described in the previous section. This novel taste recognition model is trained using an evolving algorithm similar to the one explained in Chapter 7 Section 7.4.2 where all L2 neurons are created and trained during one-pass training. Furthermore, the training patterns can be presented to the network only once. The trained network is also able to learn new tastants as they become available. Because the model is based on simple integrate-and-fire neurons

which are explained in Chapter 7 Section 7.2.5, it is more biologically realistic than the approaches that use rate-coded neural networks. The details of the new model are described below (Fig. 8.2).

The latency of the L2 neurons' firing is decided by the order of the incoming spikes. To recall, as described in Chapter 7 Section 7.3.1, ROC is a simple and easy to implement technique that encodes information with a set of neurons where each neuron can spike at most once. The spikes are ranked over a set of neuronal units. Earlier spikes are given a higher importance. In other words, earlier spikes are given a lower rank order (0, 1, ...) and later spikes are given a higher rank order (... , $n-1$, n). Here, the higher the taste sensor value, the lower the rank, and the sooner the L1 that is connected to the sensor is fired. This type of coding has been chosen because of the following reasons:

1. ROC is capable of modelling spiking neural dynamics. As explained in Section 8.1.2, the dynamics of the gustatory system are important for taste coding and recognition.
2. In ROC the exact timing of the spikes is not explicitly present in the model, only the order is important. This makes ROC appropriate for our data which lacks a temporal component. The data and its features are detailed later in Section 8.5.
3. ROC has been proven as an efficient and effective way of modelling the visual system (Thorpe, Delorme, and Van Rullen, 2001) and it has been used in a preliminary audio model (Wysoski, Benuskova, and Kasabov, 2007b). A simple way of integrating visual and auditory modalities using fast neurons with ROC has also been suggested (Wysoski, Benuskova, and Kasabov, 2007a). Although the reaction times of the taste recognition system are slower than those of the visual system (Bonnet *et al.*, 1999), ROC is still preferred over other encoding scheme. As discussed earlier in Section 8.1.2, the perception of food is multimodal, *i.e.* influenced by the visual, auditory, somatosensory and olfactory systems. Therefore, using encoding that can model the signals from a variety of sensory systems can help create a more accurate artificial model of the human multimodal perception of food and beverages.
4. ROC is a simple and easy to implement. At the moment, this is not a very important feature because the networks built in this thesis comprise a small number of neurons and work in an off-line mode. But the simplicity of this

algorithm might be beneficial in the future when more tastant data becomes available.

The above approach is extended by introducing population coding of individual features with multiple Gaussian receptive fields. In Chapter 7 Section 7.3.2, we reviewed the work by Bothe, Kok and La Poutré where multiple Gaussian receptive fields were used to encode each feature with a population of m neurons. They translated each feature into an m -dimensional vector of spike times. In their work, each feature value is encoded with m real values in the range $[0, 1]$ where each value represents the firing time of the neurons encoding the respective variable. Here, each input pattern p is encoded into $s \times m$ neurons and rank order coded, where s is the number of taste sensors, and m is the number of equally spaced Gaussian receptive fields. The Gaussian fields' centre values c_i were determined by:

$$c_i = g_{min} + \frac{(2i-3)(g_{max}-g_{min})}{2(m-2)} \quad (8.1)$$

$$g_{min} = d_{min} - d_{av} \quad (8.2)$$

$$g_{max} = d_{max} - d_{av} \quad (8.3)$$

$$d_{av} = \frac{d_{max}-d_{min}}{2} \quad (8.4)$$

where d_{max} and d_{min} are the maximum and minimum values of the input data and $i \in (1, m)$. In this design, the input data was normalised (*i.e.* $d_{max} = 1$ and $d_{min} = 0$). Therefore, the proposed coding method is a two-pass process; first, each input pattern is encoded into $s \times m$ neurons, second, each of the $s \times m$ neurons is assigned a rank.

Fig. 8.3 shows how one sensor value v is encoded with $m = 6$. The dashed lines show the range of data ($[0, 1]$). The input value v is translated into an order of firing. Six Gaussian values (L1 neurons) are obtained from six points where the vertical line representing v crosses the six Gaussian curves. The values below some positive constant β (in this work $\beta = 0.01$) were set to 0 and they do not contribute to the post-synaptic potentials of the L2 neurons. All newly created L1 neurons are rank coded. The highest value L1 neuron ($L1_4$) is given the order $O = 0$, the second highest L1 neuron ($L1_3$) is given order $O = 1$, $L1_5$ is given order $O = 2$, and so on. Thus, the input value v is translated into an m -dimensional order vector $O = \{-, 3, 1, 0, 2, -\}$, where '-' represents the values that do not contribute to the membrane potential of the L2 neurons, *i.e.* the

values below β . A number of experiments observing how m affects the classification accuracy of the trained network, and consequently the taste recognition system, were conducted. The results of these experiments and the observations regarding m and the classification accuracy are reported later in Section 8.6.

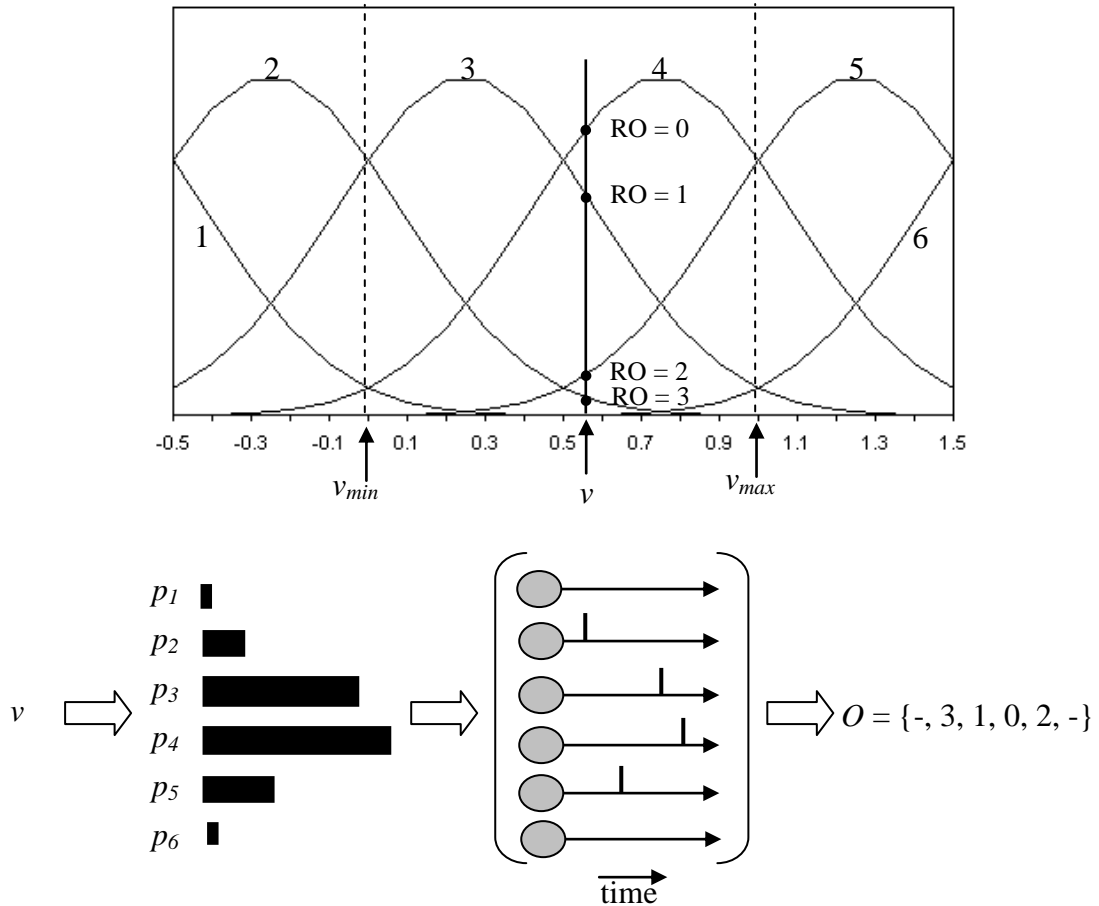


Fig. 8.3. Coding of an input sensor value v , and its corresponding orders O defined by the intersections of the vertical line representing v with all six Gaussian curves ($m = 6$). The value v is translated into orders $O = \{-, 3, 1, 0, 2, -\}$, where ‘-’ represents $p_i < \beta$.

There are several reasons why population coding should be used for encoding of taste data:

1. As presented in (Bothe, La Poutre and Kok, 2002), population coding increases the capacity of information transition. The number of different patterns that can be transmitted in ROC is $n!$ (n factorial) where n is the number of pre-synaptic neurons (Gautrais and Thorpe, 1998). Therefore, increasing the number of L1 neurons from n to $m \times n$ increases the number of patterns that can be transmitted from $n!$ to $(m \times n)!$.

2. Population coding has the potential to increase the accuracy of a network of spiking neurons (Bothe, Kok and La Poutré, 2002).
3. Using population coding for taste recognition tasks is supported by several recent findings that taste receptor cells may modify the activity of adjacent cells (Huang *et al.*, 2005; Zhao *et al.*, 2005; Katz, Nicolelis & Simon, 2002). In our model, this influence is simulated by encoding each input variable with Gaussian receptive fields. The reaction of the L1 neurons to the signals coming from the taste sensors decrease according to the Gaussian formula described by 8.1. – 8.4. Namely, one cell (v) influences m L1 neurons ($p_1 - p_m$) where each L1 is given a different excitation (Fig. 8.3). Using population coding makes our simulation of the gustatory system more biologically reasonable.
4. Population coding increases the temporal distance between the taste samples. This is discussed in Section 8.5 where the properties of the data employed in this work are introduced.

The schematic representation of the proposed taste recognition system is given in Fig. 8.4. The system consists of an electronic tongue, a Gaussian receptive field (GRF) layer and an evolving spiking neural network (ESNN). The model is named ESNN-PC-TR. In the illustration in Fig. 8.4, the electronic tongue contains only seven sensors ($s = 7$) for sampling liquids. Each sensor value is coded using m equally spaced receptive fields. At the moment all values are encoded using the same number of receptive fields, although the number of receptive fields (m) differs from experiment to experiment. The spiking neural network comprises two layers, L1 and L2, similar to the network explained in Chapter 7 Section 7.4.1.

8.4 Software simulation of the ESNN-PC

Spiking neurons and spiking neural networks have been implemented in hardware (Murray, 1999) and simulated in software on a general-purpose computer. To date, the majority of hardware implementations of artificial neural networks are implementations of MLP. Recently the use of FPGA as the hardware platform suitable for the implementation of SNN as well as other artificial neural networks has been explored (Al-Kazzaz & Khalil, 2008; Chalhoub, Muller & Auguin, 2006; Brunelli *et al.*, 2005). FPGA are high density digital devices that can operate in a highly parallel manner. This makes them a logical choice for building parallel processing systems such as artificial

neural networks (Reaz *et al.*, 2002). We propose an FPGA implementation of the ESNN-PC in Chapter 9.

SNN are most often simulated using custom-build software or off-the-shelf simulating environments that allow for detailed biophysical simulations (Delorme *et al.*, 1999) such as NEURON and GENESIS. Brette *et al.* (2007) divide algorithms for the simulation of spiking neural networks into:

- Synchronous (clock-driven), in which all the neurons' membrane potentials are calculated and threshold conditions are checked simultaneously at the same time;
- Asynchronous (event-driven), in which the neurons' membrane potentials are calculated only for the activated neurons when they receive pre-synaptic spikes or when they produce post-synaptic spikes;
- Hybrid, a combination of the above.

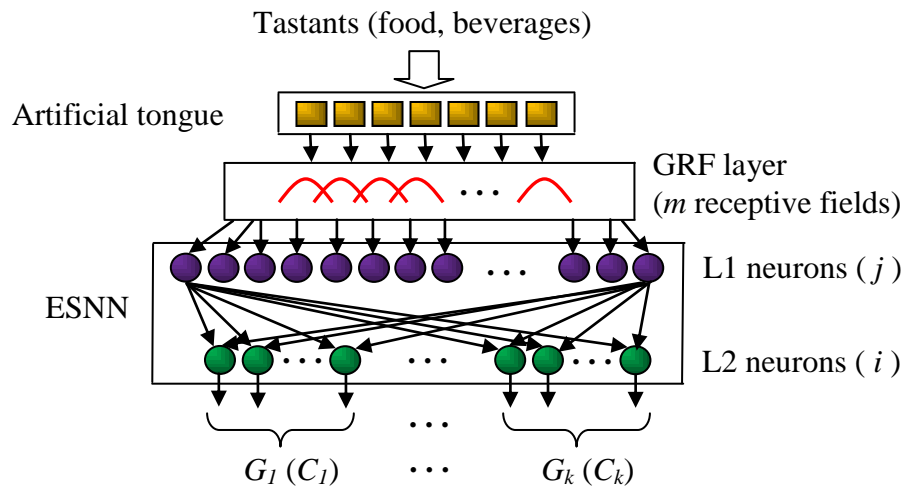


Fig. 8.4. The proposed taste recognition system comprises three layers: an electronic tongue that samples tastants, a GRF layer with m equally spaced Gaussian receptive fields which perform the population coding of sensor values, and an ESNN in which L1 represents the taste receptors whose values are encoded using ROC and L2 are the simple integrate-and-fire units sensitive to the order of the incoming spikes (Soltic, Wysocki & Kasabov, 2008).

Pseudo-code for the basic clock-driven and event-driven algorithms are given in Fig. 8.5. Clock-driven algorithms are simpler to implement, mainly due to the fact that post-synaptic spikes are determined at the same time as the post-synaptic potential is calculated. The determination of the post-synaptic spikes in the event-driven simulators

is more complex because the post-synaptic spikes do not need to occur at the times of the incoming spikes. For the sake of simplicity only the pseudo-code for a simple event-driven model is given in Fig. 8.5, where updating the post-synaptic potential of the neurons and checking the threshold conditions is done at the same time. Analysis of the clock-driven and event-driven algorithms is outside the context of this thesis however an in-depth analysis can be found in (Marian, 2002).

<pre> // clock-driven t = 0 while t < T for every neuron calculate PSP end for every neuron if PSP > PSP_θ emit spike PSP = 0 end end end t = t + dt end </pre>	<pre> // event-driven t = 0 while there are spikes and t < T get earliest spike calculate PSP of targetNeuron if PSP > PSP_θ for all downstream connections insert spike in the queue end PSP = 0 end end end </pre>
---	--

Fig. 8.5. Pseudo-code for the clock-driven and event-driven algorithms. Note that the post-synaptic spikes in the event-driven algorithm occur at the arrival time of the pre-synaptic spikes and updating the post-synaptic potential of the neurons and checking the threshold conditions is done at the same time. If this was not the case, the algorithm would be more complex.

A simulation tool specially designed for this investigation has been implemented in Java and the model was also hosted in FPGA (see Chapter 9). The simulations were run on a general-purpose computer (Intel Core 2, 2 GHz, 1 GB RAM, Windows XP). The neurons were updated and their spiking checked at the times of the incoming spikes (a version of the event-driven approach). The simulator allows the entry of the model parameters and for browsing the memory for the input data. It provides information about the status of the simulation and the final classification results (Fig. 8.6). The simulator is available on www.manukau.ac.nz/departments/e_e/staff/soltic.asp.

A leave-one-out cross-validation version of the ESNN-PC-TR simulator to test the performance of the ESNN based taste recognition system was also developed.

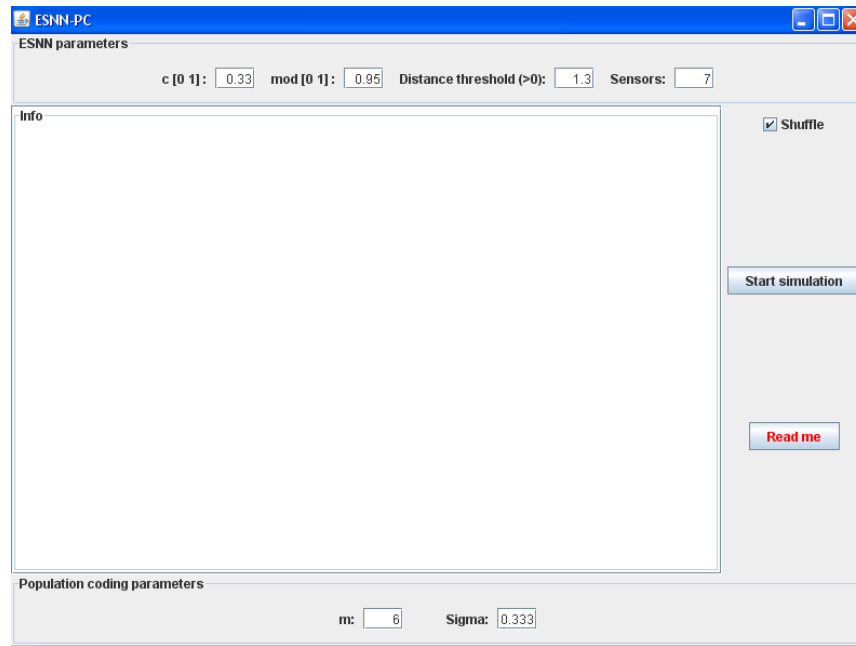


Fig. 8.6. A snapshot of the simulator’s GUI. The interface allows the ESNN-PC parameters to be customised.

8.5 Data characteristics

In order to investigate the suitability of the ESSN-PC-TR model for taste recognition two subsets of datasets that have been previously successfully used in the experiments reported by Humberto de Sousa (de Sousa *et al.*, 2002) and Antonio Riul Jr. (Riul Jr. *et al.*, 2004) and their teams were employed. In their work, the pattern recognition systems were based on MLP networks, but they suggested that the data is also appropriate for use in other techniques. The samples were generated by an array of seven non-selective sensor units (taste sensors) based on conducting polymers that are able to discriminate the basic tastants (Riul Jr. *et al.*, 2002). The sensor units are made of 5-layer Langmuir-Blodgett ultra thin (to the order of nm) films made up of four different types of material; polypyrrole (PPy), 16-mer polyaniline (16-mer), stearic acid (SA), a natural polymer (chitosan), and a layer-by-layer mixed film of PPy and SA, 16-mer and SA, and PPy and 16-mer. The sensors convert the chemical composition of a tastant into the electrical domain as illustrated earlier in Fig. 8.1. Each tastant excites each sensor to a lower or a greater degree by changing the sensor’s capacitance (the cross-fiber model of taste-coding). Collectively the sensors acquire enough information for the discrimination of the liquids. Measured signals have no time-dependent structure (Riul Jr. *et al.*, 2002).

Two sets of data have been used. Set 1 (small and unbalanced), the water set, contains 49 measured patterns, 10 patterns of two different mineral water brands (Water Class1 and Water Class2), 16 patterns of Milli-Q water (Water Class3) and 13 patterns of one other water (Water Class4). Set 2 (large and balanced), the wine set, contains 750 measured patterns, 150 patterns of each of the five wines; same brand but different vintages (Wine Class1 and Wine Class2), same brand but different grapes (Wine Class 3 and Wine Class5) and one other wine (Wine Class4).

Fig. 8.7 shows the average responses of the 7-sensor array to different water types. Three important features can be observed in this figure: (i) there is a similarity between the Class1 and Class2 averages and between the Class3 and Class4 averages; (ii) the difference in magnitude between Class3 and Class4 averages and the Class1 and Class2 averages; (iii) the small standard deviations which are shown as the error bars, especially for the Class2 samples.

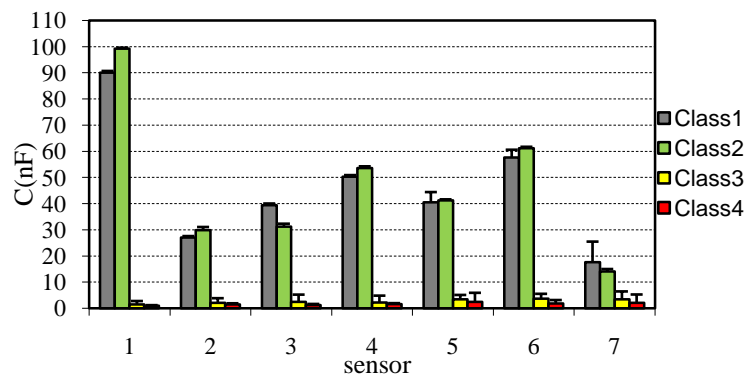


Fig. 8.7. Averages and standard deviations of the water samples. Note: the Class1 and Class2 averages are very similar, as are the Class3 and Class4 averages; the Class3 and Class4 averages are obviously of lesser magnitude than the Class1 and Class2 averages. The standard deviations in this data set are small.

Fig. 8.8 shows the result of encoding the water Class1 and Class3 patterns with rank order coding (ROC, on the left) and rank order with population coding using six receptive fields (RO-POP C, $m = 6$) on the right. The encoded dataset contains 10 Class1 patterns and 16 Class3 patterns. The greyscale levels represent the level of excitation of each L1 neuron. Darker levels are associated with more excited neurons and the white levels represent the inactive neurons, *i.e.* those neurons that do not have an effect on the postsynaptic potential of the connected L2 neurons. While the ROC plot is rather puzzling, there is a visible temporal deference between the excitations of the L1 neurons ‘belonging’ to different classes in the RO-POP C plot. There is clearly a

pattern when population coding was used. Because all values are coded using the same set of Gaussian receptive fields, the small sample differences result in L1 neurons with the same or very similar rank orders.

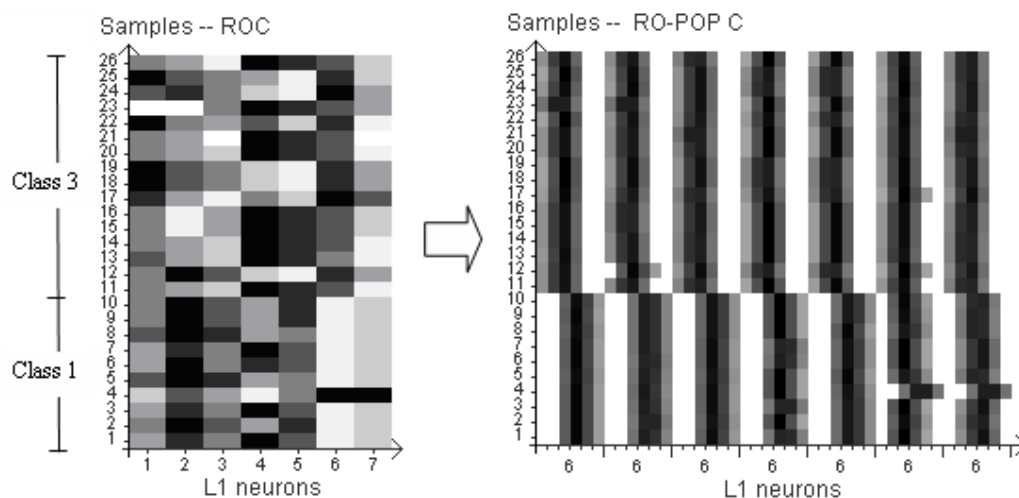


Fig. 8.8. Water Class1 and Class3 data encoded using rank order coding (ROC) and rank order population coding (RO-POP C) with six receptive fields ($m = 6$). See Chapter 7 Section 7.4.1 for an explanation of the L1 neurons. The greyscale levels represent the levels of excitation for each of the L1 neurons. Darker levels are associated with more excited L1 neurons (Soltic, Wysocki & Kasabov, 2008).

The sensors' average responses (μ) to Class1 and Class3 samples and the classes' standard deviations (σ) are detailed in Table 8.2. All values are expressed in nF and rounded to 2 significant figures. The sparseness of the data has been increased, resulting in more accurate classification results which are reported later on in Section 8.6.

TABLE 8.2

Details of Class1 and Class3 samples. Averages (μ) and standard deviations (σ) of Class1 and Class3 samples are expressed in nF and rounded to 2 s.f. (Soltic, Wysocki & Kasabov, 2008)

		S1	S2	S3	S4	S5	S6	S7
C_1	μ	90	27	39	50	41	58	18
	σ	0.63	0.57	0.69	0.58	3.9	2.9	7.8
C_3	μ	1.4	2.1	2.5	2.2	3.4	3.7	3.4
	σ	1.4	1.8	2.8	2.7	1.7	1.8	3.1

As introduced earlier, the wine dataset includes 750 measured patterns of five different wine brands/vintages (Class1, Class2, Class3, Class4 and Class5). Each wine is represented by 150 samples with average and standard deviation values shown in Fig. 8.9. It is interesting to observe that this dataset has bigger inner-class and inter-class variations than the water dataset. The inner-class variations can make the signatures of two samples from different classes very similar or identical, affecting the training of the ESNN-PC-TR and the overall system accuracy.

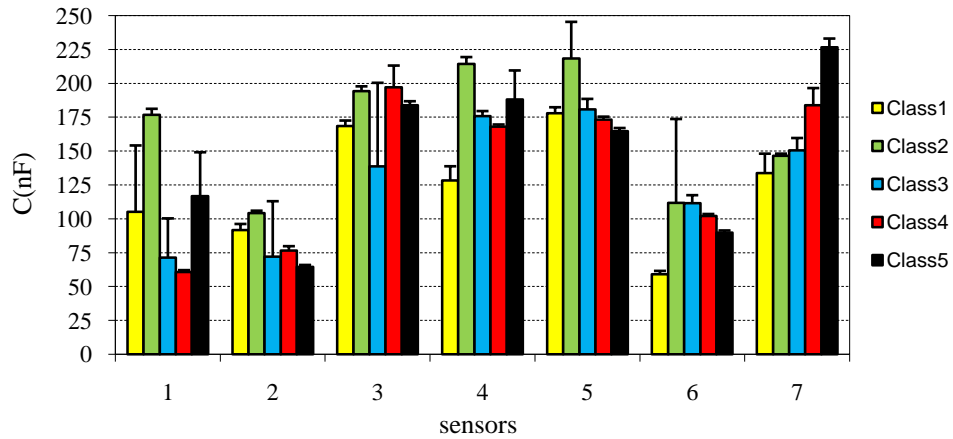


Fig. 8.9. Wine dataset - average values and standard deviations of the 150 samples in each of the five different wine classes.

8.6 Applications of the ESNN-PC-TR model for water and wine recognition

The ESNN-PC-TR was evaluated on six different scenarios of taste recognition as summarized in Table 8.3. The training datasets (D_{tr}) were obtained by taking random subsets, typically about 50%, of the original data D . The remaining samples made up the testing sets (D_t) used to validate the trained network. Because the order of training data can influence the creation of the network and affect the network's classification accuracy, each scenario was repeated 10 times; each time the data was randomly split, the network was created and evaluated and the average accuracy was calculated. The network parameters mod , c and σ were kept constant in all experiments, *i.e.*, $mod = 0.95$, $c = 0.33$, and $\sigma = 0.333$. These parameters were hand tuned to achieve the maximum classification accuracy. Changing one of these values can result in poorer accuracies. Other network parameters, m and S_θ , were hand tuned to achieve the highest classification accuracy for the particular data set. The data was normalized to $[0, 1]$ before being encoded.

TABLE 8.3

Details of the six scenarios designed to test the ESNN-PC-TR taste recognition model
(Soltic, Wysoski & Kasabov, 2008)

Scenario	Class	Samples	Classes	Balanced
1	Water 1&3	12	2	No
2	Water 1&2	10	2	Yes
3	Water 1-4	40	4	Yes
4	Wine 2&3	300	2	Yes
5	Wine 1-4	600	4	Yes
6	Wine 1-5	750	5	Yes

8.6.1 Water recognition

The aim of these simulations was to discriminate between the samples of different types of water. Scenario 1 and 2 were designed to evaluate the ROC and RO-POP C coding schemes. In these scenarios the model was tested only on 2-class problems, *i.e.* binary classification. Particularly, networks were built and trained to distinguish between Class1 and Class3 samples, and Class1 and Class2 samples. Note that both subsets had a small number of samples and that in the first scenario using Class1 and Class3 there were 60% more Class3 samples than Class1 samples.

As stated earlier, the distance threshold (S_θ) influences the number of L2 neurons (the neurons in the output layer) representing different classes, *i.e.*, smaller values for S_θ result in more L2 neurons and vice versa. It is possible to build a network with one dedicated L2 neuron for each training sample. Although, in this case the accuracy of the model would be high for the samples the network is trained on, its generalization capability might be affected. Furthermore, it would be impractical and unnecessary to do this for a large number of training samples. For the sake of comparison, S_θ was set to allow the creation of only two L2 neurons, one per each class.

In both scenarios, the sparseness of data introduced by population coding increased the average accuracy. The accuracy of classifying the Class1 and Class3 samples increased from 50% to 100% ($m = 6$) and the accuracy of classifying the Class1 and Class2 samples increased from 66% with ROC encoding to 100% ($m = 8$) with RO-POP C. With RO-POP C coding two L2 neurons each representing one class, were sufficient to achieve this excellent accuracy. Note that, more receptive fields were needed to classify

the Class1 and Class2 samples than to classify the Class1 and Class3 samples. The higher number of receptive fields needed to classify Class1 and Class2 samples can be attributed to the fact that the Class1 and Class2 samples exhibit a higher similarity than the Class1 and Class3 samples (please revisit Fig. 8.7). This indicates that more receptive fields might be required to discriminate between classes with low inter-class distances. Note that in this evaluation equally spaced receptive fields with the same width σ were used. Changing σ or placing the receptive fields in a different place can affect the performance.

Since population coding improved the accuracy of classifying the water samples, the RO-POP C coding of sensory data was further evaluated in a more complex scenario. Scenario 3 investigated the ability of the network to discriminate between more than two classes. Ten networks ($m = 6, S_\theta = 0.5$) were trained to classify four water types with 5 training samples per class. The network parameters m and S_θ were again kept constant in all ten runs. On average 4.7 L2 neurons were created and an accuracy of $91\% \pm 3.94\%$ was achieved. This scenario was explored in more detail using leave-one-out cross-validation. The experiments were repeated ten times, keeping the network parameters constant ($m = 6, S_\theta = 0.75$) in all runs. In all 10 simulations five L2 neurons were created and trained and 37 samples out of 40 were classified correctly (92.5%). The same three samples are misclassified in all ten trials. As expected, these three samples were Class3 and Class4 samples.

8.6.2 Wine recognition

The proposed system was successful in discriminating four types of water. Clearly, the ESNN-PC-TR model did not have problems classifying water samples from a small dataset (40 samples). What happens if the model is faced with numerous samples? The scenarios with the wine datasets were designed to address this question. All wines were represented with 150 samples.

Scenario 4 is similar to Scenario 2 (2-class problem) but the number of samples per beverage was increased more than 10 times. Nevertheless, in all ten simulation runs the network ($m = 8$, two L2 neurons) achieved a 100 % success rate. In Scenario 5, the complexity was increased to a 4-class problem. The samples were coded with 16 receptive fields ($m = 16$) and the mean accuracy over ten runs was 99.62% with a standard deviation of 0.82%.

In addition, the simulations were used to observe how the number of L2 neurons affected the overall performance of the network. For this purpose the number of receptive fields was kept constant ($m = 8$) while the threshold S_θ was changed in each of the runs. This resulted in a different number of L2 neurons being created, which consequently affected the accuracy (a) of the model. The results of this experiment, *i.e.* $a = f(S_\theta)$, are plotted in Fig. 8.10. The numbers next to the each accuracy point represent the average number of L2 neurons created over ten simulations, *e.g.*, on average 43 L2 neurons were created when $S_\theta = 0.1$ and 10 when $S_\theta = 1$. For smaller values of S_θ the accuracy was near or equal to 100%, but as S_θ increased, accuracy declined, reached a minimum value (a_{min}) and then increased slightly. There was no significant change in the accuracy with the introduction of the fifth type of wine (Scenario 6) with the accuracy reaching $99.69\% \pm 0.82$ with an average of 25.5 L2 neurons.

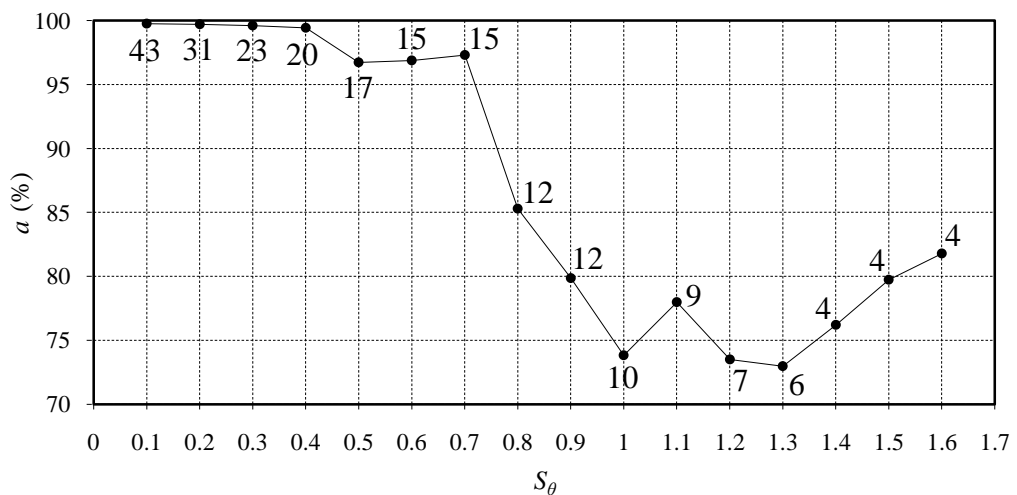


Fig. 8.10. Scenario 5 dataset: accuracy (a) and number of L2 neurons vs. the distance threshold (S_θ), $m = 8$. The numbers next to each accuracy point represent the average number of L2 neurons created in the ten runs (n_{L2av}) (Soltic, Wysoski & Kasabov, 2008).

8.6.3 Comparison with existing work

Here the performance of the ESNN-PC-TR model is compared with the work reported in (de Sousa *et al.*, 2002). While the experiments in the two works being compared are not exactly the same there is enough similarity to discuss the general accuracy levels reached with both approaches. In (de Sousa *et al.*, 2002) conventional neural networks were used for the classification of the water and wine samples. The authors experimented with MLP networks with 5, 10, 15, 20 and 25 hidden neurons trained with

four different algorithms (Standard backpropagation, Backpropagation momentum, Quickprop and RPROP). The correct recognition rate was in the range of $84.71 \pm 23.1\%$ and 100% for the classification of wines and between $65.55 \pm 21.8\%$ and 100% for the classification of water samples. In general, our results from Scenarios 1-6 present the same accuracy levels with the ESSN-PC-TR model more consistently above the 90% mark for both the wine and water datasets.

8.7 Knowledge discovery from ESNN-PC-TR on the case study problems

One of the goals of this thesis is to design new methodologies for CBDS that would allow for knowledge discovery from real-world datasets; therefore it is important to evaluate the knowledge discovery compatibilities of ESNN-PC. For this purpose we examine the knowledge accumulated by the ESNN-PC-TR about the case study problems (Table 8.3). It was proved that ESNN-PC-TR could differentiate between the types of mineral water and also between different brands of wine.

Spiking neural networks receive a lot of research attention. Most of the attention is allocated to designing new spiking neuron models (Izhikevich, 2003; Kunkle & Merrigan, 2002; Kistler, Gerstner & van Hemmen, 2001), new training algorithms (Booij & tat Nguyen, 2005; Bothe, La Pourté & Kok, 2002; Gerstner *et. al*, 1996), design and application of SNN (Belatreche, Maguire & McGinnity, 2007; Brody & Hopfield, 2003; Delorme, Perrinrt & Thorpe, 2001) and encoding of input values (Martinelli, D'Amico & Di Natale, 2006; Loisel *et. al*, 2005; Eurich & Wilke, 2000; Gautrais & Thorpe, 1998; Hopfield 1995). Disproportionally small amount of time is allocated to research centred on the representation of knowledge extracted from data by a SNN in the form of IF-THEN rules. We propose that ESNN-PC can be used for knowledge discovery in applications where information is collected in a brain-like way. We are particularly interested in validity, comprehensibility and usefulness of the extracted knowledge for the user.

8.7.1 Knowledge in ESNN-PC

ESNN-PC is an evolving network that acquires its knowledge from temporal data samples as they become available. We show that this knowledge can be represented in a form of zero-order Takagi-Sugeno fuzzy rules, *e.g.*:

$$\begin{aligned}
& \text{IF } cond_1 \text{ AND } \dots \text{ AND } cond_i \dots \text{ AND } cond_n \text{ THEN } C_j \\
& cond_i = x_i \text{ is } s_k
\end{aligned}
\tag{8.5}$$

where $cond_i$ are conditions of input variables, x_i is an input feature, s_{ki} is a linguistic value represented by its membership function such as *SMALL*, *MEDIUM*, *LARGE* etc. and C_j represents a class label (the output). Conditions, $cond_i$, in ESNN-PC are found through an analysis of the connections between L1 and L2 neurons in a trained network (Fig. 8.4).

In ESNN-PC the input values are encoded using a family of Gaussian receptive fields. A single input value is distributed to multiple L1 neurons through m delayed synaptic connections. There is no delay at the centres of Gaussian receptive fields and this delay increases towards the receptive field edges. Fig. 8.11 shows an example of encoding of two input values v_i and v_j , where $v_i < v_j$ (represented by the two dashed lines in the figure), both from an interval $[0, 1]$ (represented by the two thick green lines in the figure). The values are deliberately chosen so they cause a maximum excitation of two different L1 neurons, *i.e.* two earliest spikes are sent along the LI_{i3} and LI_{j4} terminals, and two very similar spike patterns occur out of sequence by 1 L1 neuron. There is a relationship between which L1 neuron spikes first in a set of L1 neurons and the value encoded into this set of L1 neurons. When smaller values are encoded the first neuron to spike tends towards the lower end of the observed set (LI_{i1} in this example). When higher values are encoded the first neuron to spike tends towards the higher end of the observed set (LI_{i6} in this example).

During training, the change in synaptic weight between L1 and L2 neurons depends on the firing time of the L1 neurons. The synaptic weights associated with the connections which convey earlier spikes increase more than those which convey later spikes. Therefore, based on the weight patterns it is possible to deduce the size of an input value and the contribution of this input value to the modelled output.

Let v_i and v_j belong to classes C_i and C_j , respectively. Assume that two neurons are created and trained, $L2_i$ and $L2_j$. Their theoretical patterns of weights are shown in Fig. 8.12. The $L2_i$ has been trained to recognise the C_i samples and $L2_j$ to recognise the C_j samples. The knowledge about the relationship between v_i and v_j (*e.g.* that $v_i < v_j$) is stored in the synaptic weights of the L2 neurons. However, before this relationship is deduced, one final step is necessary. Consider the network shown in Fig. 8.13. The network comprises two L2 neurons, six L1 neurons and its input is a one-dimensional

vector. At time t a spike is arriving to excite the membrane potentials of the two L2 neurons via the two synaptic terminals (w_{im}, w_{jm}):

$$\begin{aligned} \Delta PSP_i(t) &= w_{im} \bmod^{order_m} \\ \Delta PSP_j(t) &= w_{jm} \bmod^{order_m} \end{aligned} \quad (8.6)$$

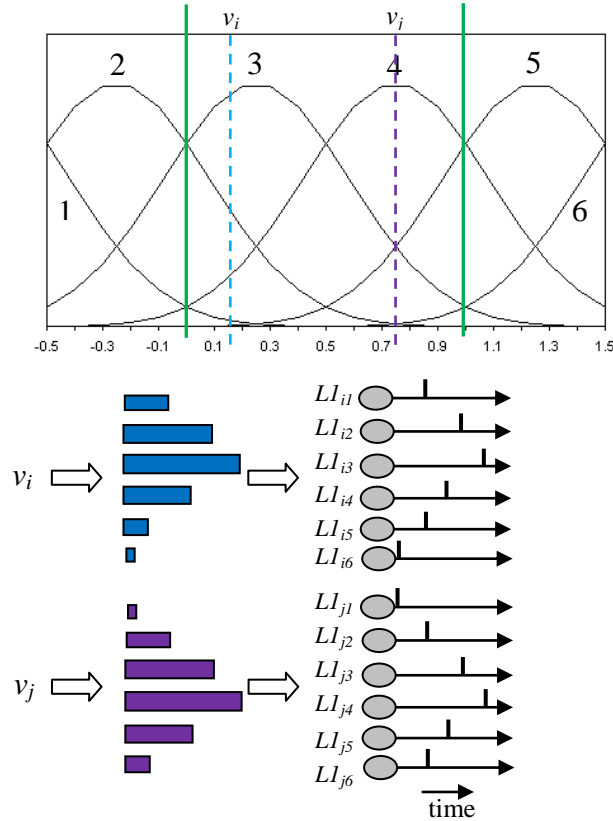


Fig. 8.11. Population encoding of two values v_i and v_j is based on the Gaussian receptive fields. The values are chosen so they cause patterns of spikes, which appear out of sequence by 1 L1 neuron.

The difference between these two excitations is:

$$\Delta PSP_i(t) - \Delta PSP_j(t) = \bmod^{order_m} (w_{im} - w_{jm}) \quad (8.7)$$

Hence the difference in the excitation of the two neurons is a function of difference between their synaptic weights. As the spikes arrive the post-synaptic potentials PSP_i and PSP_j increase by different amount and finally one of the two L2 neurons spikes first. The $|w_{im} - w_{jm}|$ values for the theoretical pattern in Fig. 8.12 are shown in Fig. 8.14. It can be seen that the $L2_i$ neuron ‘favours’ lower input values, *i.e.* a lower input value

will cause a bigger ΔPSP_i . The $L2_j$ neuron does the opposite. Hence the lower values will be classified as C_i and the higher as C_j or:

$$\begin{aligned} \text{IF } v \text{ is SMALL THEN } C_i \\ \text{IF } v \text{ is LARGE THEN } C_j \end{aligned} \tag{8.8}$$

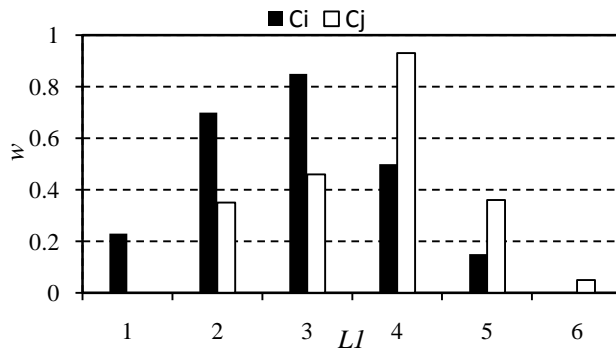


Fig. 8.12. A theoretical pattern of weights of the two L2 neurons connected to the same set of L1 neurons and trained to differentiate between samples belonging to two classes, C_i and C_j .

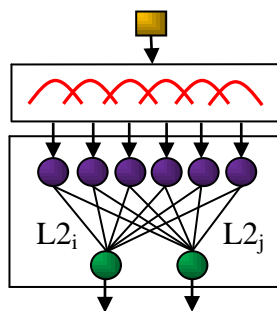


Fig. 8.13. Two L2 neurons receiving spikes from six L1 neurons. As the spikes arrive the post-synaptic potentials of the two neurons, PSP_i and PSP_j , increase by different amounts depending on their synaptic weight values.

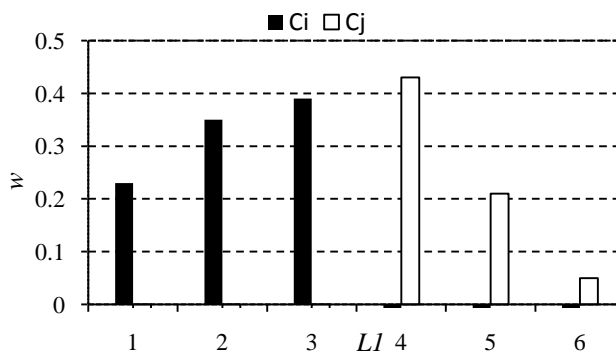


Fig. 8.14. The $|w_{im} - w_j|$ values for the theoretical pattern in Fig. 8.12. As a result it is very likely that smaller (higher) input values are classified C_i (C_j).

8.7.2 Knowledge extracted by ESNN-PC-TR

We start with an explanation of the knowledge discovered in Scenario 2 where two types of water were classified:

Rule 1:

IF x_1 is *SMALL* AND x_2 is *LARGE* AND x_3 is *SMALL* AND x_4 is *LARGE* AND x_5 is *SMALL* AND x_7 is *SMALL* THEN C_0

Rule 2:

IF x_1 is *LARGE* AND x_2 is *SMALL* AND x_3 is *LARGE* AND x_4 is *SMALL* AND x_5 is *LARGE* AND x_7 is *LARGE* THEN C_1

In Fig. 8.15 the values of the synaptic weights in one of the trained ESNN-PC-TR networks built in Scenario 2 are shown. To recall, seven sensors were used to sample four types of water and five brands of wine. Hence there are seven sub-patterns in the weight graphs (the patterns are separated by the red thick vertical lines). In Scenario 2 m was 8. Hence each sub-pattern comprises of 16 bars which are split evenly between two neurons. Note that some w values are equal to zero. It is interesting to observe that in the sixth sensor pattern, x_6 , the neurons' weights are very similar. As a result the contributions of the spikes through these terminals to the post-synaptic potential are very similar. We believe that the values of the sixth sensor did not contribute to the process of distinguishing between the two mineral water samples. This is even more visible in Fig. 8.16 where the corresponding $|w_{1m} - w_{2m}|$ values are shown.

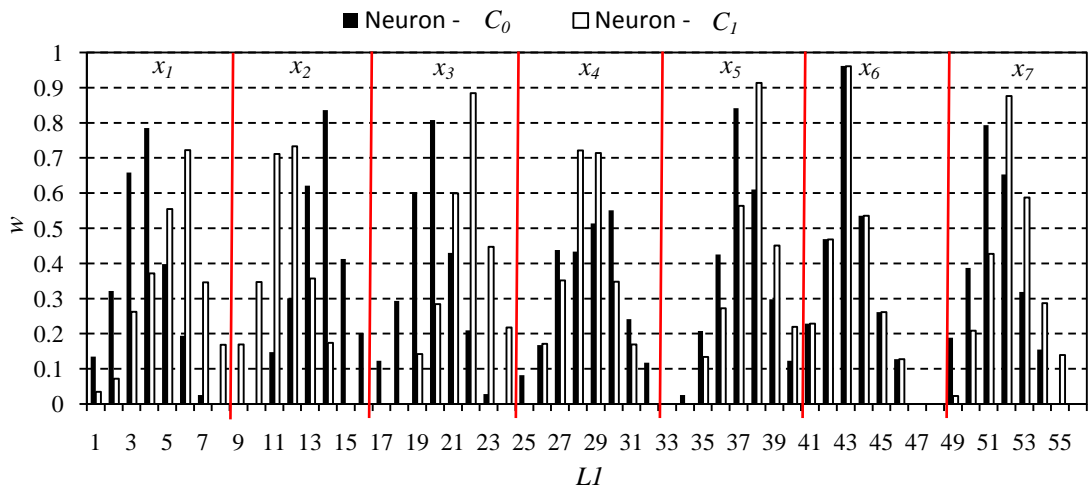


Fig. 8.15. The synaptic weights in one of the trained ESNN-PC-TR networks built in Scenario 2 where the ESNN-PC-TR was trained to distinguish between the two types of mineral water.

Furthermore, it can be seen from Fig. 8.16 that the spikes coming from Sensors 1 – 3 (x_1, x_2, x_3) contribute more to the neurons' *PSP*, than those coming from Sensors 4, 5 and 7 (x_4, x_5, x_7). In order to enable better validation of the rules the data values of all ten samples used to train the ESNN-PC-TR are shown in Fig. 8.17. For example, Rule 1 states that x_3 must be smaller in C_0 than in C_1 . This is justified in Fig. 8.17. As theorised all the x_3 values in C_0 are smaller than the x_3 values in C_1 .

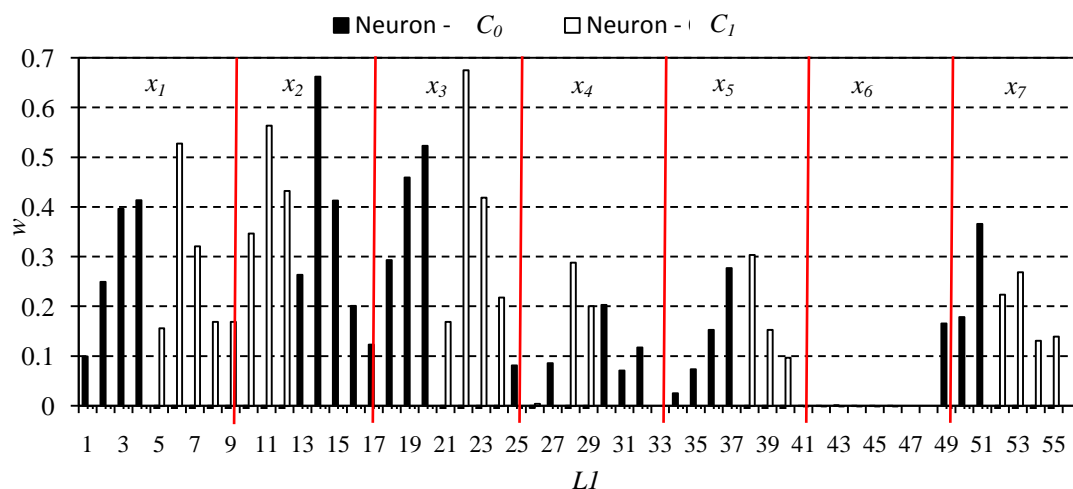


Fig. 8.16. The $|w_{im} - w_{jm}|$ values for the weights in Fig. 8.15.

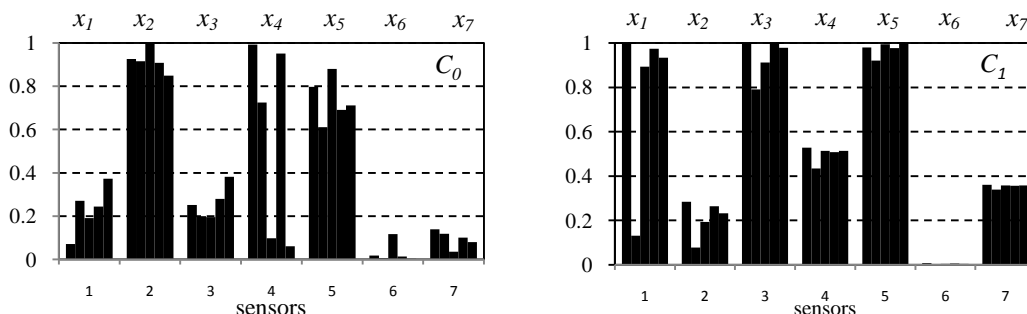


Fig. 8.17. Values of the ten samples (C_0, C_1) used to train the ESNN-PC-TR in Scenario 2.

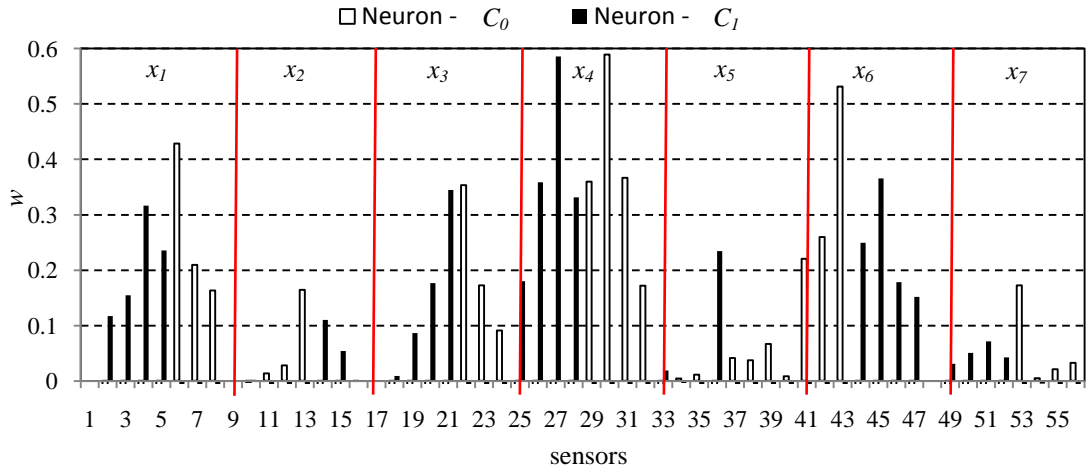
We also illustrate the knowledge discovery capability on the data set from Scenario 4 where two brands of wine were classified. Each brand was represented by 150 samples, randomly shuffled and equally split into training and testing datasets. The $|w_{im} - w_{jm}|$ patterns are shown in Fig. 8.18. It is noteworthy that all the sensor values contribute to the classification of the wines, including Sensor 6 (x_6) which seemed irrelevant in the classification of the water types. Using only two membership functions, *e.g.* *SMALL* and *LARGE*, the following two rules about how two wines can be distinguished between were devised:

Rule 0:

IF x_1 is *SMALL* AND x_2 is *LARGE* AND x_3 is *SMALL* AND x_4 is *SMALL* AND x_5 is *LARGE* AND x_6 is *LARGE* AND x_7 is *SMALL* THEN *Class 0*

Rule 1:

IF x_1 is *LARGE* AND x_2 is *SMALL* AND x_3 is *SMALL* AND x_4 is *LARGE* AND x_5 is *LARGE* AND x_6 is *SMALL* AND x_7 is *LARGE* THEN *Class 1*



8.18. The $|w_{im} - w_{jm}|$ values for the weights in Scenario 4 where the ESNN-PC-TR was used to classify the two brands of wine.

8.7.3 Comment on knowledge extracted by ESNN-PC

Two membership functions are used in ESNN-PC, representing *SMALL* and *LARGE* values respectively. In the two problems explained above these membership functions were manually defined by us, *i.e.* they are obtained from human experts. Our approach can be further developed as a quantitative way of mapping the real membership function from the receptive fields to pre-defined fuzzy membership functions similar to:

$$\begin{aligned}
 & \text{IF } cond_1 \text{ AND } \dots \text{ AND } cond_i \text{ } \dots \text{ AND } cond_n \text{ THEN } C_j \\
 & cond_i = x_i \text{ is } R_j(MD_i)
 \end{aligned} \tag{8.9}$$

where $cond_i$ are the conditions of the input variables, x_i is an input feature, R_j is a fuzzy value represented by its membership function. MD_i represents membership degree to which each x_i belongs to the corresponding receptive field and therefore to the corresponding membership function. C_j represents the output, *i.e.* a class label. Similar as in EFuNN (Kasabov, 2003), each x_i value belongs to the receptive fields to different degrees. We also recognise that using only two membership functions in some two-class

problems may not be enough. This fuzzification would be too coarse-grained for multi-class classification problems. Increasing the degree of granularity of the fuzzification will come naturally when the real receptive fields will be mapped to a pre-defined set of membership functions. The authors are aware that it is still necessary to do quite a lot of work to improve the rule extraction technique before ESNN-PC can become a knowledge-based neural network such as EFuNN.

8.8 Conclusion

For the first time, the applicability of spiking neural networks for the problem of taste recognition have been investigated and a special type of ESNN, the ESNN-PC, that consists of simple integrate-and-fire neurons with rank order population coded inputs was proposed for taste recognition (ESNN-PC-TR). The network architecture evolves during the training, accommodating new samples as they become available. The taste recognition system has three layers: an electronic tongue that samples the tastants, a GRF layer with equally spaced Gaussian receptive fields which perform population coding of each of the sensor values, and an ESNN in which the L1 neurons represent taste receptors and the L2 neurons are simple integrate-and-fire units sensitive to the order of the incoming spikes.

Six scenarios were designed to test the proposed model, where an ESNN-PC-TR network was trained and evaluated on small balanced and unbalanced datasets containing samples of 2 and 4 different brands of water and the tests were repeated on larger balanced datasets comprising samples of 2, 4, and 5 different types of wine. It was demonstrated that the ESNN-PC-TR (ESNN-PC) is capable of learning and classifying beverages from small and large datasets with high accuracy.

Population encoding was successfully applied to the rank order coding model. The introduction of population encoding increased the classification accuracy and made the model more biologically realistic. The experiments with the number of receptive fields (m) showed how the accuracy of the taste recognition system changes as a function of m . Representing the input features in a population of receptive field neurons was effective in increasing accuracy, however there is an optimal number after which additional receptive fields no longer increase accuracy and can even cause accuracy to deteriorate. Furthermore, the experiments with the number of neurons in the output layer (L2 neurons) revealed that for a given number of receptive fields, the addition of

neurons can indeed improve performance, but this will not always be the case (Fig. 8.10).

A framework for extracting knowledge from a trained ESNN-PC was proposed. This framework was validated in two scenarios where the knowledge about which data features (sensor measurements) were important for distinguishing between two mineral waters and also between two brands of wine. Our experiments show that ESNN-PC could be used for knowledge discovery in temporal datasets.

Although our intention was not to build a completely realistic model of taste recognition, the proposed model is more realistic than the benchmark models based on the traditional artificial neurons. The integrate-and-fire neurons used in the proposed model are better at modelling the dynamic behaviour of real neurons than the summation units fed with rate coded inputs which are used in traditional networks. Also, the proposed model can learn new tastants without forgetting the ones that it has already learnt. This is a very realistic approach because human beings do not forget all the previously learned tastes as they are introduced to new ones. ESNN-PC is also able to explain its knowledge in the form of IF-THEN rules. Another advantages of the ESNN-PC is that it learns quickly, using one-pass learning algorithms. Furthermore, data is encoded using a temporal encoding scheme that simulates the action potentials occurring in the brain. Finally, the approach used to encode the taste data is supported by several recent findings on the behaviour of taste receptor cells.

A limitation of the proposed model is that the network parameters (mod , m , σ , c and S_θ) are hand-tuned. This has two drawbacks: (i) the manual tuning is time consuming; and (ii) the manual tuning of parameters does not necessarily results in us finding the optimal parameters. Thus further work is required if we want to automate the tuning process. This may further improve network performance and make the model even more biologically realistic.

The survey of published work on the encoding of tastants at the periphery and in the brain showed that regardless of the attention that researchers have given to taste processing the physiology of taste is still not fully understood. Therefore, models of taste processing are less biologically plausible than models of other better understood sense modalities. We believe that the results obtained in the six scenarios simulated in our work contribute to the promotion of spiking neural networks as a possible new solution for taste recognition systems.

Although the proposed system has proven to be suitable when performing the classifications of beverages, further developments are envisaged. In Chapter 9, the details of an FPGA implementation of the proposed model are detailed. A hardware implementation of taste recognition will allow us to build artificial sensory modalities based on spiking neurons in hardware and embed them in autonomous robots. These robots can then be used in situations where a human's presence is not practical. Furthermore, despite the results showing that high accuracies can be achieved using GRF with only one width it would be interesting to explore how the usage of a GRF with different widths for different input variables and/or in different ranges of the input variables can influence the model's performance. Also, instead of using the same number of receptive fields for all features, the number of receptive fields could be tailored to each specific feature. In addition, in this work the steady-state sensor values were available. It remains to be explored how the model would discriminate sensory signals with a time-dependent structure. This could make the use of spiking neurons, capable of spatio-temporal processing even more important. Overall, the framework proposed here provides huge potential for building a multimodal model of taste perception. Therefore, this work should be viewed as the first step rather than an end.

Chapter 9

FPGA implementation of the taste recognition model

A compact and fast hardware implementation of the ESNN-PC-TR would open many possibilities for its application. Hardware implementations of the gustatory model would enable the creation of autonomous robots that have the ability to taste liquid and food as humans do. In turn, these intelligent robots could be employed in situations where a human's presence is not practical. This chapter proposes a hardware implementation of the ESNN-PC model which was proposed in the previous chapter.

To start with, a review of the literature on the FPGA-based hardware implementations of ANN is provided. At a first glance, moving a neural network from a software simulation to an FPGA-based hardware implementation seems like a straightforward exercise. Unfortunately, this is not the case as many features found in software are not available when using hardware. For example, multiplying a number in software is not an issue, but when doing FPGA implementations this becomes the number one issue due to the limited resources in the FPGA chips. Therefore, a description of the challenges faced in FPGA implementation and the techniques to help overcome these problems are presented.

In addition, a detailed description of our FPGA-based implementation of the ESNN-PC model is given. This implementation has many desirable features and these are presented and described in detail. The descriptions are supported with schematic diagrams of all hardware-implemented ESNN-PC units. Special attention is given to the implementation of the on-chip and on-line learning feature, recognising that the network is not only trained on-line but that it is also built on-line. Building a network on-line is a unique approach not often used in implementations which have been published to date.

Further, in order to verify its performances, the FPGA implementation was tested on the real-world taste recognition problem presented in the previous chapter and the results of this test were compared to the software simulation results.

9.1 Literature review

The main advantages of building the FPGA-based implementations of artificial neural networks are the speed of these implementations compared to the speeds achieved in software simulation (Himavathi, Anitha and Muthuramalingam, 2007; Lee and Ko, 2006) and the re-configurability and flexibility of FPGA (Liu & Liang, 2005). However, FPGA implementations are still rare compared to the number of cases where artificial neural networks are simulated in software. A survey in 2005 (Liu & Liang, 2005) concluded that despite the benefits FPGA implementations there are still no standard FPGA-based ANN models.

FPGA implementations of artificial neural networks are linked to a number of issues, such as: data representation, precision, reconfiguration time overheads, limited hardware resources and calculation speed. As the FPGA chips become more sophisticated, the lack of suitable SNN learning algorithms is starting to emerge as the main reason for SNN not being hosted in hardware more frequently (Maguire *et al.*, 2007; Burgsteiner, 2006). Currently, the learning is often performed in a simulation and the final weights are fixed into the FPGA (Schrauwen and Van Campenhout, 2006; Pearson *et al.*, 2005). In (Allen, Halliday and Tyrrell, 2006) a hybrid system is proposed where a SNN with Hebbian learning is hosted in a FPGA chip and a microprocessor based GA evolves the learning rules of the network.

The first FPGA-based implementations of ANN were published in the early 1990s. In 1992, a fully digital connectionist classifier named GANGLION was implemented in an off-the-shelf Xilinx device (Cox & Blanz, 1992). Even though this first FPGA implemented neural network (Zhu & Sutton, 2003) targeted a specific application, it tackled some common issues that arise from such hardware implementations. In (Zhu & Sutton, 2003), the designers approximated and pre-computed the nonlinear activation function and saved its values in a look-up table (LUT) to increase the processing speed and reduce resource usage. Furthermore, they also avoided large 8×8 multipliers and replaced each 8×8 multiplication with two 8×4 multiplications and one addition. The downside of LUT approximations is that they introduce a quantization error (Al-Kazzaz & Khalil, 2008). Acknowledging that FPGA chips have finite resources, a number of techniques are used to maximise the speed of the implementation and minimise the area employment. The main optimization techniques are discussed in the following section.

9.1.1 The main challenges for FPGA implementations of artificial neural networks

The presence of multipliers in artificial neural network architectures has been one of the main limiting factors for their implementation in FPGA. In 1993, M. Marchesi and his team (1993), inspired by the work of White and Elmasry (1992), proposed the power-of-two or binary-radix MLP where real-value weights are approximated by the integer power-of-two values. This allowed them to replace multipliers by shift registers and shift-add operations, and lead to the creation of a multiplier-less neural network. A reduction to the FPGA resources required for the multiplications can also be achieved by encoding input signals into stochastic bit-streams. As a result, the multiplication of two bit-streams can be done with a single 2-input logic gate (Bade & Hutchings, 1994).

Inspired by the idea that multiplier-less neural networks are feasible Hiroomi Hikawa proposed a multilayer neural network with a modified back-propagation algorithm and on-chip learning (Hikawa, 1995 & 1997). In this network, the error signals were represented by pulse signals and the back-propagation algorithm was modified and implemented with pulse-mode operations. The multiplications were implemented with a bit-serial arithmetic. This approach has evolved in the following years through the work of Hikawa (2003 & 1999) and others (Damak *et al.*, 2006; Torres-Huitzil, 2006; Maeda & Tada, 2003; Nedjah & de Macedo Mourelle, 2003). A comparison of stochastic and binary-radix neural network implementations showed that the former were slightly slower but required less resources than the latter (Nedjah & de Macedo Mourelle, 2003). For a medium sized stochastic network, the speed-resource product was an order of magnitude smaller than the product of the same size binary-radix implementation. An experiment on the effect of reducing the bits of a floating-point arithmetic unit from 32 bits to 24 bits reduced the FPGA resource usage by 25% with a 0.32% deterioration in the model accuracy (Lee & Ko, 2006). However, bit reduction from 32 bits to 16 bits saved 50% of the resources but caused a 55% deterioration in the accuracy.

Often moving a network from a software simulation to an FPGA hardware implementation requires a change from floating-point to fixed-point precision. Using shorter integers consumes less FPGA resources but often results in a loss of precision (Savich, Moussa & Areibi, 2007), and therefore trade-offs between accuracy and the utilised area are often necessary. However, an experiment on the influence of limited fixed point precision on the accuracy of a back-propagation multilayer neural network trained on a number of benchmark datasets found that the classification accuracy can be

affected by the size of the hidden layer rather than the floating-point to fixed-point transition (Holt & Baker, 1991). The size of the multipliers can be kept down by reducing the internal resolution to 12-bit (Schrauwen & Van Campenhout, 2006; Hikawa, 1995) or 8-bit precision (Himavathi, Anitha & Muthuramalingam, 2007; Labib *et al.*, 2005). Although more than one internal precision might be required to overcome the quantization distortions of some parameters (Ševčík, 2006).

Activation functions are also often simplified (Larkin *et al.*, 2006; James-Roxby & Blodget., 2000; Hikawa, 1995) or replaced by a new function to keep the resource usage low. The use of the sinusoidal activation function decreased hardware resource employment by about 32% compared to the employment when using the sigmoid neuron implementation (Vitabile *et al.*, 2005), and using a spline-based activation function approximation can save around 46% of the hardware resources (Larkin *et al.*, 2006). Activation function values are often stored in the LUT (Himavathi, Anitha & Muthuramalingam, 2007; Chalhoub, Muller & Auguin, 2006). Furthermore, various optimization techniques can be used in order to achieve an optimal implementation (Chalhoub, Muller & Auguin, 2006).

In time-multiplexed architectures, resource limitations can be overcome at the expense of speed. In these architectures a single neuron or a layer of m neurons with one or n synapses are implemented at any given time and the connections between the neurons or between adjacent layers are time-multiplexed (Vitabile *et al.*, 2005; Pearson *et al.*, 2005; Eldredge & Hutchings, 1994). Only the resources needed to implement one or m neurons are engaged and other neurons use the same resources later on, resulting in a lower percentage of chip usage. The downside is that the time-multiplexing required in those implementations may be very time consuming. Implementing a single layer in an 8-5-5-5-3 MLP saved 50% in resources but introduced a 17.7% speed overhead (Himavathi, Anitha & Muthuramalingam, 2007). Pipelining has been used to compensate for speed overheads introduced by the time multiplexing of connections between neurons in the same or different layer. Various network parameters and data are saved in registers or in internal or external memory. Pipelining allows for processing of later neurons before finishing the earlier ones. In an implementation of a pulse coded neural network pipelining throughput was improved from a little more than a million neuron iteration per second to 55 million neuron iterations per second (Waldemark, *et al.*, 2000). FPGA's limited but in-situ reconfigurable resources inspired Luiz Brunelli *et al.* (2005) to improve resource utilization by an approach they named Execution

Patterns. In this approach, data is at fixed locations and processing elements are dynamically instantiated next to the data when required by the processing algorithm. This approach resulted in a saving of 80% of the reconfigurable interconnect resources in respect to the traditional approach.

FPGA devices with their in-situ re-programmability, ability to work in parallel and high-speed processing seem a logical tool for building large neural networks with neurons that have changeable synaptic weights. Furthermore, a recent comparison study examining the speed of pure software and pure hardware implementation of the multi-layer back-propagation network found that the pure hardware architecture on FPGA was around 500% faster than the pure software implementation (Al-Kazzaz & Khalil, 2008). The ever-increasing types of FPGA devices and their speeds suggest that it is very likely that the problems researchers reported in the past will become obscure in the future. The number of multipliers in an Altera family of FPGA devices is shown in Table 9.1. For example, an EE4SE360 has 4506 various multipliers including 260 single-precision and 140 double-precision floating-point multipliers.

TABLE 9.1

The number of multipliers in the Stratix IV E FPGA DSP family devices (Altera).

FP – floating-point

Device	MULTIPLIERS							Single FP	Double FP
	9×9	12×12	18×18	18×36	36×36	18×18 complex			
EP4SE110	512	512	512	256	128	128	128	51	
EP4SE230	1288	1288	1288	644	322	322	322	128	
EP4SE290	800	800	800	400	200	200	200	80	
EP4SE360	1040	1040	1040	502	260	260	260	104	
EP4SE530	1024	1024	1024	512	256	256	256	102	
EP4SE680	1360	1360	1360	680	340	340	340	136	

9.1.2 Learning in FPGA

Learning is an important feature of all ANN. Two setups exist for FPGA implementations: one including an on-chip training facility (Zhuang, Low & Yau, 2007; Damak *et al.*, 2006; Maeda & Tada, 2003; Hikawa, 1995; Eldredge & Hutchings, 1994) and one without an on-chip training facility (Pearson *et al.*, 2007; Schrauwen & Van

Campenhout, 2006). In implementations without on-chip learning the network parameters (including weights) are first optimised in software and then frozen in an FPGA during the programming phase (Merchant *et al.*, 2006; Schrauwen & Van Campenhout, 2006). This enables higher processing speeds and lower resource costs but comes at a cost because the network is unable to retrain in-situ for new data.

An on-chip learning mechanism can occupy as much resource as the network itself and this can be impractical for a very large neural network. Therefore, Eldredge and Hutchings (1994) suggested that the on-chip learning hardware can be implemented during the training in the same area as occupied at other times by the trained network. Availability of soft processor cores within a single FPGA enables embedded hardware and software co-design. A soft core processor can be used to run a GA for on-line evolution and learning (Merchant *et al.*, 2006; de Garis & Korkin, 2002) and as an interface between the FPGA and the rest of the system (Roggen *et al.*, 2003). However, running GA has a high cost and therefore this type of learning might not be appropriate when the processing speed is crucial, such as in some real-time applications.

9.1.3 FPGA implementations of spiking neural networks

Whilst implementations of classical neural networks on FPGA continue there are more and more papers addressing the issues of FPGA implementations of spiking neural networks, predominantly implementations of large networks of IF neurons on a single FPGA device. Implementations of SNN often target a specific task. A network of spiking neurons was designed for use on mobile robotic devices (Pearson *et al.*, 2007, Roggen *et al.*, 2003), for image processing (Waldemark *et al.*, 2000) and for hardware modelling of neuronal ion channel dynamics (Mak *et al.*, 2005). Schrauwen and Van Campenhout (2006) implemented a LIF model with various synapse models using parallel processing and serial arithmetic. They used pipelining and 12 bit precision to optimise hardware usage and built the networks on a small, low cost FPGA and on a state-of-art FPGA. Whilst the size of neural network increased by 2500% through the use of the state-of-art FPGA, the speed only doubled. They also highlighted the fact that it is often hard to compare space and speed numerical values between different implementations because usually different FPGA devices are used.

As multipliers still limit the size of FPGA implementations of neural networks, including implementations of spiking neural networks, multiplier-less approaches are still investigated and proposed (Zhuang, Low & Yau, 2007). Other techniques used to

optimise the classical neural networks are also used to optimise spiking neural networks. For example, time-multiplexing and fixed-point integer arithmetic was studied in (Pearson *et al.*, 2005) and later extended in (Pearson *et al.*, 2007) where a spiking neural processing element was designed for the control of a mobile robot. They implemented the same network in software and hardware and compared the responses of these implementations on repetitive presentations of the same input stimuli generated using randomly seeded generators. The observed correlation between responses was 0.7662. The discrepancies between the two responses were found to be mainly due to the differences between the random number generators in the two systems. Custom-built simulation tools (Hellmich & Klar, 2004) have supported FPGA prototyping.

The lack of FPGA resources is a significant limiting factor when building large neural networks comprising thousands of neurons. McGinnity *et al.* at the University of Ulster have been conducting research in the area of SNN focusing on FPGA implementations of large-scale SNN. In their 2005 paper, they proposed a platform for the implementation of large scale SNN on FPGA devices based on a time-division multiplexing strategy (Glackin *et al.*, 2005). In this scheme, a multi processor system is built where each soft-core processor is responsible for processing a portion of the network's neurons. They showed that networks containing 4200 IF neurons with 1964200 synapses were feasible. They also introduced a multiplier-less strategy and found that the network size that can be implemented on an FPGA chip drops considerably with the neuron to synapse ratio (Maguire *et al.*, 2007). The major bottleneck was the limited availability of multipliers.

9.2 FPGA implementation of the ESNN-PC-TR model

This section proposes a new FPGA implementation of the evolving spiking neural network for taste recognition introduced in Chapter 8. This network has also been simulated in software and applied to taste recognition problems as explained in Chapter 8. The classification results showed that the proposed approach is suitable as an approximation of the gustatory system. Here, an FPGA-based implementation of the same taste recognition model is presented and compared to the software simulations in terms of their processing speed and classification accuracy. The complete ESNN-PC-TR system, including the population encoder, rank order coder and a network of spiking neurons, was implemented on a single FPGA (Fig. 9.1). Furthermore, the hardware required to build and evolve the network and the network itself was hosted on the same

chip. However, the implementation relies on a host PC to deliver data samples for evolving the spiking neurons and, once the network is trained, for the classification of unseen data samples.

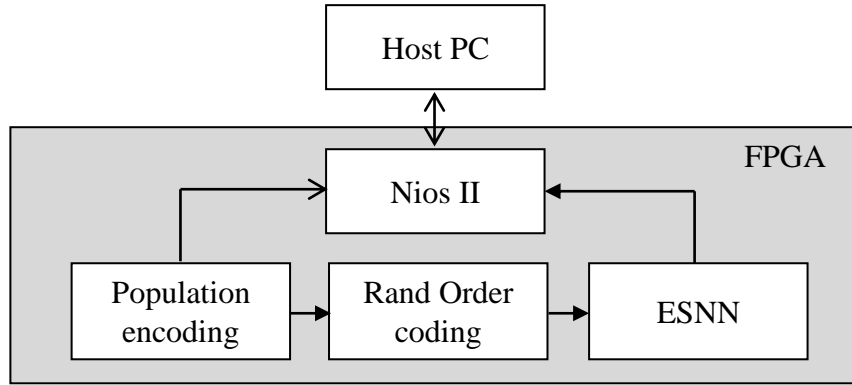


Fig. 9.1. Block diagram of the proposed FPGA implementation of the ESNN-PC-TR.

The implementation was based on an Altera Cyclone II EP2C35F672C6 FPGA on an Altera NIOS II Development Kit. The Cyclone II FPGA chip provides an on-chip memory, a NIOS II soft processor core and interfaces to the on-board chips (Fig. 9.2). The board was chosen because of its availability and state-of-the-art features. The development board was connected to a host computer via a JTAG USB-Blaster and RS232 serial cable. A NIOS II soft processor core was used as well. The operating frequency of the FPGA model was 50 MHz ($t_{CLK} = 20$ ns). This development board is supported by the Quartus II (version 6.0, build 202, SP1), SOPC Builder (version 6, build 202) and NIOS II IDE (version 6.0, build 93.3, SP1) software from Altera.

The utilization of FPGA resources has been kept down by a careful optimization of the word length across the whole implementation. A 32-bit fixed-point mixed integer-fractional format was chosen for this FPGA implementation. The Q1.30 number format (one integer bit and 30 fractional bits, in a range of $[0, 2)$) was used for normalised sensory data, population encoded data features and synaptic weights, and the Q7.24 (7 integer bits and 24 fractional bits, in a range of $[0, 128)$) was used for PSP and PSP_{θ} . Using minimal resources lowers the power consumption. While the main aim was ensuring that resource consumption was as low as possible we made sure that numeric fidelity was not compromised.

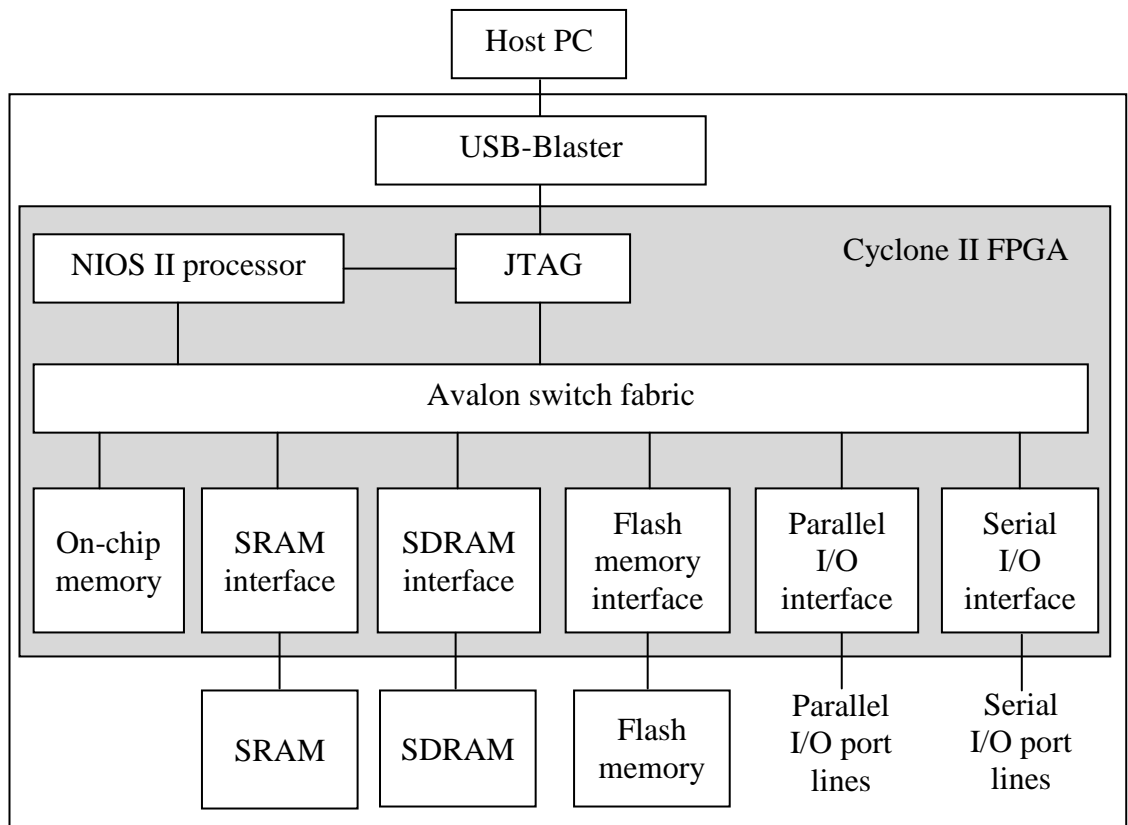


Fig. 9.2. A simplified block diagram of Cyclone II FPGA: NIOS II soft processor core and the interfaces needed to connect to other chips (Altera).

9.2.1 Embedded processor system

A NIOS 16-bit embedded processor was instantiated alongside the ESNN-PC-TR neural network. The processor used a set of 16 and 32-bit I/O ports to control and interact with the neural network. The processor was responsible for loading sensory data samples to the population-encoding module and retrieving spike times from the spiking network. The data was received from and the spike times were sent to the host computer via a serial link. Furthermore, during testing the processor was used for retrieving intermediate data from encoding and ranking modules as well as the values of all network parameters. Once the network was functional the processor's tasks were scaled down to conveying data samples from the host PC to the system and issuing control signals to insure correct functioning of the system's modules.

9.2.2 A novel FPGA implementation of population encoding

In our novel FPGA implementation, normalized sensory data first undergoes population encoding using the module shown in Fig. 9.3. The Gaussian values required to encode

the input sensory data are saved in a look-up-table (*i.e.* the Gaussian Curve LUT). Only a half of one Gaussian receptive field values were saved and used to produce the values for m receptive fields. This was possible because all Gaussian curves used were symmetrical about the centre and had the same width. Other values were calculated by subtracting and adding the receptive field centre value c_i (Table 9.2) from the sensory data value and presenting the difference to the Gaussian LUT. In this work the centre values c_i were determined by (8.1) – (8.4). Also, an out-of-range detection circuitry is used to detect sensory data features too small to contribute to the membrane potential of the L2 neurons. The features that exceed three standard deviations from c_i are set to zero, and consequently L1 neurons conveying these values do not spike.

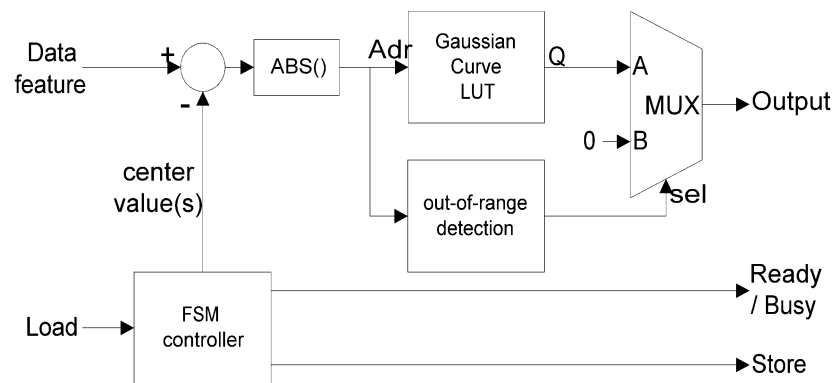


Fig. 9.3. The block diagram of the population encoding module. The encoder has an out-of-range detection circuitry to detect sensory data features too small to contribute to the membrane potential of the L2 neurons. (Zuppichich & Soltic, 2008)

TABLE 9.2

Centre values c_i for Gaussian receptive fields ($m = 6$ and $m = 8$).

m	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
6	-0.75	-0.25	0.25	0.75	1.25	1.75	-	-
8	-0.66	-0.33	0	0.33	-0.66	1	1.33	1.66

The sensory data features are received via the embedded processor. The processor also issues a ‘Load’ signal signalling that a feature is ready to be encoded. The population encoding module uses a ‘Ready/Busy’ signal to notify the processor when it is ready to receive another feature for encoding. When the feature is encoded, a ‘Store’ signal is sent to the rank order module informing it that the data is available to be received. A description of the ranking module used is given in the next section. The population

encoding module provides one Gaussian value for every clock cycle ($t_{CLK} = 20$ ns) and a feature is fully encoded every $m t_{CLK}$ or every $0.12 \mu\text{s}$ ($m = 6$) and $0.16 \mu\text{s}$ ($m = 8$). It takes the population encoding module $m t_{CLK} s$ seconds to encode one data sample comprising s features (where s is the number of sensors used to taste tastants). However, due to the delay introduced by the processor $t_{dCPU} = 3 \mu\text{s}$ the whole process of encoding the data slows down considerably to $s (m t_{CLK} + t_{dCPU})$ seconds. While the total time needed to encode one data sample with $m = 6$ is $1.12 \mu\text{s}$, when the delay caused by the processor is included this time increases to around $22 \mu\text{s}$.

The number of points used in modelling Gaussian curves was critical to the taste recognition system's classification accuracy. An error at this processing stage could potentially decrease the accuracy of the ranking and consequently the correct classification of sensory data samples. While higher resolutions are desirable, the available FPGA resources limited the implementations. Here, a 22-bit Gaussian LUT containing 8192 pre-calculated Gaussian values was implemented. The step size between two values was $1.2e-4$ with a maximum error of 0.12%. This design allowed us to achieve accuracy comparable with the accuracy of the software implementations. The population module can be further optimised by interpolating missing values. Unfortunately, this would compromise the resource usage and processing speed of the implementation.

9.2.3 A novel FPGA implementation of the rank order coder

The data from the population-encoding module is sent to the rank order coding module for ordering shown in Fig. 9.4. The data components with higher values represent the earlier pre-synaptic spikes which are sent to the neural network before the smaller values which represent the later spikes. A highly parallel bit-wise rank coder was implemented using an N -stage 32-bit shift register and a finite state controller (FSM controller). When all N values are received from the population-encoding module and loaded into the shift register the processor issues a 'Start' signal and the ranking begins. The FSM controller runs a top-down search through the bit patterns. The rank coder produces the first values after 0.06 to $3 \mu\text{s}$, while the ranking lasts anything between 6 and $16 \mu\text{s}$. The network informs the ranker that it has processed the current spike and that it is ready for a new one through an 'Ack' signal. The rank order provides the rank of a synapse 'Index' together with a 'Valid' status signal to the neural network module. These 'Valid' signals inform the network of the presence of a valid incoming spike. A

‘Done’ status signal is sent to the processor once all Gaussian values have been ranked and the ranker is ready to start receiving a new set representing another sensory data sample. By that time, the neural network has also processed all incoming spikes. During training, the neural network processes all incoming spikes but during classification the processing of spikes stops once the first L2 neuron spikes, signalling that this sample has been classified. This detection of the first spikes results in fast classifications.

9.2.4 A scheme for an FPGA implementation of the spiking integrate-and-fire neuron and its synapse

Integrate and fire neurons used in this work are specified by the membrane equation (7.10) where the membrane potential of $L2_i$ (i.e. PSP_i) at time t depends on the firing orders of all its pre-synaptic neurons. The calculation of the membrane potential of one neuron consists of three operations: rising to the power, multiplication and summation. The influence of an incoming spike via synapse j is calculated by multiplying the corresponding synaptic weight (w_{ji}) and modulation factor (mod) risen to a power dependent on the order of the incoming spike ($order_j$). The influences of the spikes coming from all the pre-synaptic neurons via their synapses are then added together. Given that the number of multipliers in an implementation must be kept to minimum, the multipliers required to perform the power calculations were avoided by storing the mod^{order} values into a look-up table. As a result, the calculation of PSP values was reduced from the power-multiply-and-add operations to one multiply-and-add operation per synapse. However, multipliers could not be easily circumvented altogether without sacrificing the processing speed.

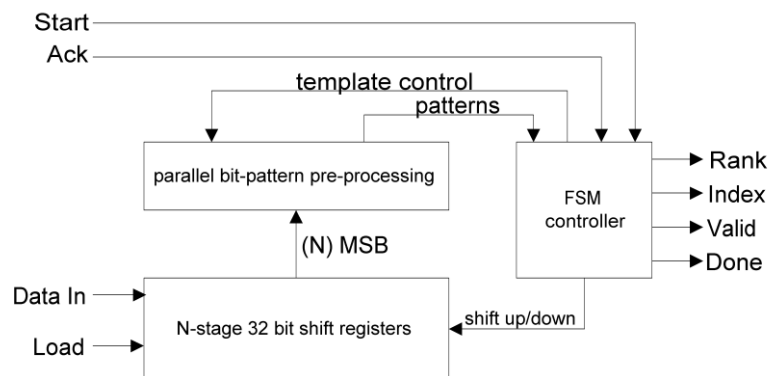


Fig. 9.4. The block diagram of the proposed parallel bit-wise rank order coding module based on an N-stage 32-bit shift register and a FSM controller. N is equal to $m \times s$, where m is the number of Gaussian receptive fields and s is the number of features per sensory data sample. (Zuppichich & Soltic, 2008)

The proposed hardware implementation of the spiking neuron and its synapse is shown in Fig. 9.5. Note that the learning circuitry is not shown, as it is described in detail in the next section. The neuron operates within a discrete time domain with intervals defined by *order*, where $order \in (0, N-1)$ and $N = m \times s$ is the number of synaptic inputs. The PSP and PSP_{θ} values are held in the on-chip memory (the PSP memory block and the Threshold memory block). At each interval t_{order} the neuron is presented by the ranker with a ‘Rank’ and index ‘Synapse Index’ values signalling the rank and the synapse of the incoming spike. The values for mod^{order} and w_{ji} are taken from the look-up tables, multiplied and added to the post-synaptic potential PSP_i of this neuron. If the neuron’s PSP_{θ_i} is exceeded, the neuron produces a spike. The process is repeated for all feature values. While it is important to process all features to find PSP_{imax} during the neuron creation, during the classification the processing is stopped after the first spike is detected.

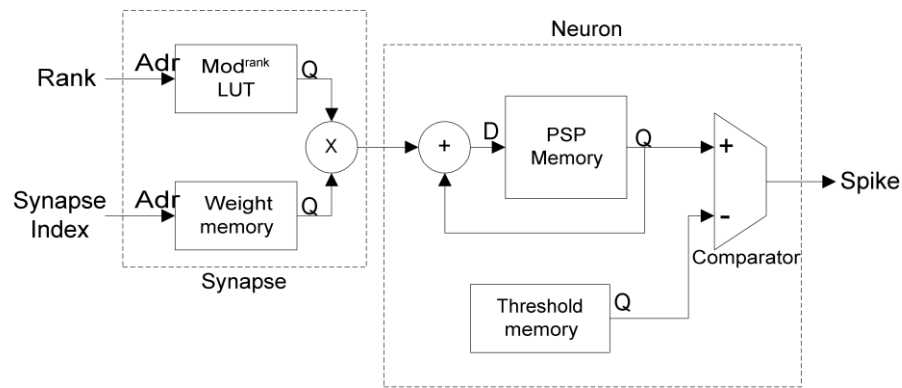


Fig. 9.5. The block diagram of the proposed circuit for a spiking neuron and its synapse. Note that the circuitry required for the on-chip learning is omitted. Synapse weight values and the weight contributions are saved in two LUT. The neuron is modelled using two memories and a comparator. Note that there is one multiplier in this module. (Zuppichich & Soltic, 2008)

9.2.5 A method for an FPGA implementation of on-chip learning

In ESNN-PC all L2 neurons are created during the evolving process. As a L2 neuron is created it learns to recognise samples from a class. One L2 neuron can learn a number of samples and is able to classify correctly a previously unseen sample. On-chip learning is implemented using two modules. One module compares the newly created neuron to the pre-existing neurons $L2_o$ for the same class of data samples (Fig. 9.6) and one module aggregates the newly created neuron if the similarity is found to be less than the network parameter S_{θ} (Fig. 9.7).

The block diagram in Fig. 9.6 shows the distance measure module. The weight values and the sum are each in the range of $[0, 1]$ and $[0, m \times s]$ respectively, where $m \times s$ is the number of L1 neurons, m is the number of Gaussian receptive fields and s is the number of input features. As the comparison starts, a single weight w_i of the newly created $L2_i$ is placed on the 'Weight' input and a 'Store' signal is issued. Then the weight of an existing $L2_o$ is placed on the 'Weight' input and the two values are subtracted, squared and added to the current distance. This process is repeated for all weight values. Once all weights are processed, the total distance is compared to S_θ . It takes this module $5 t_{CLK}$ to process one synapse and $(5 m s + 2) t_{CLK}$ to compare one $L2_i$ to one $L2_o$. This includes 2 overhead t_{CLK} per neuron. If the number of L1 is 42, one synapse is processed in 100 ns and two L2 neurons are compared in 4.24 μ s. If there are 56 L1 neurons 2 neurons are compared in 5.64 μ s.

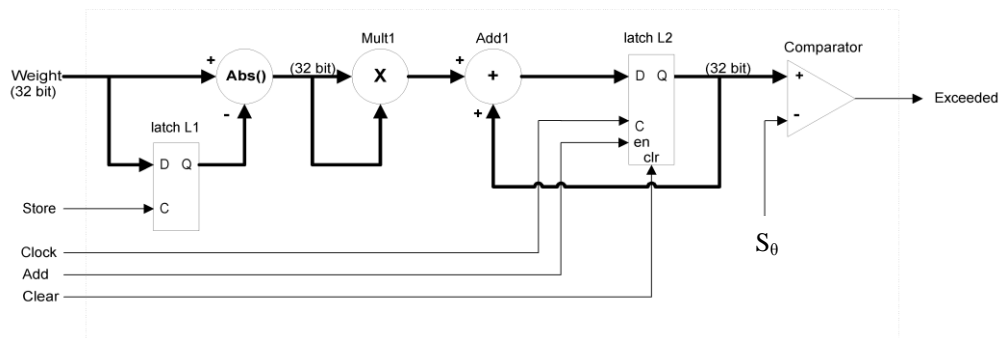


Fig. 9.6. The block diagram of the proposed distance measuring module that compares a newly created neuron against all pre-existing neurons for the same class of data. Note that there is only one multiplier in this module. (Soltic & Zuppich, 2008)

A block diagram of the aggregation module is given in Fig. 9.7. If $L2_i$ is found to be similar to $L2_o$, these two neurons are aggregated. The new weights and the threshold are set to the arithmetic averages of $L2_o$ and $L2_i$. It takes $10 \times t_{CLK}$ to process one synapse or 11.2 μ s when there are 56 L1 neurons. The aggregation of the neurons' thresholds is a process similar to that of the aggregation of the weights and therefore two instances of this module are created, one for calculating the new threshold and one for calculating the new weights. The two modules work in parallel during the update of the weights.

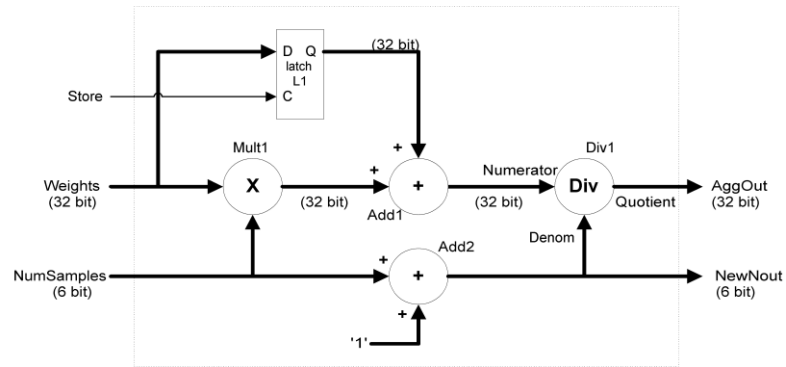


Fig. 9.7. The block diagram of the proposed aggregation module. Note that there is only one multiplier in this module. (Soltic & Zuppichich, 2008)

9.2.6 A discussion on the size of ESNN-PC-TR when implemented on an FPGA

FPGA implementations of ANN are limited by the availability of FPGA resources. Clearly, the problem of limited resources is even more profound when an evolving architecture is considered. In an evolving neural network, the number of neurons depends on the characteristics of the input data and the values of the network's parameter. An evolving neural network may create one neuron per sample and if there are many samples the network may grow very big. While, it would be great to have an FPGA that is sufficiently large to allow the network to evolve freely the reality is that this is currently impossible. Therefore, the number of neurons that are allowed in a network must be kept under control despite the impact that may have on the accuracy of the network.

In this thesis, the proposed implementation of ESNN-PC-TR is based on the concepts of virtual neurons and virtual synapses, namely only a single neuron with one synapse is instantiated. The neurons' parameters (PSP , PSP_{θ} , class label and the number of aggregated neurons) are saved in the memory and a finite state machine addresses this memory space using a 6-bit address bus. This particular design allows the ESNN-PC-TR network to evolve to 64 neurons with up to 64 synapses each, *i.e.* have a total of 4096 synapses, on the Altera Cyclone II EP2C35F672C6 FPGA chip.

9.3 Implementation results

One important implication of the limit to the number of neurons in the network was that we were not able to repeat all six scenarios designed to test the ESNN-PC-TR. This

limitation also meant we were unable to directly compare the performances, advantages and disadvantages of the software and FPGA implementations on all six data sets. However, it was possible to test Scenarios 1, 2, 3 and 4 shown in Chapter 8 Table 8.3. The test showed that the accuracies obtained with the FPGA implemented ESNN-PC-TR were comparable to the accuracies achieved with the software simulations (Table 9.3). Unfortunately, Scenarios 5 and 6 could not be repeated with the FPGA implementation because the number of required Gaussian receptive fields was far too high for the FPGA chip's resources.

Given that processing speed is one of the key drives of hardware implementations of ANN, special attention was given to the timing analysis of both implementations. Acknowledging that an increase in processing speed cannot be achieved without employing additional resources, FPGA resource usage and operating speeds are discussed in the next section.

TABLE 9.3

The accuracies of the FPGA and software implementations of ESNN-PC-TR when used for water and wine recognition.

Scenario	Class	Samples	Classes	Software (%)	FPGA (%)
1	Water 1&3	12	2	100	100
2	Water 1&2	10	2	100	100
3	Water 1-4	40	4	91	87
4	Wine 2&3	300	2	100	100

9.3.1 FPGA resource usage and operating speed

In order to minimise FPGA resource consumption the number of multipliers was kept to a minimum and the evolving networks were built around the virtual neuron concept. While instantiating only one neuron with one synapse and the use of a single learning circuit kept the number of multipliers low, *i.e.* down to 23% of the available multiplexers, the memory usage was substantial as shown in Table 9.4, 76% of the memory space was employed in this design. Half of all memory space was used by the population-encoding module and its LUT. It is interesting to see that the processor itself

uses only 11% of the memory. On the other hand, the on-chip learning circuit uses 14% of the multipliers and 4% of the logic elements.

TABLE 9.4

Available FPGA resources and their usage.

Resources	Available	NIOS	POP	Ranking	Neuron	Learning	%
Logic elements	33216	7000	252	5000	1200	1460	45
Memory(kbits)	483	52	180	0	138	0	76
9×9 Multipliers	70	4	0	0	2	10	23

As expected the FPGA implementation was faster than the software simulations. In order to compare the speeds of both implementations the same set of data samples and network parameters were used, the experiments were repeated 10 times and the training speeds averaged. While the software simulations were run on a 2 GHz Intel Core 2, the FPGA chip was clocked at a much slower frequency of 50 MHz. We must bear in mind that the processing speed of the FPGA implementation was influenced considerably by the delivery of data samples.

We examine how long it takes to evolve an ESNN-PC-TR. Table 9.5 show the times that the software and FPGA implementations require to train ESNN-PC-TR on one sensory data sample. It takes the NIOS processor around 20 μs ($t_{dCPU} \approx 20 \mu\text{s}$) to deliver one sensory data sample with 7 features compared to the 20 ns required to encode one data feature, or 1.12 μs to encode all 7 features with 6 Gaussian receptive fields ($m = 8$, 56 L1 neurons). The first value is provided by the ranking unit after 0.06 to 3 μs and all values are ranked after 6 to 16 μs . A neuron is created after 30 to 70 μs . In total it takes 40 – 80 μs to create and train one L2 neuron on a 56-dimensional data sample. This is 10 to 25 times faster than in software simulations of the same network ($t_{\text{soft}} = 1040 \mu\text{s}$)

Both type of implementations are faster during the classification than during the learning phase as there is no need to evolve the network and train the L2 neurons. Also, the processing of one data sample stops when the first L2 spikes are detected. A sample is classified within 6 μs by the FPGA implementation (Table 9.6). It takes 445 μs by the software. The time required to population encode one sample by hardware is the same as during the training. We can see from Table 9.6 that the FPGA implementation classifies one sample 74 times faster than the software tool.

TABLE 9.5

Training times for software (t_{soft}) and FPGA (t_{FPGA}) implementations required to train an ESNN-PC-TR network on one sensory data sample (56 L1, $t_{dCPU} \approx 20 \mu\text{s}$) (Soltic & Zuppichich, 2008)

	$t_{\text{soft}} (\mu\text{s})$	$t_{\text{FPGA}} (\mu\text{s})$
Population encoding	123	$1.12 + t_{dCPU}$
Ranking	320	6 - 16
First ranks available	320	0.06 - 3
Training	600	30 - 70
Total time	1040	40 - 80

TABLE 9.6

Classification times for software (t_{soft}) and FPGA (t_{FPGA}) implementations required to classify one sensory data sample (56 L1, $t_{dCPU} \approx 20 \mu\text{s}$) (Soltic & Zuppichich, 2008)

	$t_{\text{soft}} (\mu\text{s})$	$t_{\text{FPGA}} (\mu\text{s})$
Population encoding	123	$1.12 + t_{dCPU}$
Ranking	320	2
First ranks available	320	0.06 - 3
Classifying	105	2
Total time	445	6

9.3.2 Control signals and timing

The rank order coder's control signals and their timing during the network evolving stage is shown in Fig.9.8. Please note that time durations are not to scale. As stated earlier, the rank order coder uses four handshaking signals to communicate with the rest of the system. The ranking starts when a 'Start' signal is received. Due to its design, the ranker can produce the first spike value ($order_j = 0$) after as little as 60 ns. Once the first spike is ready the neural network can commence with the processing. Every time the rank order coder finds the next highest value (*i.e.* spike) it issues a 'Valid' signal and provides the order ($order_j$) and the index (j) of that spike. As the ranking progresses, the time intervals between two 'Valid' signals became shorter and shorter as there are fewer spikes to rank. While the first spike can be ready for processing after around 600 ns, the time required to find the next spike decreases to less than 100 ns. It takes the network $8T_{CLK}$ or 160 ns to process one spike. In other words, the network is ready to receive a

new spike for processing every 160 ns. An ‘Ack’ signal is sent to the ranker from the network acknowledging that it is ready to receive a new spike. Without aggregation, it takes $(8 m s + 4)t_{CLK}$ to create one L2 neuron. For example, a neuron with 56 synapses is created in 15 to 25 μ s. However, a possible aggregation introduces a further delay and one L2 is created and trained within 40 to 80 μ s. When the last synapse is being processed, the ‘Ack’ signal is delayed to allow for possible L2 aggregations.

The control signals and their timing during the classification stage are shown in Fig. 9.9. The time taken by the network to process one synapse depends on the number of L2 neurons, i.e it is $5 n t_{CLK}$, where n is the number of L2 neurons. Therefore, the time to process all synapse is $((5 n + 2) m s + (2 n))t_{CLK}$, where $m \times s$ is the number of synapses, m is the number of Gaussian receptive fields and s is the number of taste sensors. The formula includes the time t_{CLK} required to acknowledge that all L2 neurons have been processed and to reset the *PSP* values to zero so that the network is ready to start classifying the next data sample.

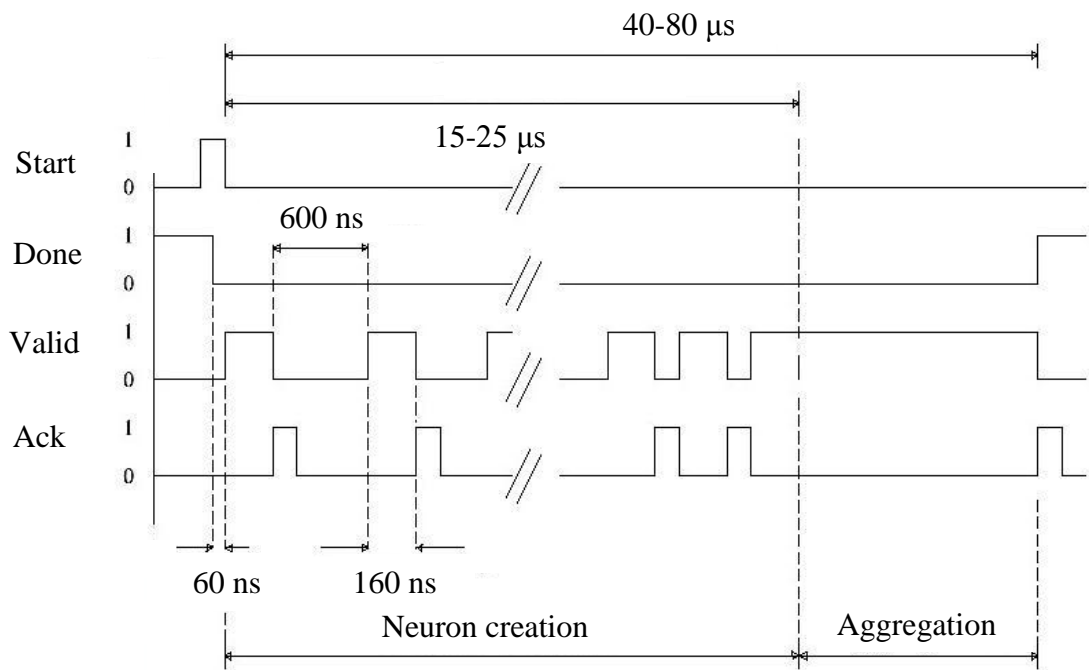


Fig. 9.8. Signals and timing during the L2 evolving stage. One L2 is created in 15 to 25 μ s but a possible aggregation introduces a further delay and one neuron is created and trained in 40 to 80 μ s. (Soltic & Zuppich, 2008).

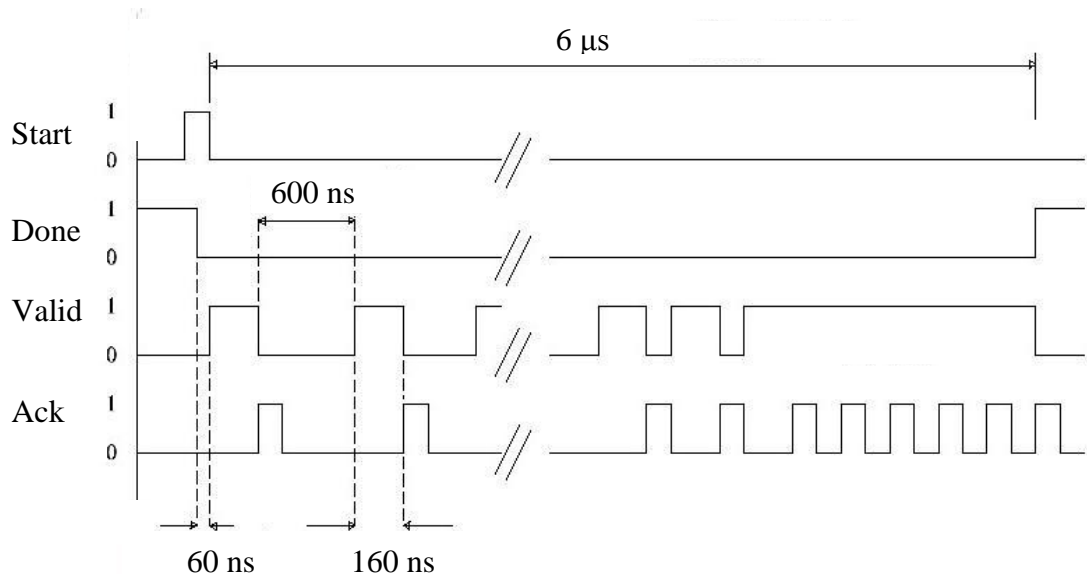


Fig. 9.9. Signals and timing during the classification stage. A sample is classified in 6 μs (Soltic & Zuppicich, 2008).

9.4 Conclusion

The evolving spiking neural network for taste recognition proposed in Chapter 8 has been successfully implemented on an Altera Cyclone II FPGA. The complete ESNN-PC including the population encoder, rank order coder and network of spiking neurons, together with its learning was implemented on a single FPGA. This is a big step towards a hardware representation of the human gustatory system. A NIOS embedded processor was instantiated alongside the ESNN-PC model. Its main function was to load sensory data samples, retrieve results, and control and monitor the operation of the system.

During the design of the FPGA implementation, the focus was on minimizing hardware utilization without slowing down the neural processing and compromising the classification precision. To achieve this, a virtual neural based architecture was used where only one neuron with one synapse was implemented and all other neurons used the same resources. Furthermore, the whole system was implemented around look-up-tables, minimising the need for multipliers and logic elements. The precision was set to 32-bit by using two 32-bit fixed-point mixed integer-fractional formats (Q1.30 and Q7.24). This design allowed evolving networks of up to 64 integrate-and-fire neurons with up to 4096 synapses on the Altera Cyclone II EP2C35672C6 FPGA. The implemented network used 45% of the logic elements, 76% of the memory and 23% of the dedicated multipliers. The on-chip learning circuitry used 4% of the logic elements and 14% of the multipliers.

The proposed FPGA implementation of ESNN-PC was successfully applied to the classification problem of taste recognition. The performance of the FPGA implementation was compared to the performance of the software implementation of the same network architecture. The experimental results have shown that the FPGA implementation was 10 to 25 times faster during the evolving of an ESNN-PC network and 74 times faster during the classification of tastants than the software simulations that had similar accuracies. The on-chip algorithm created and trained one L2 neuron within 30–70 μs compared to the 600 μs required for training by the software implementation. The FPGA implementation relies on a host PC to deliver data samples.

Obtaining data samples directly from the sensors could not only reduce processing speed but also open up the possibility of using the proposed ESNN-PC architecture for taste recognition tasks in an autonomous robot.

Chapter 10

Conclusion and future directions

10.1 Research summary

In this thesis, two generic computational intelligence methods were developed and then applied to two ecological problems.

First, a new adaptive model for risk prediction based on local probability and named the local probability adaptive model (LPAM) was proposed. The LPAM is characterised by its dynamic structure, knowledge discovery facility and incremental learning. The model was first evaluated on two benchmark problems. It was then successfully employed to model the risk of pest insects' invasions, to capture and explain the relationships between the insects' distributions and climatic characteristics of their habitats and to prepare species distribution risk maps. A simple technique for the visualization of the rules extracted using the LPAM based on the information contained in the model was also proposed.

This thesis described a variety of issues related to ecological modelling, in particular when dealing with predictive modelling of species distributions:

1. There is a lack of good quality benchmark data on species distributions;
2. Typically, no knowledge usually exists about the quality of the modelling data;
3. Absence data is hard to validate, *i.e.* the absence data is often more noisy than the presence data, while the volume of noise in the absence and presence data is unknown;
4. There is a lack of knowledge about the characteristics of invasive species, *i.e.* what variables influence the distributions of these species;
5. The relationships of the predictive variables in ecology are non-linear, complex and unknown;
6. The modelling techniques typically used to model species' distributions, *e.g.* MLP or SOM, do not reveal the relationships between the predictor variables and the prediction. In other words, the predictions can be very accurate but one still cannot deduce the influence of each predictor variable on the modelled output.

Our dynamic system, LPAM, uses DENFIS and comprises a noise cancelling module. The DENFIS module delivers species-habitat relationships while the noise cancelling module reduces the influence of noisy data on the prediction accuracy. We suggest that adopting the LPAM could lead to acquiring knowledge that is impossible to extract using the traditional neural network models.

In addition, we have examined the suitability of transductive modelling for building predictive models in ecology. The performance of ‘personalised’ models based on transductive reasoning was compared to the performance of local and global models built using inductive reasoning. For this purpose, we used the recorded distributions of three pest insects (*Aspidiella hartii*, *Geococcus coffeae* and *Xyleborus perforans*) that have a high economic importance to New Zealand. The datasets had a rather moderate number of samples suggesting that ‘personalised’ models would be more accurate than global and local models. Our experimental analysis showed that when measured in terms of Cohen’s kappa statistics (κ), the personalized models build for *A. hartii* were the best of all other models over all threshold values (achieving an excellent accuracy, $\kappa = 0.85$). The transductive approach was less accurate when predicting the establishment potential for *G. coffeae* and the least accurate when predicting the distribution for *X. perforans*. However, none of the studied approaches were particularly successful in modelling the distribution of those two insects. In particular, all models had a problem modelling the distribution for *X. perforans*. According to our experiments, the suitability of a method is highly dependent on the relationships being modelled. Therefore, we suggest that an integrated approach using global, local and ‘personalised’ models in a hybrid system must be used for an improvement of ecological models.

Second, the ESNN-PC, an evolving spiking neural network based on the ROC integrate-and-fire types of neurons with population-coded inputs was proposed. For the first time, the Gaussian receptive fields were employed in a ROC network. The ESNN-PC was then successfully used for taste recognition (ESNN-PC-TR) where different types of wine and water were classified. The ESNN-PC-TR was implemented in software and on an Altera FPGA. Both implementations were evaluated in terms of accuracy and processing speed. The results of our experimental analysis demonstrated that the ESNN-PC-TR model of taste recognition is capable of learning to distinguish and classify beverages from small and large datasets with a high degree of accuracy and of knowledge discovery from the data samples used to train it.

In this thesis, we described the issues related to modelling taste recognition and applied the developed ESNN-PC-TR on two taste recognition problems. We highlighted that:

1. There is still very little known about the perception of taste, both in terms of taste coding and processing;
2. The gustatory system is the least explored sensory system;
3. Taste recognition models are typically built either using MLP or PCA;
4. There is a lack of benchmark taste data;
5. Taste data is usually collected using in-house built sensors.

Our ESNN-PC-TR is a better model of taste recognition than any other taste recognition model published in literature because:

1. It uses integrate-and-fire neurons capable of modelling the dynamic behaviour of real neurons;
2. Data samples are encoded using a temporal encoding scheme that models the action potentials occurring in the brain;
3. It has a layer simulating taste cells-to-cell interactions;
4. It learns quickly, using a one-pass learning algorithm;
5. It learns new tastants without forgetting the ones that it has already been learnt;
6. It has knowledge discovery facility and delivers its knowledge in the form of zero-order Takagi-Sugeno rules.
7. It adapts its structure in time to accept new data samples and tastants.

We also proposed an FPGA implementation of our ESNN-PC-TR model. The ESNN-PC, including the hardware required to build the network, was hosted in an FPGA chip. We described issues related to the FPGA implementation of artificial neural networks:

1. Limited hardware resources;
2. Limited numerical range-precision;
3. Compromises in terms of speed and accuracy;
4. Lack of hardware-friendly training and learning algorithms.

Given that the processing speed is the driver for creating hardware implementations of ANN, the following design approaches were taken to obtain an optimal speed without sacrificing the network size and accuracy:

1. The system was implemented around look-up-tables, minimising the need for multipliers and logic elements;
2. To minimise the hardware usage, a virtual neural based architecture was used where only one neuron with one synapse was implemented and all other neurons used the same resources;
3. The quantization errors were kept low by representing data with 32-bit signed fixed-point words.

In the hardware implementation up to 64 neurons with up to 64 synapses each were evolved in the particular chip. Our implementation used 76% of the available FPGA memory, 45% of the available logic elements and 23% of the available dedicated multipliers. The on-chip learning circuit used 14% of the multipliers and 4% of the logic elements.

Finally, we performed some experiments to measure the speed of the proposed FPGA implementation. The experimental results have shown that the FPGA implementation was 10 to 25 times faster during the evolving of an ESNN-PC-TR network and 74 times faster during the classification than the software simulations with similar accuracy. Training required between 40 μ s and 80 μ s per sample vector. We think that this work is a big step towards building a hardware representation of the human gustatory system.

10.1.1 Answering thesis questions

In Section 1.3 of Chapter 1, nine thesis questions were proposed. Here, we summarise to what extent these questions have been answered during the course of this research.

1. *How can we enhance DSS to enable them to efficiently process dynamic real-world data and perform knowledge discovery?* In Chapter 2 Sections 2.2.1 and 2.2.4 we stated why the dynamic knowledge based connectionist systems are promising tools for building dynamic DSS. Then in Chapter 3 we concentrated on the benefits of using these systems in the field of ecological modelling. We proposed an ECOS model named LPAM in Chapter 4 and used the model on pest insect data in Chapter 5. We showed that the proposed model has superior knowledge discovery capabilities when compared with traditional neural networks. Furthermore, in Section 7.5 of Chapter 7 we discussed how the benefits of using spiking neurons for building DSS have not been thoroughly explored. This is particularly true for ecological DSS where spiking neural networks have never been used. In Chapter 8

we proposed an evolving neural network, named ESNN-PC, based on spiking neurons capable of processing temporal data and capable of knowledge discovery from temporal datasets. We showed how this model can be used for the classification of beverages (Section 8.6). We also showed that this model can be used for qualitative analysis in environmental monitoring (Section 8.7).

2. *How can we present the information learned by such a DSS to enhance the understanding of information contained in the created model?* In Section 4.3 of Chapter 4, we proposed the visualization of extracted rules using LPAM based on the information contained in the local models. The extracted knowledge is plotted in 2D diagrams where the regression coefficients of each local model are plotted against the values of the local models' centres. In Section 8.7 (Chapter 8) we proposed a framework for extracting knowledge from a trained ESNN-PC.
3. *How can we take advantage of the temporal information processing in one model?* Our literature survey of biological and computational models of taste-coding resulted in the conclusion that the current taste recognition models are not sufficient to describe complicated taste processing (Chapter 8 Section 8.1). We also found that while there is evidence that taste recognition is dynamic the existing artificial models are based on traditional static neural networks that are unable to explore the temporal data collected using taste sensors. Therefore, in Chapter 8 Section 8.3 we proposed a novel dynamic taste recognition system based on spiking neurons, the ESNN-PC-TR, and then we tested it on two real-world taste datasets.
4. *How can data collected by sensors be represented for analysis by spiking neurons in a taste recognition model? What is the advantage of doing that?* We experimented with the population encoding of sensory data and checked if population encoding can improve the classification accuracy of a taste recognition model based on spiking neurons. In Section 8.6.1 of Chapter 8, we explored the classification accuracy of an evolving spiking neural network with rank order coded inputs with and without population encoding of sensory data. We found that the accuracy of the network increased when population encoding was used. In addition, using population encoding increased the biological plausibility of our taste recognition model.
5. *Are local modelling techniques suitable for modelling complex nonlinear species-environment relationships and what are the advantages of using these models?* In

Chapter 3 Section 3.3 we introduced the problems associated with modelling invasions of exotic pest insects and in Chapter 3 Section 3.2 we reviewed the benchmark work in this modelling field. We also summarised the issues experienced with the techniques that are currently being used to build predictive ecological models. In Chapter 4 we then proposed a dynamic model, the LPAM, that addressed most of the issues set out in Chapter 3. We validated the model in Chapter 4 on two benchmark problems and in Chapter 5 and Chapter 6 we used the model for predicting the distributions of pest insects. We found that local modelling is suitable for modelling the establishment potential of pest insects and concluded that a number of local models rather than a global model can better explore an insect's invasion characteristics (Chapter 5 Section 5.4).

6. *Can local models be used to aid in the discovery of relationships between climatic factors and pest insect distributions?* We stated in Chapter 3 Section 3.2.2 that one of the main goals of modelling pest-environmental relationships is to obtain a better understanding of which climatic factors govern species invasions. We referenced published papers where the structures of trained MLP networks have been studied to obtain a better understanding of these relationships. We then proposed a dynamic technique that provides a knowledge extraction facility. In addition, in Chapter 4 Section 4.3 we provide a visualisation tool for assessing the extracted knowledge and for studying the contributions of each individual predictor variable to the prediction estimates. We also used the proposed model to prepare, for the first time, a world risk map for the establishment of the insect *P. citri*. The map can be updated if new data becomes available due to for instance the pest establishing at a new location or a climate change at the modelled locations.
7. *Is 'personalised' modelling suitable for building predictive pest distribution models?* Chapter 6 is dedicated to answering this question. 'Personalized' predictive models were built for three pest insects. Through experiments that used real-world data with 75, 78 and 191 samples we showed that 'personalized' models performed better or as good as local and global models. However, the explanatory capabilities of 'personalized' models are limited.
8. *Can a spiking neural network be used for taste recognition tasks, e.g. to classify beverages?* We review various spiking neuron architectures and coding schemes in Chapter 7 and the current knowledge about taste perception in Chapter 8 Section

8.1. In Section 8.2, we discussed the shortcomings of the currently published taste recognition models. We proposed our ESNN-PC-TR model based on spiking neurons in Section 8.3. Through software simulations of the ESNN-PC-TR we showed that it is possible to build a biologically inspired taste recognition model. We concluded that taste perception can be simulated by a network of spiking neurons with population encoded inputs.

9. *What are the benefits of a hardware implementation of a connectionist system?* We also built an FPGA implementation of the ESNN-PC-TR model and explained the implementation in Chapter 9. We discussed the issues associated with implementing an ANN in FPGA in Chapter 9 Section 9.1 and gave the details of our implementation in Section 9.2. We tested the implementation on the same set of real-world taste data we used to evaluate the ESNN-PC-TR in Chapter 8. Our results showed that the FPGA implementation was 10 to 25 times faster than the software simulations during the evolving phase and 74 times faster during the classification of the beverages. The other benefits of using a hardware implementation are the lower implementation costs (Damak, 2006), its scalability and suitability for embedded design (Lee & Ko, 2006).

10.2 Analysis of the proposed models

Two models inspired by computational intelligence methods are proposed in this thesis; a dynamic model for risk evaluation (LPAM) and a dynamic model of taste recognition (ESNN-PC-TR). Here, we review their strengths and weaknesses.

10.2.1 LPAM model strengths and weaknesses

The strengths of the LPAM model lie in its ability to explain what it has learnt about the modelling problem. The LPAM is based on DENFIS and accordingly it has knowledge extraction capabilities. This is a very important feature when this model is applied to ecological problems because it opens up the possibility of improving our understanding of ecological processes. The probability evaluation module included in the LPAM makes the model less sensitive to the noise found in the data. This should be useful whenever there is an unknown amount of noise in the modelling data. Furthermore, the LPAM is trained using incremental learning and therefore can learn new information as it becomes available without forgetting what it has already learnt. As a result, the model can be built for one set of conditions and it will adjust when those conditions change.

This makes the LPAM suitable for modelling the influence of climate change on ecological processes.

We think that local modelling is a logical approach for modelling the habitat of invasive insects over smaller geographical areas because the information about insect locations is better conserved in the estimation. However, the number of local models must be optimised. In our work, this number was hand-tuned and therefore, there is a further opportunity to automate this process.

10.2.2 ESNN-PC model strength and weaknesses

The main strength of the ESNN-PC model is its biologically inspired processing and structure. The model is based on spiking neurons capable of modelling neural dynamics and proven as efficient units for modelling the visual (Thorpe, Delorme, and Van Rullen, 2001) and audio systems (Wysoski, Benuskova, and Kasabov, 2007b). In this thesis we have shown that these neurons are also suitable for modelling taste recognition. A possible integration of the taste recognition model leading to the creation of a model of multimodal perception of food and beverages is suggested in the future work section. The dynamic nature of the model allows it to learn new tastants without forgetting the ones that it has already learnt, a feature that is natural to humans. The model also delivers its knowledge in the form of IF-THEN rules. One weakness in our design is that the model's parameters must be optimised for accurate taste recognition. While, our optimization which has been done by hand has provided an accurate classification, an automated approach would be more desirable.

Hosting the model in hardware not only improved its processing speed but also created new opportunities for the use of this model. Some possible applications are suggested in the following section. At the moment, the data samples are fed to the ESNN-PC-TR from the host computer via the NIOS embedded processor. This slows down the processing considerably. Another weakness in the current design is due to the fact that the network evolves as it learns new samples. Evolving results in a new formation of neurons and connections. The number of neurons depends on the number of classes used to train the ESNN-PC-TR and on the variation among the samples from those classes. While neurons are typically aggregated during the evolving stage, if we allow the ESNN-PC-TR to operate in a life-long manner it can become too big for the size of the FPGA chip used. In the future we can rethink this design and include some safety

mechanisms to stop the network evolving when the number of neurons becomes too big for the chip.

10.3 Suggestions for future work

In this section, we suggest four possible directions where the taste recognition model ESNN-PC-TR and the local probability adaptive model LPAM may be used and further extended. The directions can be explored separately or in sequence, where for instance, the second direction would benefit from the first and the third direction could be explored after the first two are completed.

10.3.1 The ESNN based multimodal taste recognition system

In this thesis, a taste recognition model based on spiking neurons with rank order coded inputs has been proposed and tested on two real-world datasets. The experimental results, using both a software simulation and hardware implementation, showed that the model was able to differentiate between samples of wine and water. The same type of neuron has been proven in modelling the visual (Thorpe, Delorme, and Van Rullen, 2001) and audio system (Wysoski, Benuskova, and Kasabov, 2007b). The same neurons were also used in an integration of the visual and auditory modalities (Wysoski, Benuskova, and Kasabov, 2007a). Clearly, the perception of taste is multimodal (Verhagen and Engelen, 2006; Yoshimura *et al.*, 2004; Dulac, 2000; Toko, 2000). Not surprisingly, a combination of smell and taste data samples give a better result than when only one type of data is used (Wide *et al.*, 1998).

Therefore, one direction in which this thesis can be extended is to build an olfactory model (ESNN-PC-O). This model can be based on the same spiking neurons and coding scheme as used in our taste recognition model and it can be integrate with the current taste model (ESNN-PC-TR) to provide a multimodal model of taste perception (Fig. 10.1). One could explore different techniques of fusing the information coming out of the separate modules to achieve the best discrimination properties. Acknowledging that the auditory system plays a role in taste perception (Verhagen and Engelen, 2006), a taste-smell-audio model of food perception could also be built. Adding visual information could also further increase the biological realism of the taste recognition system. Ultimately, the resulting model could be implemented in hardware on one chip or on a number of separate chips.

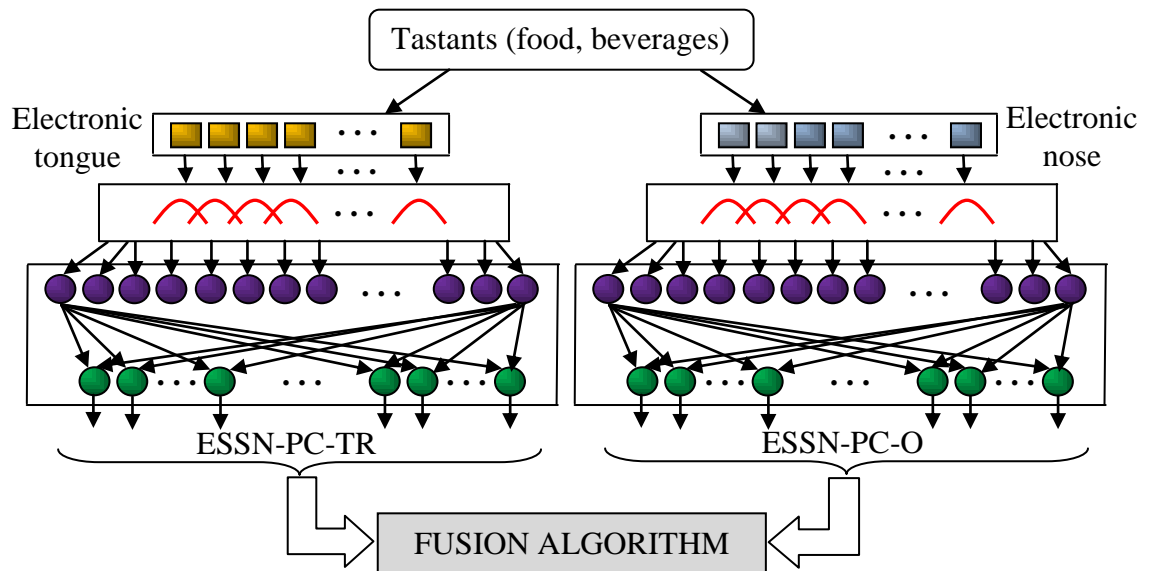


Fig. 10.1. The multimodal taste recognition model based on the ESNN-PC approach.

10.3.2 Smarter mobile robots

Spiking neural networks have been used to control robotic behaviour (Alnajjar & Murase, 2006; Kubota & Sasaki, 2004; Nielsen & Lund, 2003) where the robots learnt to navigate through changing environments. Providing a robot with the ability to taste and smell would make the robot an ideal environmental monitoring tool. To achieve this goal, two main requirements must be satisfied. Firstly, we need to have quality taste and smell sensors that are able to operate continuously outside experimental environments. Secondly, we need a robust hardware implementation of the taste recognition system. In our current implementation of taste recognition, data samples were collected in advance and fed to the system via a personal computer and an on-chip microprocessor. To avoid this interface between the tastants and the taste recognition system, the sensors would need to be mounted on the robot and measurements fed to the taste module. It is very likely that the signals will need to be pre-processed in a fast interface module. The robot could be trained on some tastants, but would also be able to acquire new tastes during its operational phase. A group of testing and smelling robots could communicate wirelessly, forming an autonomous sensor net that could potentially provide continuous environmental monitoring for a large area. Some other envisaged applications are in biosecurity, the chemical industry, the food industry, security, the automotive industry, and in home automation. This leads us to the third avenue of the proposed future work.

10.3.3 Pervasive computing

One direction in which this work can be extended is in its utilization in distributed sensory systems. Particularly, the FPGA implementation of the ESNN-PC opens up the opportunity to embed this system into various pervasive devices. For example, ESNN-PC systems could be used to taste (ESNN-PC-TR) or smell (ESNN-PC-O) if beverages and foods have passed their ‘best before’ dates (Fig. 10.2). Together with an appropriate set of sensors capable of ‘smelling’ different species, the ESNN-PC could be used to detect the presence of unwanted species. ESNN-PC embedded in smart shipping containers (Fig. 10.3) could detect unwanted pest species before the containers arrived to their destination. Thus would allow risk management in transit or on departure. Our design of having the whole system on one chip satisfies the requirement that the pervasive device must have a low price and must be easy to fit into a wide range of deployed infrastructures (Hansmann *et al.*, 2003). While currently the main obstacle to the creation of such system is the lack of off-the-shelf sensors, as technology evolves, this obstacle will soon disappear.

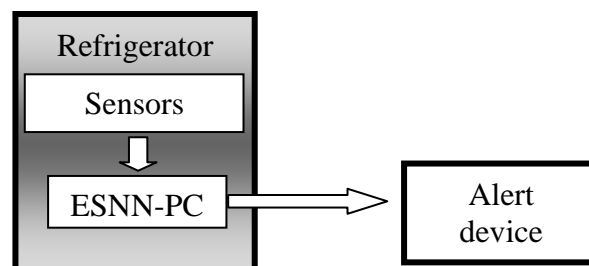


Fig. 10.2. An ESNN-PC embedded in a smart refrigerator.

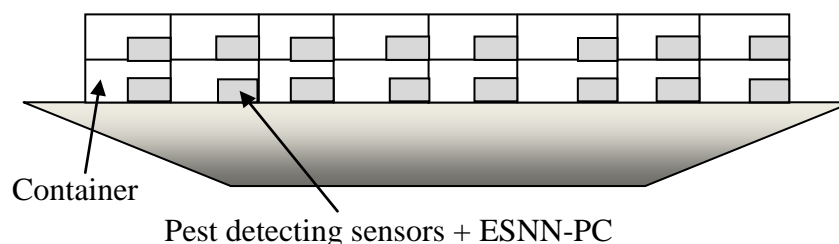


Fig. 10.3. A ship with container cargo using embedded ESNN-PC modules for in-situ detection of unwanted species posing bio-security risks.

We can also envisage the scenario shown in Fig. 10.4 where different sensors (temperature, moisture, pollutant, ...) continuously measure the climatic characteristics of an area. These arrays of sensors are paired up with LPAM modules where the LPAM

provide data analysis at a local level thus providing local modelling before sending the information to a higher-level global monitoring system. The LPAM's ability to adapt to the environmental dynamics makes the LPAM suitable for life-long monitoring operations.

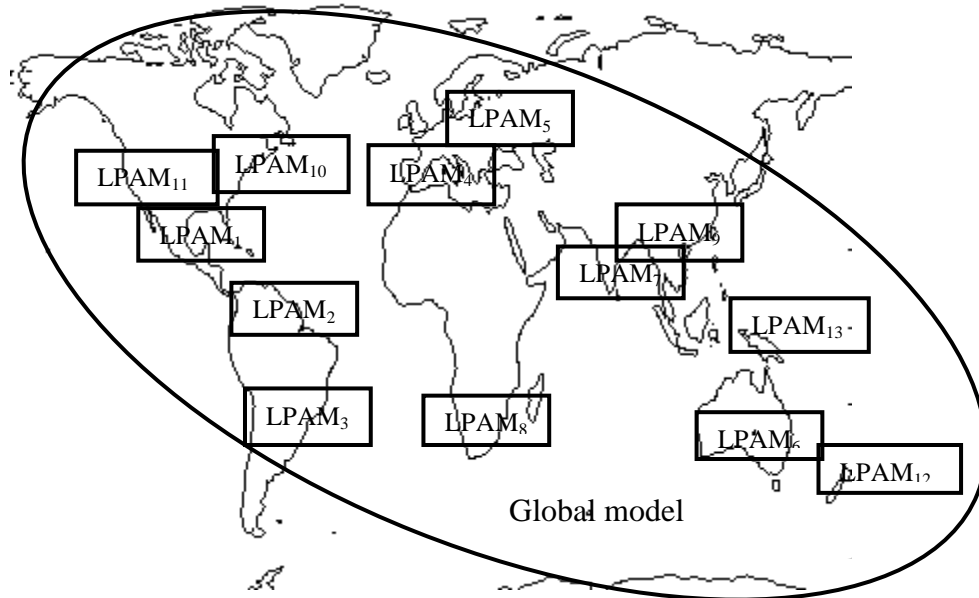


Fig. 10.4. LPAM for local modelling in a distributed environmental monitoring system.

10.3.4 Quantum evolving spiking neural networks - QSNN

In our work, despite the fact that the models are able to evolve their structures, a number of the models' parameters must be optimised to achieve accurate modelling. As stated in Section 10.2, while the manual optimization has provided accurate models, an automated approach would be more desirable. Studies by Kasabov (Kasabov, 2007) showed that optimising the neural network parameters and selecting the relevant data features in one optimisation improved modelling accuracy. However, classical search algorithms are slow and inappropriate for building adaptive SNN. In (Defoin-Platel, Schliebs & Kasabov, 2007) an extremely fast quantum inspired evolutionary algorithm for feature selection is proposed. In (Venayagamoorthy & Gaurav, 2005) quantum-inspired evolutionary algorithms are used to train MLP networks.

Encouraged by promises of faster and more accurate neural networks (Narayannan & Maneer, 2000), we propose a future extension to the ESNN-PC model. In this extension the currently used spiking neurons are replaced by novel quantum spiking neurons where the spikes are represented as quantum entities characterised by a probability $p_i(t)$ which is dependent on, for instance, the time integral between two spikes. In the new

QESNN-PC model both the optimization of the features and the network parameters are done at the same time as the network is built using a novel quantum inspired search algorithm. The QESSN-PC would replace the ESNN-PC in the applications suggested in Sections 10.3.1 – 10.3.3.

10.4 Overall conclusion

The main goal in this thesis is to develop new generic ECOS for decision support that have the following features: (1) adaptive, evolving learning; (2) knowledge extraction and knowledge discovery; and (3) accurate predictions. This thesis presented LPAM, a new adaptive predictive model based on local probability applied for assessing the establishment potential of pest insects, and ESNN-PC-TR, a new adaptive taste recognition model based on spiking neurons. Both approaches were evaluated on real-world datasets and their performances were successful. The LPAM was as accurate as other published models but provided a better understanding of modelling problems. The ESNN-PC-TR is more biologically plausible than any taste recognition model published up to date. The ESNN-PC was implemented in software and hardware and the two implementations were compared in terms of processing speed and accuracy. Both models, particularly the ESNN-PC and its FPGA implementation, provide very exciting avenues for future work.

The software implementation of the ESNN-PC (ESNN-PC-TR) is available free from http://manukau.ac.nz/departments/e_e/staff/soltic.asp.

References

- Abbott, L. F. & Nelson, S. B. (2000). Synaptic plasticity: timing the best. *Nature neuroscience supplement*, 3, 1178-1183.
- Abeles, M., Bergman, H., Margalit, E. & Vaadia, E. (1993). Spatiotemporal firing patterns in the frontal cortex of behaving monkeys. *Journal of Neurophysiology*, 70 (4), 1629-1638.
- Al-Kazzaz, S. A. & Khalil, R. A. (2008). FPGA Implementation of artificial neurons: comparison study. 3rd International Conference on Information and Communication Technologies: From Theory to Applications, 7-11 Apr, 1-6.
- Allen, D., Halliday, D. M. & Tyrrell, A. M. (2006). A hybrid bio-inspired system: Hardware spiking neural network incorporating Hebbian learning with microprocessor based Evolutionary control algorithm. *IEEE Congress on Evolutionary Computation*, Vancouver, BC, Canada, 16-21 Jul, 2958-2965.
- Allen, J. N., Abdel-Aty-Zohdy, H. S. & Ewing, R. L. (2005). Plasticity recurrent spiking neural networks for olfactory pattern recognition. 48th Midwest Symposium on Circuits and Systems MWSCAS2005, 7-10 Aug, 2, 1741-1744.
- Allen, J. N., Abdel-Aty-Zohdy, H. S. & Ewing, R. L. (2004). Electric nose inhibition in a spiking neural network for noise cancellation. *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology CIBCB '04*, 7-8 Oct, 129- 133.
- Allen, J. N., Abdel-Aty-Zohdy, H. S., Ewing, R. L. & Chang, T. S. (2002). Spiking networks for biochemical detection. 45th Midwest Symposium on Circuits and Systems MWSCAS-2002, 4-7 Aug, 3, III-129–III-132.
- Alexouda, A. (2005). A user-friendly marketing decision-support system for the product line design using evolutionary algorithms, *Decision support systems*, 38, 495-509.
- Alnajjar, F. & Murase, K. (2006). Use-dependent synaptic connection modification in SNN generating autonomous behavior in a Khepera mobile robot. *IEEE Conference on Robotics*, Dec, 1-6.
- Altera, <http://www.altera.com/>
<http://university.altera.com/materials/boards/unv-de2-board.html>
<ftp://ftp.altera.com/up/pub/Tutorials/DE2/ComputerOrganization/tutDE2sdramvhd1.pdf> . Last assessed on 15/09/2008

- Ambard, M. & Martinez, D. (2006). Inhibitory control of spike timing precision. *Neurocomputing*, 70, 200-205.
- Amin, H. H. & Fujii, R. H. (2004). Input arrival-time-dependent decoding scheme for a spiking neural network. European Symposium on Artificial Neural Networks ESANN'2004, Bruges, Belgium, 28-30 Apr, 355-360.
- Anderson, K. M., Odell, P. M., Wilson, P. W. F. & Kannel, W. B. (1991). Cardiovascular disease risk profiles. *American Heart Journal*, 121, (1 Pt 2), 293-298.
- Anderson, R. P., Lew, D. & Peterson, A. T. (2003). Evaluating predictive models of species' distributions: criteria for selecting optimal models. *Ecological Modelling*, 162, 211-232.
- Andò, B., Baglio, S., Conduro, D., Graziani, S. & Pitrone, N. (1998). On air quality monitoring in industrial areas. IEE Instrumentation and Measurements Technology Conference, St. Paul Minnesota, USA, 18-21 May, 325-329.
- Araújo, M. B. & Williams, P. H. (2000). Selecting areas for species persistence using occurrence data. *Biological Conservation*, 96, 331-245.
- Asuncion, A. & Newman, D.J. (2007). UCI Machine Learning Repository <http://www.ics.uci.edu/~mlern/MLRepository.html>. Last accessed on 27.12.2008.
- Aurelle, D., Lek, S., Giraudel, J.-L., Berrebi, P. (1999). Microsatellites and artificial neural networks: tool for the discrimination between natural and hatchery brown trout (*Salmo trutta*, L.) in Atlantic populations. *Ecological Modelling*, 120, 313-324.
- Austin, M. (2007). Species distribution models and ecological theory: a critical assessment and some possible new approaches. *Ecological Modelling*, 200, 1-19.
- Baddeley, R. (1996). An efficient code in V1?. *Nature*, 381 (6583), 560-561.
- Bade, S. L. & Hutchings, B. L. (1994). FPGA-Based Stochastic Neural Networks – Implementation. IEEE Workshop on FPGAs for Custom Computing Machines, 10-13 Apr, 189-198.
- Baker, R. H. A (2002). Predicting the Limits to the Potential Distribution of Alien Crop Pests. In G. Halman and C. P. Schwalbe (Eds.), *Invasive arthropods and agriculture: problems and solutions* (pp. 207-241). Enfield, New Hampshire: Science Publishers Inc.

- Barendregt, A. & Bio, A. M. F. (2003). Relevant variables to predict macrophyte communities in running waters. *Ecological Modelling*, 160, 205-217.
- Barker, G. M, Stephens, A., Hunter, C., Rutledgw, D., Harris, R. J., Lariviere, M. C. and Gough, J. D. (2002). Biosecure – a model for analysis of biosecurity risk profiles, in S. L. Goldson, S.L. and D. M. Suckling, New Zealand Plant Protection Society Inc., 73-91.
- Bayoh, M. N., Thomas, C. J. & Lindsay, S. W. (2001). Mapping distributions of chromosomal forms of *Anopheles gambiae* in West Africa using climatic data. *Medical and Veterinary Entomology*, 15, 267-274.
- Belatreche, A., Maguire, L. P. & McGinnity, M. (2007). Advances in design and application of spiking neural networks. *Soft Computing*, 11, 239-248.
- Binns, N. A. & Eiserman, F. M. (1979). Quantification of fluvial trout habitat in Wyoming. *Transactions of the American Fisheries Society*, 108 (3), 215-228.
- Bishop, C. M. (1995). *Neural networks for pattern recognition*, Oxford: Clarendon Press.
- Blanning, R. W. & Reinig, B. A. (2005). A framework for conducting political event analysis using group support systems. *Decision Support Systems*, 38, 511-527.
- Blum, A. & Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. 8th International Conference on Machine Learning ICML 2001, 19-26.
- Bonnet, C., Zamora, M. C., Buratti, F. & Guirao, M. (1999). Group and Individual gustatory reaction times and Piéron's law. *Physiology and Behavior*, 66 (4), 549-558.
- Booij, O. & tat Nguyen, H. (2005). A gradient descent rule for spiking neurons emitting multiple spikes. *Information Processing Letters*, 95, 552-558.
- Bothe, S. M. (2004). The evidence for neural information processing with precise spike-times: a survey. *Neural Computing*, 3 (2), 1-13.
- Bothe, S. M. (2003). *Spiking neural networks*. PhD Thesis University of Leiden.
- Bothe, S. M., Kok, J. N. & La Poutré, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48, 17-37.
- Bothe, S. M., La Poutré, H. & Kok, J. N. (2002). Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer RBF networks. *IEEE Transactions on Neural Networks*, 13 (2), 426-435.

- Bose, I., Eryarsoy, E. & He, L. (2005). Multi-period design of survivable wireless access networks under capacity constraints. *Decision Support Systems*, 38, 529-538.
- Bosnić, Z., Kononenko, I., Robnik-Šikonja M. & Kukar, M. (2003). Evaluation of prediction reliability in regression using the transduction principle. *IEEE Region 8 Computer as a Tool EUROCON 2003*, Ljubljana, Slovenia, 22-24 Sep 2, 99–103.
- Bowers, J. A. & Shedrow, C. B. (2000), Predicting stream water quality using artificial neural networks (ANN). 8th International Conference on Development and Application of Computer Techniques to Environmental Studies ENVIROSOFT 2000, 89-97.
- Bradbury, J. (2004). Taste perception: Cracking the code. *PLoS Biol.* [On line]. 2 (3). Available: <http://biology.plosjournals.org>. Last accessed on 20/09/2008.
- Bradshaw, C. J. A., Davis, L. S., Purvis, M., Zhou, Q. & Benwell, G. L. (2002). Using artificial neural networks to model the suitability of coastline for breeding by New Zealand fur seal (*Arctocephalus forsteri*). *Ecological Modelling*, 148, 111-131.
- Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J. M., & *et al.* (2007). Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of Computational Neuroscience*, 23, 349-398.
- Brody, C. D. & Hopfield, J. J. (2003). Simple networks for spike-timing-based computation, with application to olfactory processing. *Neuron*, 37, 843-852.
- Brosse, S., Guegan, J.-F., Tourenq, J.-N., Lek, S. (1999). The use of artificial neural networks to assess fish abundance and spatial occupancy in the littoral zone of a mesothropic lake. *Ecological modelling*, 120, 299-311.
- Brunelli, L., Melcher, E. U. K., de Brito, A. V. & Freire, R. C. S. (2005). A novel approach to reduce interconnect complexity in ANN hardware implementation. *IEEE International Joint Conference on Neural Networks*, 31Jul-4Aug, 5, 2861-2866.
- Buračas, G. T., Zador, A. M., De Weese, M. R. & Albright, T. D. (1998). Efficient discrimination of temporal patterns by motion-sensitive neurons in primate visual cortex. *Neuron*, 20, 959-969.
- Burgsteiner, H. (2006). Imitation learning with spiking neural networks and real-world devices. *Engineering Applications of Artificial Intelligence*, 1, 741-752.

- CAB International. (2004). CABI crop compendium. Wallingford, United Kingdom. CAB International.
- Cariani, P. A. (2004). Temporal codes and computations for sensory representation and scene analysis. *IEEE Transactions on Neural Networks*, 15 (5), 1100-1111.
- Castillo, P. A., Merelo, J. J., Prieto, A., Rivas, V. & Romero, G. (2000). G-Prop: Global optimization of multilayer perceptrons using GAs. *Neurocomputing*, 35, 149-163.
- Cawley, G. C., Janacek, G. J., Haylock, M. R. & Dorling, S. R. (2007). Predictive uncertainty in environmental modelling. *Neural Networks*, 20, 537-549.
- Céréghino, R., Giraudel, J. L. & Compin, A. (2001). Spatial analysis of stream invertebrates distribution in the Adour-Garonne drainage basin (France), using Kohonen self organizing maps. *Ecological Modelling*, 146, 167-180.
- Chalhoub, N., Muller, F. & Auguin, M. (2006). FPGA-based generic neural network architecture. *International Symposium on Industrial Embedded Systems IES'06*, 18-20 Oct, 1-4.
- Chandrashekar, J., Hoon, M. A., Ryba, N. J. & Zuker, C. S. (2006). The receptors and cells for mammalian taste. *Nature*, 444, 288-294.
- Chefaoui, R. M. & Lobo, J. M. (2008). Assessing the effects of pseudo-absences on predictive distribution model performance. *Ecological Modelling*, 210, 478-486.
- Chen, Q. & Mynett, A. E. (2003). Integration of data mining techniques and heuristic knowledge in fuzzy logic modelling of eutrophication in Taihu Lake. *Ecological Modelling*, 162, 55-67.
- Chen, Y., Wang, G. & Dong, S. (2003). Learning with progressive transductive support vector machines. *Pattern Recognition Letters*, 24, 1845-1855.
- Cherkassky, V. (1998). Fuzzy Inference Systems: a critical Review. In O. Kaynak (Ed.). *Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications* (pp. 177-197), Berlin, New York: Springer.
- CLIMEX, <http://www.climatemodel.com/climex.htm>. Last accessed on 28/09/2008.
- Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, XX (1), 37-46.
- Cohen, S. D. (1998). Evaluating the Risk of Importation of Exotic Pests Using Geospatial Analysis and Pest Risk Assessment Model. 1st International Conference on Geospatial Information in Agriculture and Forestry, Lake Buena

Vista, Florida, USA. 1-3 Jun. Available:

<http://www.aphis.usda.gov/ppd/evaluating.pdf>. Last accessed on 25.09.2008.

- Cook, W. C. (1931). Notes on predicting the probable future distribution of introduced insects. *Ecology*, 12 (2), 245-247.
- Cortina, M., Duran, A., Alegret, S. & del Valle, M. (2006). A sequential injection electronic tongue employing the transient response from potentiometric sensors for anion multidetermination. *Analytical and Bioanalytical Chemistry*, 358 (7), 1186-1194.
- Cowley, M. J. R., Wilson, R. J., León-Cortés, J. L., Gutiérrez, D., Bulman, C. R. & Thomas, C. D. (2000). Habitat-based statistical models for predicting the spatial distribution of butterflies and day-flying moths in a fragmented landscape. *Journal of Applied Ecology*, 37 (Suppl. 1), 60-72.
- Cox, C. E. & Blanz, W. E. (1992). GANGLION – A fast Field-Programmable Gate Array implementation of a connectionist classifier. *IEEE Journal of Solid-State Circuits*, 27 (3), 288-299.
- Damak, A., Krid, M., Sellami Masmoudi, D. & Derbel, N. (2006). FPGA implementation of programmable pulse mode neural network with on-chip learning. *International Conference on Design and Test of Integrated Systems in Nanoscale Technology DTIS2006*, 5-7 Sep, 159-164.
- de Garis, H. & Korkin, M. (2002). The CAM-brain machine (CBM): an FPGA-based hardware tool that evolves a 1000 neuron-net circuit module in seconds and updates a 75 million neuron artificial brain for real-time robot control. *Neurocomputing*, 42, 35-68.
- Defoin-Platel, M., Schliebs, S. & Kasabov, K. (2007). A versatile quantum inspired evolutionary algorithm. *IEEE Congress on evolutionary computation CEC 2007*, 25-28 Sept, 423 – 430.
- Delorme, A., Perrinet, L. & Thorpe, S. J. (2001), Networks of integrate-and-fire neurons using Rank Order Coding B: Spike timing dependent plasticity and emergence of orientation selectivity. *Neurocomputing*, 38-40, 539-545.
- Delorme, A., Gautrais, J., Van Rullen, R. & Thorpe, S. (1999). SpikeNET: A simulator for modelling large networks of integrate and fire neurons. *Neurocomputing*, 26-27, 989-996.

- Deng, D. & Kasabov, N. (2000). ESOM: An algorithm to evolve self-organising maps from on-line data streams. IEEE-INNS-ENNS International Joint Conference on Neural Networks IJCNN 2000, 24-27 Jul, 3-8.
- Dentener, P. R., Whiting, D. C. & Connolly, P. G. (2002). Thrips *palmi* Karny (*Thysanoptera: Thripidae*): could it survive in New Zealand?. New Zealand Plant Protection, 55, 18-24.
- Derbeko, P., El-Yaniv, R. & Meir, R. (2003). Error bounds for transductive learning via compression and clustering, In P. Husbands & I. Harvey (Eds.) Advances in Neural Information Processing Systems NIPS, 16, 1085-1092.
- de Sousa, H. C., Carvalho, A. C. P. L. F., Riul Jr., A. & Mattoso, L. H. C. (2002). Using MLP networks to classify red wines and water readings of an artificial tongue. The VII Brazilian Symposium on Neural Networks SBRN'02, 13-18.
- Di Lorenzo, P. M. & Victor, J. D. (2003). Taste response variability and temporal coding in the nucleus of the solitary tract of the rat. Journal of Neurophysiology, 90, 1418-1431.
- Dobesberger, E. J. (2002). Multivariate techniques for estimating the risk of plant pest establishment in new environments. North American Plant Protection Organization International Symposium on Pest Risk Analysis, Puerto Vallarta, Mexico, 18-21 Mar.
- Dobesberger, E. (2000). Climate based modelling of pest establishment and survival in support of rest risk assessment. Annual report 1999-2000, North American Plant Protection Organization, 35-36. Available:
<http://www.nappo.org/Reports/AnnRep-99-00-e.pdf>
<http://www.nappo.org/PRA-Symposium/PDF-Final/Dobesberger.pdf>. Last accessed on 19/09/2008.
- Drumm, D., Purvis, M. & Zhou, Q. (1999). Spatial ecology and artificial neural networks: modeling the habitat preference of the sea cucumber (*Holothuria leucospilota*) on Rorotonga Cook Island. 11th Annual Colloquium of the Spatial Information Research Centre University of Otago SIRC 99, Dunedin, New Zealand, 13-15 Dec.
- Available: <http://waitaki.otago.ac.nz/~martin/Documents/SIRC-99Drumm.pdf>. Last accessed on 20/09/2008.
- Dulac, C. (2000). The physiology of taste, vintage 2000. Cell, 100, 607-610.

- Eldredge, J. G & Hutchings, B. L. (1994). RRANN: A hardware implementation of the backpropagation algorithms using reconfigurable FPGAs. IEEE International Conference on Neural Networks, 27 Jun-2 Jul, 4, 2097–2102.
- El-Yaniv, R. & Gerzon, L. (2005). Effective transductive learning via objective model selection. Pattern Recognition Letters, 26, 2104-2115.
- Eurich, C. W. & Wilke, S. D. (2000). Multidimensional encoding strategy of spiking neurons. Neural Computation, 12, 1519-1529.
- Eyre, M. D., Rushton, S. P., Luff, M. L. & Telfer, M. G. (2005). Investigating the relationships between the distribution of British ground beetle species (*Coleoptera, Carabidae*) and temperature, precipitation and altitude. Journal of Biogeography, 32 (6), 973-983.
- Farber, O. & Kadmon, R. (2003). Assessment of alternative approaches for bioclimatic modeling with special emphasis on the Mahalanobis distance. Ecological Modelling, 160, 115-130.
- Fielding, A. H. (1999). An introduction to machine learning methods. In A. H. Fielding (Ed.) Machine Learning Methods for Ecological Applications. Kluwer academic publishers group, The Netherlands, 1-35.
- Fielding, A. H. & Bell, J. F. (1997). A review of methods for the assessment of prediction errors in conservation presence/absence models. Environmental Conservation, 24 (1), 38-49.
- Furukawa, S., Xu, L. & Middlebrooks, J. C. (2000). Coding of sound-source location by ensembles of cortical neurons. Journal of Neuroscience, 20 (3), 1216-1228.
- Gallardo, J., Alegret, S., Muñoz, R., De-Román, M., Leija, L., Hernández, P. R. & del Valle, M. (2003). An electronic tongue using potentiometric all-solid-state PVC-membrane sensors for the simultaneous quantification of ammonium and potassium ions in water. Analytical and Bioanalytical Chemistry, 377 (2), 248-256.
- García, P., Lastra, J., Marquínez, J. & Nores, C. (2007). Detailed model of shelter areas for the Cantabrian brown bear. Ecological informatics, 2, 297-307.
- Gardner, J. W., Shin, H. W., Hines, E. L. & Dow, C. S. (2000). An electronic nose system for monitoring the quality of potable water. Sensors and Actuators B, 69, 336-341.

- Gautrais, J. & Thorpe, S. (1998). Rate coding versus temporal order coding: a theoretical approach. *BioSystems*, 48, 57-65.
- Gavashelishvili, A. & Lekarevskiy, V. (2008). Modelling the habitat requirements of leopard *Panthera pardus* in west and central Asia. *Journal of Applied Ecology*, 1-10.
- GENESIS <http://www.genesis-sim.org>
- Gerstner, W. & Kistler, W. M. (2002). *Spiking Neural Models, Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.
- Gerstner, W (1999). Computing with spiking neurons. In W. Maass and C. H. Bishop (Eds.) *Pulsed Neural Networks*, WA Bratford Book.
- Gerstner, W., Kempter, R., van Hemmen, J. L. & Wagner, H. (1996). A neural learning rule for sub-millisecond temporal coding. *Nature*, 383, 76-78.
- Gevrey, M., Worner, S., Kasabov, N., Pitt, J. & Giraudel, J.-L. (2006). Estimating risk of events using SOM models: a case study on invasive species establishment. *Ecological Modelling*, 197, 361-372.
- Gevrey, M., Dimopoulos, I. & Lek, S. (2003). Review and comparison of methods to study the contribution of variables in artificial neural network models. *Ecological Modelling*, 160, 249-264.
- Gibson, L. A., Wilson, B. A., Cahill, D. M. & Hill, J. (2004). Spatial prediction of rufous bristlebird habitat in a coastal heathland: a GIS-based approach. *Journal of Applied Ecology*, 41, 213-223.
- Gilbertson, T. A. (1998). Gustatory mechanisms for the detection of fat. *Current Opinion in Neurobiology*, 8 (4), 447-452.
- Glackin, B., McGinnity, T. M., Maguire, L. P., Wu, Q. X. & Belatreche, A. (2005) A novel approach for implementation of large scale spiking neural networks on FPGA hardware. *IWANN, Lecture Notes in Computer Science*, Springer-Verlag, New York. LNCS 3512, 552-563.
- Glendinning, J. I, Davis, A. & Rai, M. (2006). Temporal coding mediates discrimination of “bitter” taste stimuli by an insect. *The Journal of Neuroscience*, 26 (35), 8900-8908.
- Gomes, C. F. S., Nunes, K. R. A., Xavier, L. H., Cardoso, R. & Valle, R. (2008). Multicriteria decision making applied to waste recycling in Brazil. *Omega*, 36, 395-404.

- Guisan, A. & Zimmermann, N. E. (2000). Predictive habitat distribution models of ecology. *Ecological modelling*, 135, 147-186.
- Gutés, A., Calvo, D., Céspedes, F. & del Valle, M. (2007). Automatic sequential injection analysis electronic tongue with integrated reference electrode for the determination of ascorbic acid. *Microchimica Acta*, 157, 1-6.
- Gutés, A., Ibáñez, A. B., Céspedes, F., Alegret, S. & del Valle, M. (2005). Simultaneous determination of phenolic compounds by means of an automated voltammetric “electronic tongue. *Analytical and Bioanalytical Chemistry*, 382, 471-476.
- Gutiérrez, M., Gutiérrez, J. M., Alegret, S., Leija, L., Hernández, P. R., Favari, L. et al. (2008). Remote environmental monitoring employing a potentiometric electronic tongue. *International Journal of Environmental Analytical Chemistry*, 88, 103-117.
- Han, S. H., Kim, D. Y., Kim, H. & Jang, W.-S. (2008). A web-based integrated system for international project risk management. *Automation in Construction*, 17, 342-356.
- Harva, M. (2007). A variation EM approach to predictive uncertainty. *Neural Networks*, 20, pp. 550-558.
- Hansmann, U., Merk, L., Nicklous, M. S. & Stober, T. (2003). *Pervasive computing*, Springer, 2nd edition.
- Hauptmann, P., Borngraeber, R., Schroeder, J. & Auge, J. (2000). Application of novel sensor electronics for quartz resonators in artificial tongue. *IEEE/EIA International Frequency control symposium and exhibition*, 100-105.
- Hellmich, H. H. & Klar, H. (2004). An FPGA based simulation acceleration platform for spiking neural networks. *47th IEEE International Midwest Symposium on Circuits and Systems, II* 389-II 392.
- Hikawa, H. (2003). A new digital pulse-mode neuron with adjustable activation function. *IEEE Transaction on Neural Networks*, 14 (1), 236-242.
- Hikawa, H. (1999). Frequency-based multilayer neural network with on-chip learning and enhanced neuron characteristics. *IEEE Transaction on Neural Networks*, 10 (3), 545-553.
- Hikawa, H. (1997). Improvement on the learning performance of multiplierless multilayer neural network. *IEEE International Symposium on Circuits and Systems*, 9-12 Jun, Hong Kong, 641-644.

- Hikawa, H. (1995). Implementation of simplified multilayer neural networks with on-chip learning. *IEEE International Conference on Neural Networks*, Nov/Dec, 4, 1633-1637.
- Himavathi, S., Anitha, D. & Muthuramalingam, A. (2007). Feedforward neural network implementation in FPGA using layer multiplexing for effective resource utilization. *IEEE Transactions on Neural Networks*, 18 (3), 880-888.
- Hirzel, A. & Guisan, A. (2002). Which is the optimal sampling strategy for habitat suitability modelling. *Ecological modelling*, 157, 331-341.
- Ho, S.-S. & Wechsler, H. (2003). Transductive confidence machine for active learning. *International Joint Conference on Neural Networks*, 20-24 Jul, 2, 1435-1440.
- Holt, J. L. & Baker, T. E. (1991). Back propagation simulations using limited precision calculations. *International Joint Conference on Neural Networks*, 8-14 Jul, II121-II126.
- Hopfield, J. J. (1995). Pattern recognition computation using action potential timing for stimulus representation. *Nature*, 376 (6536), 33-36.
- Huang, H., Pasquier, M. & Quek, C. (2008). Financial market trading system with a hierarchical coevolutionary fuzzy predictive model. *IEEE Transactions on Evolutionary Computation*, PP (99) Accepted for future publication.
- Huang, L., Song, Q. & Kasabov, N. (2005). Evolving connectionist systems based role allocation of robots for soccer playing. *IEEE International Symposium on Intelligent Control*, Cyprus, 27-29 Jun, 36-40.
- Huang, Y. J., Maruyama, Y., Lu, K. S., Pereira, E. & Roper, S. D. (2005). Mouse taste buds release serotonin in response to taste stimuli. *Chemical Senses*, 30 (Supplement 1), i39-i40.
- Iannella, N. & Kindermann, L. (2005). Finding iterative roots with a spiking neural network. *Information Processing Letters*, 95, 545-551.
- IEEE Computational Intelligence Society, Intelligent Systems Applications Technical Committee (ISATC), Taskforce "Earth and Environmental Sciences" TF EES
Available: http://www.unesco-ihe.org/hi/sol/TF_CIEES/. Last accessed on 16.09.2008.
- INNS Special Interest Group (SIG) on Computational Intelligence in Earth and Environmental sciences, Available:
http://www.unesco-ihe.org/hi/sol/SIG_CIEES/. Last assessed on 13.09.2008.

- Ivarsson, P., Kikkawaka, Y., Winquist, F., Krantz-Rülcker, C., Höjer, N.-E., Hayashi, K. *et al.* (2001). Comparison of a voltmetric electronic tongue and a lipid membrane taste sensors. *Analytica Chimica Acta*, 449, 59-68.
- Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14 (6), 1569-1572.
- James-Roxby, P., Blodget, B. J. (2000). A study of high-performance reconfigurable constant coefficient multiplier implementations. J. Schewel *et al.* (Eds.), *Reconfigurable Technology: FPGAs for Computing and Applications II*, SPIE 4212, 150-161.
- Jeong, K.-S., Kim, D.-K. & Joo, G.-J. (2006). River phytoplankton prediction model by artificial neural network, model performance and selection of input variables to predict time-series phytoplankton proliferations in a regulated river system. *Ecological Informatics*, 1, 235-245.
- Joachims, T. (1999). Transductive interface for text classification using support vector machines. 6th International Conference on Machine Learning, Morgan San Francisco, CA, USA: Kaufmann Publishers Inc.
- Jørgensen, S. E. (2005). Ecological modelling: editorial overview 2000-2005. *Ecological Modelling*, 188, 137-144.
- Karacapilidis, N (2006). An overview of future challenges of decision support technologies. In J. N. D. Gupta, G. A. Forgyionne & M. T. Mora, (Eds.) *Intelligent Decision-making Support Systems* (pp. 386-399). Springer, London Limited.
- Kasabov, N. (2007a). *Evolving Connectionist Systems: the Knowledge Engineering Approach*. Springer.
- Kasabov, N. (2007b). Global, local and personalised modeling and pattern discovery in bioinformatics: An integrated approach. *Pattern Recognition Letters*, 28 (6), 673-685.
- Kasabov, N. & Benuskova, L. (2005). Theoretical and Computational Models for Neuro, Genetic, and Neuro-Genetic Information Processing. In M. Rieth & W. Schommers (Eds.) *Handbook of Computational and Theoretical Nanoscience*, 10, chapter 41, (pp. 1-38), Los Angeles: American Scientific Publishers.
- Kasabov, N., Benuskova, L. & Wysoski, S. G. (2005). Computational neurogenetic modeling: integration of spiking neural networks, gene networks, and signal processing techniques. In W. Duch *et al.* (Eds) *ICANN 2005, LNCS 3697* (pp. 509-514), Springer-Verlag Berlin Heidelberg.

- Kasabov, N. & Pang, S. (2004). Transductive Support Vector Machines and Applications in Bioinformatics for Promoter Recognition. *Neural Information Processing – Letters and Reviews*, 3 (2), 31-38.
- Kasabov, N. (2003a). *Evolving Connectionist Systems, Methods and applications in Bioinformatics, Brain Study and Intelligent Machines*, Springer-Verlag.
- Kasabov, N. (2003b). *Decision Support Systems and Expert Systems*. In M. A. Arbib (Ed.). *The Handbook of Brain Theory and Neural Networks*, 2nd ed. (pp. 316-320), Cambridge, Massachusetts, London, England: The MIT Press.
- Kasabov, N., Venkov, G. And Minchev S. (2003). Neural system for solving the inverse problem of recovering the primary signal waveform in potential transformers. *International Joint Conference on Neural Networks*, 2124-2129.
- Kasabov, N. (2002). *Connectionist-Based decision Support Systems and Expert Systems*. Available:
http://www.aut.ac.nz/research_showcase/research_activity_areas/kedri/downloads/pdf/hbsnn2002-kas.pdf. Last accessed on 18.09.2008.
- Kasabov, N. K. & Song, Q. (2002). DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and Its Application for Time-Series Prediction. *IEEE Transactions on Fuzzy Systems*, 10 (2), 144-154.
- Kasabov, N. (2001). Evolving Fuzzy Neural network for Supervised/Unsupervised Online Knowledge-based learning. *IEEE Transactions on Systems, Man, and Cybernetics- Part B: Cybernetics*, 31 (6), 902-918.
- Kasabov, N., Erzegovesi, L., Fedrizzi, M., Beber, A. & Deng, D. (2000). Hybrid Intelligent Decision Support Systems and Applications for Risk Analysis and Prediction of Evolving Economic Clusters in Europe. In N. Kasabov (Ed) *Future directions for intelligent information systems and information sciences* (pp. 347-372). Springer Verlag Berlin Heidelberg.
 Available:
http://www.aut.ac.nz/resources/research/research_institutes/kedri/downloads/pdf/hidds_final.pdf. Last accessed 28.09.2008.
- Kasabov, N. (1999). Evolving connectionist systems: a theory and a case study on adaptive speech recognition. *International Joint Conference on Neural Networks IJCNN'99*, 10-16 Jul, 5, 3002-3007.

- Kasabov, N. & Fedrizzi, M. (1999). Fuzzy neural networks and evolving connectionist systems for intelligent decision making. 8th International Fuzzy Systems Association World Congress, Taiwan, 17-20 Aug, 30-35.
- Kasabov, N. (1998). Evolving fuzzy neural networks: Theory and Applications for on-line adaptive prediction, decision making and control. *Australian Journal of Intelligent Information Processing Systems*, 5 (3), 154-160.
- Kasabov, N. (1996) *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering*, MIT.
- Katz, D. B. (2003). Making time with taste. Focus on “Taste response variability and temporal coding in the nucleus of the solitary tract of the rat”. *Journal of Neurophysiology*, 90, 1375-1376.
- Katz, D. B., Nicolelis, M. A. L. & Simon, S. A. (2002). Gustatory processing is dynamic and distributed. *Current Opinion in Neurobiology*, 12, 448-454.
- Kim, J. & Kasabov, N. (1999). HyFIS: adaptive neuro-fuzzy inference systems and their application to nonlinear dynamic systems. *Neural Networks*, 12, 1301-1319.
- Kiran, N. R. & Ravi, V. (2007). Software reliability prediction using wavelet neural networks. *International Conference on Computational Intelligence and Multimedia Applications*, 195-199.
- Kistler, W. M. & van Hemmen, J. L. (1999). Short-term synaptic plasticity and network behavior. *Neural Computation*, 11, 1579-1594.
- Kistler, W. M., Gerstner, W. & van Hemmen, J. L. (1997). Reduction of the Hodgkin-Huxley equations to a single-variable threshold model. *Neural Computation*, 9 (5), 1015-1045.
- Knoblauch, A. (2005). Neural associative memory for brain modeling and information retrieval. *Information Processing Letters*, 95, 537-544.
- Kohonen, T. (1982). Self-organizing formation of topologically correct feature maps. *Biological Cybernetics*, 43(1), 59-69.
- Kolman, E. & Margaliot, M. (2005). Are artificial neural networks white boxes?. *IEEE Transactions on Neural Networks*, 16 (4), 844-852.
- Kong, F. W., Ng, G. W., Tan, Y. S. & Tan, C. H. (2007). Evolving fuzzy neural networks in adaptive knowledge bases to support task-oriented decision making for sensor management. 10th International Conference on Information Fusion, 9-12 Jul, 1-8.

- Koprinska, I. & Kasabov, N. (2000). Evolving fuzzy neural network for camera operations recognition. 15th International Conference on Pattern Recognition, 3-7 September, 2, 523-526.
- Korenman, Ya. I. & Kalach, A. V. (2003). Application of multi-sensor system for detection of nitroethane in the air. *Sensors and Actuators B*, 88, 334-336.
- Krantz-Rülcker, C., Stenberg, M., Winqvist, F. & Lundström, I. (2001). Electronic tongues for environmental monitoring based on sensor arrays and pattern recognition: a review. *Analytica Chimica Acta*, 426, 217-226.
- Krasnopolsky, V. M., Fox-Rabinovitz, M. S., Tolman, H. L. & Belochitski, A. A. (2008). Neural network approach for robust and fast calculation of physical processes in numerical environmental models: compound parameterization with a quality control of large errors. *Neural Networks*, 21, 535-543.
- Kubota, N. & Sasaki, H. (2004). Behavior learning of a partner robot with a spiking neural network. *International Conference on Fuzzy Systems*, Budapest, Hungary, 24-29 Jul, 1, 299-304.
- Kukar, M. & Grošelj, C. (2005). Transductive machine learning for reliable medical diagnostics. *Journal of Medical Systems*, 29 (1), 13-32.
- Kukar, M. (2004). Transduction and typicalness for quality assessment of individual classifications in machine learning and data mining. 4th IEEE International Conference on Data Mining ICDM'04, 1-4 Nov, 146-153.
- Kumar, K. V., Ravi., V., Carr, M. & Kiran, N. R. (2008). Software development cost estimation using wavelet neural networks. *Journal of Systems and Software*, 81 (11), 1853-1867.
- Kunkle, D. R. & Merrigan, C. (2002). Pulsed neural networks and their application. Available: <http://www.ccs.neu.edu/home/kunkle/docs/pnn.pdf>. Last accessed on 21.09.2008.
- Labib, R., Audette, F., Fortin, A. & Assadi, R. (2005). Hardware implementation of a new artificial neuron. *International Journal of Neural Systems*, 15 (6), 427-433.
- Landis, J. R. & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics* 33 (1), 159-174.
- La Morgia, V., Bona, F. & Badino, G. (2008). Bayesian modelling procedures for evaluation of changes in wildlife habitat suitability: a case study of roe deer in the Italian Alps. *Journal of Applied Ecology*, 45 (3), 863-872.

- Lankin, G., Worner, S. P., Samarasinghe, S. & Teulon, D. A. J. (2001). Can ANN systems be used for forecasting aphid flight patterns?. *New Zealand Plant Protection*, 54, 188-192.
- Larkin, D., Kinane, A. Muresan, V. & O'Connor, N. (2006). An Efficient Hardware Architecture for a Neural Network Activation Function Generator. In J. Wang *et al.* (Eds.): *Advances in Neural Networks ISNN 2006*, LNCS 3973, 1319-1327, Springer-Verlag Berlin Heidelberg.
- Lee, Y. & Ko, S-B. (2006). FPGA Implementation of a Face Detector Using Neural Networks. *IEEE Conference on Electrical and Computer Engineering CCECE/CCGEI*, Ottawa, May, 1914-1917.
- Lehmann, A., Leathwick, J. R. & Overton, J. McC. (2002). Assessing New Zealand fern diversity from spatial predictions of species assemblages. *Biosecurity and Conservation*, 11, 2217-2238.
- Lek, S. & Guégan, J. F. (1999). Artificial neural networks as a toll in ecological modelling, an introduction. *Ecological Modelling*, 120, 65-73.
- Lek, S., Delacoste, M., Baran, P., Dimopoulos, I., Lauga, J., & Aulagnier, S. (1996). Application of neural networks to modelling nonlinear relationship in ecology. *Ecological Modelling*, 90, 39-52.
- Lencioni, V., Maiolini, B., Marziali, L., Lek, S. & Rossaro, B. (2007). Macro invertebrate assemblages in glacial stream systems: A comparison of linear multivariate methods with artificial neural networks. *Ecological Modelling*, 203, 119-131.
- Li, F. & Wechsler, H. (2004). Watch list face surveillance using transductive inference. *ICBA 2004*, LNCS 3072, Springer-Verlag Berlin Heidelberg, 23-29.
- Li, H., Hou, G., Dakui, F., Xiao, B., Song, L. & Liu, Y. (2007). Prediction and elucidation of the population dynamics of *Microcystic spp.* in Lake Dianchi (China) by means of artificial neural networks. *Ecological Informatics* 2, 184-192.
- Lin, C.-T. & Lee, C. S. G. (1991). Neural-network-based fuzzy logic control and decision systems. *IEEE Transaction on Computers*, 40 (12), 1320-1336.
- Lippitt, C. D., Rogan, J., Toledano, J., Sangermano, F., Eastman, J. R., Mastro, V. & Sawyer, A. (2008). *Ecological Modelling* 210, 339-350.

- Liu, J. & Liang, D. (2005). A Survey of FPGA-Based Hardware Implementation of ANNs. IEEE International Conference on Neural Networks and Brain, Beijing, China 13-15 October, 915-918.
- Loiselle, S., Rouat, J., Pressnitzer, D. & Thorpe, S. (2005). Exploration of rank order coding with spiking neural networks for speech recognition. International Joint Conference on Neural Networks, Montreal, Canada, 31 July-4 August, 2076-2080.
- Lütolf, M., Kienast, F. & Guisan, A. (2006). The ghost of pest species occurrence: improving species distribution models for presence-only data. *Journal of Applied Ecology*, 43, 802-815.
- Maass, W. (1999). Computing with spiking neurons. In W. Maass and C. H. Bishop (Eds.) *Pulsed Neural Networks*, WA Bratford Book, p. 59.
- Maass, W. (1997a). Fast sigmoidal networks via spiking neurons. *Neural Computation*, 9, 279-304.
- Maass, W. (1997b). Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 10 (9), 1659-1671.
- Maass, W., Schnitger, G. & Sontag, E. D. (1991). On the computational power of sigmoid versus boolean threshold circuits. 32nd Annual Symposium on Foundations of Computer Science, 1-4 Oct, 767-776.
- Maeda, Y. & Tada, T. (2003). FRGA implementation of pulse density neural network with learning ability using simultaneous perturbation. *IEEE Transactions on Neural Networks*, 14 (3), 688-695.
- Maguire, L.P., McGinnity, T.M., Glackin, B., Ghani, A., Belatreche, A. & Harkin, J. (2007). Challenges for large-scale implementations of spiking neural networks on FPGAs. *Neurocomputing*, 71 (1-3), 13-29.
- Mak, T. S., Rachmuth, G., Lam, K. P. & Poon, C.-S. (2005). Field programmable gate array implementation of neuronal ion channel dynamics. 2th International IEEE EMBS Conference on Neural Engineering, 16-19 Mar, v-ix.
- Manel, S., Williams, H. C. & Ormerod, J. S. (2001). Evaluating presence-absence models in ecology: the need to account for prevalence. *Journal of applied Ecology*, 38, 921-931.
- Manel, S., Dias, J.-M. & Ormerod, S. J. (1999). Comparing discriminant analysis, neural networks and logistic regression for predicting species distribution: a case study with a Himalayan river bird. *Ecological Modelling*, 120, 337-347.

- Manel, S., Dias, J.-M., Buckton, S. T. & Ormerod, S. J. (1999). Alternative methods for predicting species distribution: an illustration with Himalayan river birds. *Journal of Applied Ecology*, 36, 734-747.
- Marakas, G. M. (2003). *Decision support systems*, 2nd ed., New Jersey: Prentice Hall.
- Marchesi, M., Orlandi G., Piazza, F. & Uncini. A. (1993). Fast Neural Networks without Multipliers. *IEEE Transactions on Neural Networks*, 4 (1), 53-62.
- Margolskee, R. F. (1993). The molecular biology of taste transduction. *BioEssays*, 15 (10), 645-650.
- Marian, I. D. (2002). A biologically inspired model of motor control of direction. MSc Thesis, University College, Dublin.
- Marshal, S. J. & Harrison, R. F. (1991). The 2nd International Conference on Artificial Neural Networks, 18-20 November, 39-43.
- Martinelli, E., D'Amico, A. & Di Natale, C. (2006). Spike encoding of artificial olfactory sensor signals. *Sensors and Actuators B*, 119, 234-238.
- Martinez, D. & Hugues, E. (2004). A spiking neural model of the locust antennal lobe: towards neuromorphic electronic noses inspired from insect olfaction. In .W. Gardner & J. Yinon (Eds.), *Electronic Noses and Sensors for the Detection of Explosives* (pp. 209-234), Springer.
- Mattes, R. D. (2005). Fat taste and lipid metabolism in humans. *Physiology and Behavior*, 86 (5), 691-697.
- McCulloch, W. S. & Pitts, W. H. (1943). A logical calculus of the ideas imminent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 115-133.
- McClurkin, J. W., Optican, L. M., Richmond, B. J. & Gawne, T. J. (1991). Concurrent processing and complexity of temporally encoded neural messages in visual processing. *Science*, 253 (5020), 675-677.
- McGraw, R. M., Lammers, C. & Steinman, J. S. (2004). Software framework in support of dynamic situation assessment and predictive capabilities for JSB-RD. In D. A. Travisani & A. F. Sisti (Eds.) *Enabling Technologies for Simulation Science VIII* (pp. 468-478), 5423, SPIE.
- McPherson, J. M., Jetz, W. & Rogers, D. (2004). The effects of species' range sizes on the accuracy of distribution models: ecological phenomenon or statistical artefact?. *Journal of Applied Ecology*, 41, 811-823.

- Merchant, S., Peterson, G.D., Ki Park, S. & Kong, S.G. (2006). FPGA Implementation of Evolvable Block-based Neural Networks. IEEE Congress on Evolutionary Computation, Vancouver, Jul 16-21, 3129-3136.
- Metz, C. E. (1978). Basic principles of ROC analysis. Seminar in Nuclear Medicine, VIII (4), 283-298.
- Mikhaleva, N. M. & Kulapina, E. G. (2005). Arrays of nonselective nonionic-surfactant sensors for the separate determination of the homologues of polyoxyethylated nonylphenols. Journal of analytical chemistry, 60 (6), 573-580.
- Mohan, N. & Kasabov, N. (2005). Transductive modeling with GA parameter optimization. IEEE International Joint Conference on Neural Networks IJCNN '05, 31 July-4 August, 2, 839-844.
- Mohanty, B. K. & Bhasker, B. (2005). Product classification in the internet business–fuzzy approach. Decision support systems, 38, 611-619.
- Monserud, R. A. & Leemans, R. (1992). Comparing global vegetation maps with the Kappa statistic. Ecological Modelling, 62, 275-293.
- Murray, A. F. (1999). Pulse-based computation in VLSI neural networks. In W. Maass & C. M. Bishop (Eds.) Pulse Neural Networks (pp. 87-109). London: A Bradford Book, The MIT Press.
- Narayanan, A. & Menneer, T. (2000). Quantum artificial neural network architectures and components. Information Sciences, 128, 231-255.
- Natschläger, T. & Ruf, B. (1998). Spatial and temporal pattern analysis via spiking neurons. Network: Computation in Neural Systems, 9, 319-332.
- Nedjah, N. & de Macedo Mourelle, L. (2003). FPGA-based hardware architecture for neural networks: binary radix vs. stochastic. 16th Symposium on Integrated Circuits and System design SBCCI'03, 111-116.
- NeuCom <http://www.theneucom.com/>. Last accessed on 13.09.2008.
- NEURON <http://www.neuron.yale.edu/neuron/> . Last accessed on 13.09.2008.
- Neter, J., Wasserman, W. & Kutner, M.H. (1990). Applied Linear Statistical Models 3rd ed., Homewood, IL: Irwin.
- Ng, G. S., Murali, T., Shi, D. & Wahab, A. (2004). Application of EFuNN for the classification of handwritten digits. International Journal of Computers, Systems and Signals, 5 (2), pp. 27-35.

- Nielsen, J. & Lund, H. H. (2003). Spiking neural building block robot with Hebbian learning. IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, Nevada, October, 1363-1369.
- NZ Ministry of Agriculture and Forestry.
<http://mafuwsp6.maf.govt.nz/uor/cgi/uor.pl/list>. Last accessed on 13.09.2008.
- NZ Ministry of agriculture and Forestry. (2006). *Didymosphenia geminata* economic impact assessment. <http://www.biosecurity.govt.nz/files/pests/didymo/didymo-econ-ia-mar-06.pdf>. Last accessed on 3/02/2009.
- Olden, J. D., Poff, N. L. & Bledsoe, B. P. (2006). Incorporating ecological knowledge into ecoinformatics: an example of modeling hierarchically structured aquatic communities with neural networks. *Ecological Informatics* 1, 33-42.
- Olshausen, B. A. & Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by V1?. *Vision Research*, 37 (23), 3311-3325.
- Oram, M. W. & Perrett, D. I. (1992). Time course of neural response discriminating different views of the face and head. *Journal of Neurophysiology*, 68, 70-84.
- Osborne, P. E., Alonso, J. C. & Bryant, R. G. (2001). Modelling landscape-scale habitat use using GIS and remote sensing: a case study with great bustards. *Journal of Applied Ecology*, 38, 458-471.
- Ozawa, S., Toh, S. L., Abe, S., Pang, S. & Kasabov, N. (2005). Incremental learning of feature space and classifier for face recognition. *Neural Networks*, 18, 575-584.
- Özesmi, S. L. & Özesmi, U. (1999). An artificial neural network approach to spatial habitat modelling with interspecific interactions. *Ecological Modelling*, 116, 15-31.
- Pang, S. & Kasabov, N. (2004). Inductive vs. transductive, global vs. local models: SVM, TSVM, and SVMT for gene expression classification problems. *IEEE International Joint Conference on Neural Networks*, 25-29 Jul, 2, 1197-1202.
- Papoyan, E. V. (1997). Antidromal and synaptic activation of neurons of the associative parietal cortex of the cat brain elicited by spike activity from the intrinsic nuclei of the pons. *Neuroscience and Behavioral Physiology*, 27 (6), 695-701.
- Park, Y.-S., Rabinovich, J. & Lek, S (2007). Sensitivity analysis and stability patterns of two-species pest models using artificial neural networks. *Ecological Modelling*, 204, 427-438.

- Park, Y.-S., Chang, J. B., Lek, S., Cao, W. X. & Brosse, S. (2003). Conservation strategies for endemic fish species threatened by the Three Gorges Dam. *Conservation Biology*, 17 (6), 1748-1758.
- Paruelo, J. M. & Tomasel, F. (2007). Prediction of functional characteristics of ecosystems: a comparison of artificial neural networks and regression models. *Ecological Modelling*, 98, 173-186.
- Paterson, B., Stuart-Hill, G., Underhill, L. G., Dunne, T. T., Schinzel, B., Brown, C. *et al.* (2008). A fuzzy decision support tool for wildlife translocations into communal conservancies in Namibia. *Environmental Modelling and Software*, 23, 521-534.
- Peacock, L., Worner, S. & Sedcole, R. (2006). Climate variables and their role in site discrimination of invasive insect species distributions. *Environmental Entomology*, 35 (4), 958-963.
- Peacock, L. (2005). Eco-climatic assessment of the potential establishment of exotic insects in New Zealand. PhD Thesis, Lincoln University, Canterbury, New Zealand.
- Pearce, J. & Ferrier, S. (2000a). Evaluating the predictive performance of habitat models developed using logistic regression. *Ecological Modelling*, 133, 225-245.
- Pearce, J. & Ferrier, S. (2000b). An evaluation of alternative algorithms for fitting species distribution models using logistic regression. *Ecological Modelling*, 128, 127-147.
- Pearson, M. J., Pipe, A. G., Mitchinson, B., Gurney, K., C. Melhuish, C., Gilhespy, I. & Nibouche, M. (2007). Implementing spiking neural networks for real-time signal-processing and control applications: a model-validation FPGA approach. *IEEE Transactions on Neural Networks*, 18 (5), 1472-1487.
- Pearson, M. J., Melhuish, C., Pipe, A. G., Nibouche, M., Gilhespy, I., Gurney, K., & Mitchinson, B. (2005). Design and FPGA implementation of an embedded real-time biologically plausible spiking neural network processor. *International Conference on Field Programming Logic and Applications*, 24-26 Aug, 582-585.
- Pearson, M., Gilhespy, I., Gurney, K., Melhuish, C., Mitchinson, B., Nibouche, M., & Pipe, A., (2005). A real-time, FPGA based, biologically plausible neural network processor. 15th International Conference on Artificial Neural Networks ICANN 2005, September 11-15, LNCS 3697, 1021-1026.

- Pearson, R. G., Dawson, T. P., Berry, P. M. & Harrison, P. A. (2002). SPECIES: A spatial evaluation of climate impact on the envelope of species. *Ecological Modelling*, 154, 289-300.
- Perrinet, L., Delorme, A., Samuelides, M. & Thorpe, S. J. (2001). Networks of integrate-and-fire neuron using rank order coding A: How to implement spike time dependent Hebbian plasticity. *Neurocomputing*, 38-40, 817-822.
- Peterson, A. T. & Vieglais, D. A. (2001). Predicting species invasions using ecological niche modeling: new approaches from bioinformatics attack a pressing problem. *BioScience*, 51 (5), 363-371.
- Peterson, R. S., Panzeri, S. & Diamond, M. E. (2002). Population coding in somatosensory cortex. *Current Opinion in Neurobiology*, 12, 441-447.
- Peterson, R. S., Panzeri, S. & Diamond, M. E. (2001). Population coding of stimulus location in rat somatosensory cortex. *Neuron*, 32, 503-514.
- Pilson, D. (1992). Aphid distribution and the evaluation of goldenrod resistance. *Evolution*, 46 (5), 1358-1372.
- Prut, Y., Vaadia, E., Bergman, H., Haalman, I., Slovin, H. & Abeles, M. (1998). Spatiotemporal structure of cortical activity: properties and behavioral relevance. *Journal of Neurophysiology*, 49, 2857-2874.
- Rafoss, T. (2003). Spatial Stochastic Simulation Offers Potential as a Quantitative Method for Pest Risk Analysis. *Risk Analysis*, 23 (4), 651-661.
- Raman, B., Sun, P. A., Gutierrez-Galvez, A. & Gutierrez-Osuna, R. (2006). Processing of chemical sensor arrays with a biologically inspired model of olfactory coding. *IEEE Transactions on Neural Networks*, 17 (4), 1015-1024.
- Reaz, M. B. I., Islam, S. Z., Ali, M. A. M. & Sulaiman, M. S. (2002). FPGA realization of backpropagation for stock market prediction. 9th International Conference on Neural Information Processing ICONIP'02, 18-22 Nov, Vol. 2, 960-964.
- Reed, D. R., Tanaka, T. & McDaniel, A. H. (2006). Diverse tastes: genetics of sweet and bitter perception. *Physiology and behavior*, 88 (3), 215-226.
- Remm, K. (2004). Case-based predictions for species and habitat mapping. *Ecological Modelling*, 177, 259-281.
- Riul Jr., A., de Sousa, H. C., Malmegrim, R. R., dos Santos Jr., D. S., Carvalho, A. C. P. L. F., Fonseca, F. J. *et al.* (2004). Wine classification by taste sensors made from ultra-thin films and using neural networks. *Sensors and Actuators B*, 98, 77-82.

- Riul Jr, A., dos Santos Jr, D. S., Wohnrath, K., Di Tommazo, R., Carvalho, A. C. P. L. F., Fonseca, F. J. *et al.* (2002). An artificial taste sensor: efficient combination of sensors made from Langmuir-Blodgett films of conducting polymers and a ruthenium complex and self-assembled films of an Azobenzene-containing polymer. *Langmuir*, 18, 239-245.
- Robertson, M. P., Peter, C. I., Villet, M. H. & Ripley, B. S. (2003). Comparing models for predicting species' potential distributions: a case study using correlative and mechanic predictive modelling techniques. *Ecological Modelling* 164, 153-167.
- Rochel, O., Martinez, D., Hugues, E. & Sarry, F. (2002). Stereo-olfaction with a sniffing neuromorphic robot using spiking neurons. 6th European Conference on Solid-State Transducers (EUROSENSORS) Prague, Check Republic, p. 4.
- Rodriguero, M. S. & Gorla, D. E. (2004). Latitudinal gradient in species richness of New World Triatominae (Reduviidae). *Global Ecology and Biogeography*, 13, 75-84.
- Rodríguez, M. Á., Belmontes, J. A. & Hawkins, B. A. (2005). Energy, water and large-scale patterns of reptile and amphibian species richness in Europe". *Acta Oecologica*, 28, 65-70.
- Roggen, D., Hofmann, S., Thoma, Y. & Floreano, D. (2003). Hardware spiking neural network with run-time reconfigurable connectivity in an autonomous robot. 2003 NASA/Dod Conference on Evolvable Hardware, 9-11 July, 189-198.
- Rojas, R. (1996). *Neural networks, A Systematic Approach*, Berlin: Springer-Verlag, p. 31.
- Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65 (6), 386-408.
- Rushton, S. P., Ormerod, S. J. & Kerby, G. (2004). New paradigms for modelling species distributions?. *Journal of Applied Ecology* 41, 193-200.
- Sahoo, G. B., Ray, C. & Wade, H. F. (2005). Pesticide prediction in ground water in North Carolina domestic wells using artificial neural networks. *Ecological Modelling*, 183, 29-46.
- Sauther, V. L. (1997). *Decision support systems*. John Wiley & Sons, Inc., p. 13.
- Savich, A. W., Moussa, M. & Areibi, S. (2007). The impact of arithmetic representation on implementing MLP-BP on FPGAs: a study. *IEEE Transactions on Neural Networks*, 18 (1), 240-252.

- Schadt, S., Revilla, E., Wiegand, T., Knauer, F., Kaczensky, P., Breitenmoser, U. *et al.* (2002). Assessing the suitability of central European landscapes for the reintroduction of Eurasian lynx. *Journal of Applied Ecology*, 39, 189-203.
- Schrauwen, B. & Van Campenhout, J. (2006). Parallel hardware implementation of a broad class of spiking neurons using serial arithmetic. European Symposium on Artificial Neural Networks ESANN'2006, Belgium 26-28 Apr, 623-628.
- Schwaighofer, A. & Tresp, V. (2002). Transductive and inductive methods for approximate Gaussian process regression. In S. Backer, S. Thrun & K. Obermayer (Eds.) *Advances in Neural Information processing Systems*, 15, 953-960.
- Sejnowski, T. J. (1995). Pattern recognition. Time for a new neural code?. *Nature*, 376 (6535), 21-22.
- Shadlen, M. N. & Newsome, W. T. (1998). The variable discharge of cortical neurons: implications for connectivity, computation, and information coding. *The Journal of Neuroscience*, 18 (10), 3870-3896.
- Softky, W. R. (1995). Simple codes versus efficient codes. *Current Opinion in Neurobiology*, 5, 239-247.
- Sohn, J.-W., Zhang, B.-T. & Kaang, B.-K. & (1999). Temporal pattern recognition using a spiking neural network with delays. *International Joint Conference on Neural Networks*, Washington, 2590-2593.
- Soltic, S. & Zuppicich, A. (2008). Comparison between software and FPGA implementation of ESNN in terms of processing speed. 15th Electronics New Zealand Conference ENZCon08, Auckland, 24-25 Nov, 56-61.
- Soltic, S., Wysoski, S. G. & Kasabov, N. (2008). Evolving spiking neural networks for taste recognition. *International Joint Conference on Neural Networks IJCNN'08*, Hong Kong, 1-6 Jun, 2092-2098.
- Soltic, S. & Peacock, L. (2006a). A comparison of inductive and transductive models for predicting the establishment potential of the exotic scale, *Aspidiella hartii* (Cockerell), in New Zealand. *Bulletin of Applied Computing and Information Technology*, 4 (2).
- Soltic, S. & Peacock, L. (2006b). Evolving connectionist systems in ecological modelling. Presented at the 1st Korean-New Zealand Joint Workshop on Advance of Computational Intelligence methods and Applications, Auckland, New Zealand, 8 Feb.

- Soltic, S., Pang, S., Peacock, L. & Worner, S. (2004a). Evolving computation offers potential for estimation of pest establishment. *International Journal of Computers, Systems and Signals IJCSS*, 5 (2), pp. 37-44.
- Soltic, S., Pang, S., Kasabov, N., Worner, S. & Peacock, L. (2004b). Dynamic neuro-fuzzy inference and statistical models for risk analysis of pest establishment. *International Conference on Neural Information Processing, ICONIP'04, Lecture Notes in Computer Science 3316*, 971-976.
- Soltic, S., Pang, P., Worner, S. & Peacock, L. (2003). Evolving computation offers potential as a probabilistic evaluation for pest risk maps generation. In N. Kasabov & Z. S. H. Chan (Eds). *Proceedings of the Conference on neuro-computing and evolving intelligence NCEI'03*, 20-21 Nov, 104-106.
- Song, Q. & Kasabov, N. (2005). NFI: A neuro-fuzzy inference method for transductive reasoning. *IEEE Transactions on Fuzzy Systems*, 13 (6), 799-808.
- Stein, J. F. & Stoodley, C. J. (2006). *Neuroscience: an introduction*, John Wiley and sons.
- Stein, R. B. (1967). The information capacity of nerve cells using a frequency code. *Biophysical Journal*, 7, 797-826.
- Stitt, J. P., Gaumond, R. P., Fraizer, J. L. & Hanson, F. E. (1997). A comparison of neural spike classification techniques. *19th International Conference IEEE/EMBS, Chicago, IL, USA, 30 Oct-2 Nov*, 1092-1094.
- Strauss, B. & Biedermann, R. (2007) Evaluating temporal and spatial generality: How valid are species-habitat relationship models?. *Ecological Modelling*, 204, 104-114.
- Stockwell, D. R. B. & Peterson, A. T. (2002). Effects of sample size on accuracy of species distribution models. *Ecological Modelling*, 148, 1-13.
- StrongWare[®], <http://www.strongware.com/>. Last accessed on 12.09.2008.
- Suárez-Seoane, S., Osborne, P. E. & Alonso, J. C. (2002). Large-scale habitat selection by agricultural steppe birds in Spain: identifying species-habitat response using generalized additive models. *Journal of Applied Ecology*, 39, 755-771.
- Ševčík, P. (2005). Implementation design of pulse coded neural network neuron into field programmable gate array device. *Applied Electronics*, 6-7 Sep, 197-200.

- Tetko, I. V. & Villa, A. E. P. (2001). A pattern grouping algorithm for analysis of spatiotemporal patterns in neuronal spike trains. 2. Application to simultaneous single unit recordings. *Journal of Neuroscience Methods*, 105, 15-24.
- Thorpe, S., Delorme, A. & Van Rullen, R. (2001). Spike-based strategies for rapid processing. *Neural Networks*, 14, 715-725.
- Toko, K. (2000). Taste sensor. *Sensors and actuators B*, 64, 205-215.
- Torres-Huitzil, C. (2006). Area-efficient implementation of a pulse-mode neural network. *International Conference on Field Programmable Logic and Applications*, 28-30 Aug, 1-4.
- Turban, E., Aronson, J. E. & Liang, T.-P. (2005). *Decision support-systems and intelligent systems*, 7th edition, New Jersey: Pearson-Practice Hall, p. 103.
- Ulrichs, C. & Hopper, K. R. (2007). Predicting insect distribution from climate and habitat data. *BioControl*, Available: <http://www.springerlink.com/content/e36j84821518232x/>. Last accessed on 18.09.2008.
- Vaňhara, J., Muráriková, N., Malenovský, I. & Havel, J. (2007). Artificial neural networks for fly identification: a case study from the genera *Tachina* and *Ectophasia* (Diptera, Tachinidae). *Biologia*, 62 (4), 462-469.
- Van Rullen, R., Guyonneau, R. & Thorpe, S. J. (2005). Spike times make sense. *TRENDS in Neuroscience*, 28 (1), 1-4.
- Van Rullen, R. & Thorpe, S. J. (2001). Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex. *Neural Computation*, 13, 1255-1283.
- Van Rullen, R., Gautrais, J., Delorme, A. & Thorpe, S. (1998). Face processing using one spike per neurone. *BioSystems*, 48, 229-239.
- Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley Inter-Science, p.736.
- Verhagen, J. V. & Engelen, L. (2006). The neurocognitive bases of human multimodal food perception: sensory integration. *Neuroscience and Biobehavioral Reviews*, 30 (5), 613-650.
- Varkevisser, B., Peterson, D., Ogura, T. & Kinnamon, S. C. (2001). Neural networks distinguish between taste qualities based on receptor cell population responses. *Chemical senses*, 26, 499-505.

- Venayagamoorthy, G. K. & Gaurav, S. (2005) Quantum-inspired evolutionary algorithms and binary particle swarm optimization for training MLP and SRN neural networks. *Journal of Computational and Theoretical Nanoscience*, 2, 561-568.
- Verstraeten, D., Schrauwen, B., Stroobandt, D. & Van Compenhout, J. (2005). Isolated word recognition with the Liquid State Machine: a case study. *Information Processing Letters*, 95, 521-528.
- Vitabile, S., Conti, V., Gennaro, F. & Sorbello, F. (2005). Efficient MLP Digital Implementation on FPGA. *IEEE 8th Euromicro Conference on Digital System Designs DSD'05*, 30 Aug-3 Sep, 218-222.
- Vogelstein, R. J., Mallik, U., Vogelstein, J. T. & Cauwenberghs, G. (2007). Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses. *IEEE Transactions on Neural Networks*, 18 (1), 253-256.
- VonPlinsky, M. J. & Crowder, E. (2002). Decision integration and support engine (DISE) for dynamic aircraft and ISR asset tasking/retasking decision support for the AOC. In A. F. Sisti & D. A. Travisani (Eds.) *Enabling Technologies for Simulation Science VI* (pp. 311-322), SPIE 4716.
- Vreeken, J. (2003). Spiking neural networks, an introduction. Available: <http://citeseer.ist.psu.edu/564316.html>. Last accessed on 15.09.2008.
- Wagner, H., Brill, S., Kempner, R. & Carr, C. E. (2005). Microsecond precision of phase delay in the auditory system of the barn owl. *Journal of Neurophysiology*, 94, 1655-1658.
- Waldemark, J., Millberg, M., Lindblad, T., Waldemark, K. & Becanovic, V. (2000). Implementation of a pulse coupled neural network in FPGA. *International Journal of Neural Systems, Special Issue on Hardware Implementations*, 10 (3), 171-177.
- Waterhouse, D. F. and Sands, D. P. A. (2001). *Classical Biological Control of Arthropods in Australia*. Australian Centre for International Agricultural Research, Canberra, Australia.
- Watts, M. J. (2009). A decade of Kasabov's evolving connectionist systems: a review. *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, 39 (3), 253-269.
- Watts, M. J. & Worner, S. P. (2008). Using artificial neural networks to determine the relative contribution of abiotic factors influencing the establishment of insect pest species. *Ecological Informatics*, 3 (1), 64-74.

- White, B. A. & Elmasry, M. I. (1992). The digi-neocognitron: a digital neocognitron neural network model for VLSI. *IEEE Transactions on Neural Networks*, 3 (1), 73-85.
- White, J. & Kauer, J. S. (1999). Odor recognition in an artificial nose by spatio-temporal processing using an olfactory neural network. *Neurocomputing*, 26-27, 919-924.
- Wide, P., Winqvist, F., Bergsten, P. & Petriu, E. M. (1998). The human-based multisensor fusion method for artificial nose and tongue sensor data. *IEEE Transactions on Instrumentation and measurement*, 47 (5), 1072-1077.
- Willems, W., Goethals, P., Van den Eynde, D., Van Hoey, G., Van Lancker, V., Verfaillie, E. *et al.* (2008). Where is the worm? Predictive modelling of the habitat preferences of the tube-building polychaete *Lanice conchilega*. *Ecological Modelling*, 212, 74-79.
- Williams, J. B. & Poff, N. L. (2006). Informatics software for the ecologist's toolbox: a basic example. *Ecological Informatics* 1, 325-329.
- Williamson, M. H. & Fitter A. (1996). The characters of successful invaders. *Biological Conservation* 78, 163-170.
- Woodford, B. J. (2008). Evolving neurocomputing systems for horticulture applications. *Applied Soft Computing*, 8, 564-578.
- Woodford, B. J., Wearing, C. H., Walker, J. T. S. & Kasabov, N. K. (1999). An adaptive agent-based distributed system for pest management. *ICONIP/ANZIIS/ANNES'99 Workshop, Dunedin, New Zealand, 22-24 Nov.*
- Worner, S. & Gevrey, M. (2006). Modelling global insect pest species assemblages to determine risk of invasion. *Journal of Applied Ecology*, 45 (5), 858-867.
- Worner, S. P. (2002). Predicting the invasive potential of exotic insects. In G. J. Hallman & C. P. Schwalbe (Eds.) *Invasive Arthropods in Agriculture: Problems and Solutions* (pp. 119-137), New Hampshire: Science Publishers Inc..
- Wu, Q. X., McGinnity, T. M., Maguire, L. P., Glackin, B. & Belatreche, A. (2006). Learning under weight constraints in networks of temporal encoding spiking neurons. *Neurocomputing*, 69, 1912-1922.
- Wysoski, S.G. (2008). Evolving spiking neural networks for adaptive audiovisual pattern recognition. PhD Thesis, Auckland University of Technology, Auckland, New Zealand.

- Wysoski, S. G., Benuskova, L. & Kasabov, N. (2007a). Adaptive spiking neural networks for audiovisual pattern recognition. *ICONIP07, Lecture Notes in Computer Science*, Springer-Verlag, New York. 406-415.
- Wysoski, S. G., Benuskova, L. & Kasabov, N. (2007b). Text-independent speaker authentication with spiking neural networks. *International Conference on Artificial Neural Networks, ICANN, Porto*, 758-767.
- Wysoski, S. G., Benuskova, L. & Kasabov, N. (2006). On-line learning with structural adaptation in a network of spiking neurons for visual pattern recognition. *ICANN 2006, Lecture Notes in Computer Science 4131, Part I*, Springer, 61-70.
- Xu, R. and Wunsch II, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16 (3), 645-678.
- Yoshimura, H., Kato, N., Sugai, T., Honjo, M., Sato, J., Segami, N. & Onoda, N. (2004). To-and-fro optical voltage signal propagation between the insular gustatory and perietal oral somatosensory areas in rat cortex slices. *Brain Research*, 1015, 114-121.
- Zampolli, S., Elmi, I., Ahmed, F., Passini, M., Cardinali, G. C., Nicoletti, S. & Dori, L. (2004). An electronic nose based on solid state sensor arrays for low-cost indoor air quality monitoring applications. *Sensors and Actuators B*, 101, 39-46.
- Zaniewski, A. E., Lehmann, A. & Overton, J. McC. (2002). Predicting species spatial distributions using presence-only data: a case study of native New Zealand ferns. *Ecological modelling*, 157, 261-280.
- Zanchettin, C. & Ludermir, T. B. (2004). Evolving fuzzy neural networks applied to odor recognition in an artificial nose. *IEEE International Joint Conference on Neural Networks*, 25-29 July, 675-680.
- Zelikovitz, S. (2004). Transductive LSI for short text classification problems. *17th International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004*, 2, 556-561.
- Zhang, K. & Sejnowski, T. J. (1999). Neural tuning: to sharpen or broaden?. *Neural Computation*, 11, 75-84.
- Zhang, L., Gove, J. H. & Heath, L. S. (2005). Spatial residual analysis of six modelling techniques. *Ecological Modelling*, 186, 154-177.
- Zhang, W. & Zhang, X. (2008). Neural network modelling of survival dynamics of holometabolous insects: a case study. *Ecological Modelling*, 211, 433-443.

- Zhao, F-I., Shen, T., Kaya, N., Lu, S.-G., Cao, Y. & Herness, S. (2005). Expression, physiological action, and coexpression patterns of neuropeptide Y in rat taste-bud cells. *National Academy of Sciences of the United States of America*, 102 (31), 11100-11105.
- Zhu, J. & Sutton, P. (2003). FPGA implementations of neural networks – a Survey of a decade of progress. In: P. Cheung, G. Constantinides and J. de Sousa *The Thirteenth International Conference on Field-Programmable Logic and Applications*, Lisbon, Portugal, 1-3 Sep, 2003.
- Zhuang, H., Low, K.-S. & Yau, W.-Y. (2007). A pulse neural network with on-chip learning and its practical applications. *IEEE Transactions on Industrial Electronics*, 54 (1), 34-42.
- Zucker, R. S. & Regehr, W. G. (2002). Short-term synaptic plasticity. *Annual Review of Physiology*, 64, 355-405.
- Zuppicich, A & Soltic, S. (2008). FPGA implementation of an evolving spiking neural network. 15th International Conference on Neural Information Processing ICONIP'08, Workshop on Neurocomputing and Evolving Intelligence, Auckland, New Zealand, 25-28 Nov, 23-24, LNCS, vol. 5506/5507.

Appendix

MATLAB code for LPAM

```
%=====
%= Code for LPAM
%= Takes in data with class designators (data_in) and class
%= labels 0 and 1.
%=
%= Uses: shuffle, ecml, denfis1 and denfis2s - all from
%= http://www.aut.ac.nz/research/research_institutes/kedri/
%=
%= @ param data_in - data with the class
%= @ param th - ECM threshold, determines the number of clusters
%= @ thD - dichotomising threshold
%=
%= @ return predictions - predicted values for all data samples,
%= @ return acc - accuracy.
%= Date: Mar, 2005; Dec 2008
%=====
function [predictions, acc] = LPAM(data_in, dthrV, thD)

[rows, columns] = size(data_in);
predictions = [];
mix = [];
data = [];
class = [];

% shuffle the original data
mix = shuffle(data_in);
data = mix(:, 1: columns - 1); % data without class labels
class = mix(:, columns); % extracts the class labels

% ECM clustering
parm0.dthr = th; % ECM threshold
res = ecml(data, parm0); % run ECM
[r, NoECMClusters] = size(res.Cluster);
% Probability evaluation
Y_Data=[];
X_Data=[];
for i=1:NoECMClusters
    tmp=res.Cluster{i}; % Cluster i
    Y_Data=[Y_Data length(find(class(tmp, 1)==1))/length(tmp)];
    [r,c] = size(data(tmp,:));
    if r == 1
        X_Data = cat(1, X_Data, data(tmp,:));
    else
        X_Data= cat(1, X_Data, mean(data(tmp,:)));
    end
end

% knowledge discovery using DENFIS
resd = denfis1([X_Data Y_Data']); % Training
ressd = denfis2s(data, resd); % Predictions
f = ressd.Oriy';
```

```

% Normalization of the predicted results
fmin = min(f);
fmax = max(f);
fnorm = (f + abs(fmin)) / (fmax - fmin);
predictions = cat(2, fnorm, class);
correct = 0;
total = 0;
[rNorm,cNorm] = size(fnorm);
for i = 1 : rNorm
    if fnorm(i) >= thD
        total = total + 1;
        switch (class(i))
            case 1
                correct = correct + 1;
            end
        end
end
acc = correct/total;

disp(' ');
disp(['          Accuracy: ' sprintf('%5.3f',acc)]);

%=====
%= Leave-one-out predictions using LPAM.
%=
%= Class labels 0 and 1. Seven dichotomising thresholds:
%=          0.3, 0.4, 0.5, 0.6, 0.7,0.8 & 0.9
%= Invokes: shuffle, ecml, denfis1 and denfis2s - all from
%= http://www.aut.ac.nz/research/research\_institutes/kedri/
%=
%= @ param data_in - input data (must contain the class labels)
%= @ param th - determines the number of clusters
%=
%= @ output predictions - predicted values for all locations,
%= @ output acc - accuracy over 7 dichotomising thresholds
%=
%= Date: Feb, 2005; Dec 2008
%=====
function [predictions, acc] = LPAM_LOO(data_in, th)

[samples, col] = size(data_in);
mix = []; // shuffled data
p = [];
acc = []; // accuracy

% shuffle the original data
mix = shuffle(data_in);

% repeat for each sample
for i = 1:samples

    % remove current test sample from the original data
    sample_i = mix(i, 1:col-1);
    % form a training set
    train_b = mix(1:i-1, :);
    train_a = mix(i+1:samples, :);

```

```

train= cat(1, train_b, train_a);

% cluster using ecm
parm0.dthr = th;
res = ecml(train(:, 1:col-1), parm0);
[r, NoClusters] = size(res.Cluster);
% probability evaluation
Y=[];
X=[];
for j=1:NoClusters
    tmp=res.Cluster{j};
    Y=[Y length(find(train(tmp, col)==1))/length(tmp)];
    [r,c] = size(train(tmp,:));
    if r == 1
        X = cat(1, X, train(tmp,1:col-1));
    else
        X= cat(1, X, mean(train(tmp,1:col-1)));
    end
end

% knowledge discovery using DENFIS
resd = denfis1([X Y']); % training
ressd = denfis2s(sample_i, resd); % estimation
f = ressd.Oriy';
p = [ p f ];
end
p = p';

% normalise predictions
pmin = min(p);
pmax = max(p);
predictions = (p + abs(pmin)) / (pmax - pmin);
[r,c] = size(pnorm);

% evaluate accuracy at different thresholds
for step = 0.3:0.1:0.9
    t = 0;
    correct = 0;
    for i = 1 : r
        if pnorm(i) >= step
            t = t + 1;
            switch (mix(i, col))
                case 1
                    correct = correct + 1;
            end
        end
    end
    correct = correct/t;
    acc = [acc correct];
end

```