Modular Neural Networks for Analysis of Flow Cytometry Data

By

Arnaud Autret

A submission presented in partial fulfilment of the requirements of the University of Glamorgan for the degree Doctor of Philosophy

School of Computing

University of Glamorgan

2003

Acknowledgments

I would like to begin by expressing gratitude to the University Of Glamorgan for giving the opportunity to start a PhD. I would also like to thanks Mr Colin Morris and Dr Paul Angel, my first and second supervisor, for their support, guidance and patience over these three and a half years.

Many thanks to Dorothée, Georgios, Charlie, Vlado, Jonathan and all the other research students for giving me a chance to enjoy myself during the weekends and for the interesting talks we had when I was having a break.

I would like to thanks Dr Malcom Wilkins for allowing me to use his RBF software and Dr Joachims Thorsten for authorising me to use his SVM Light software and port it to Microsoft Windows.

I would also like to thanks Prof Lynne Boddy and the University Of Cardiff for providing the Flow Cytometer data used all along this project

Abstract

In predicting environmental hazards or estimating the impact of human activities on the marine ecosystem, scientists have multiplied the need for sample analysis. The classical microscopic approach is time consuming and wastes the talent and intellectual abilities of trained specialists. Therefore, scientists developed an automated optical tool, called a Flow Cytometer (FC), to analyse samples quickly and in large quantities. The flow cytometer has successfully been applied to real phytoplankton studies. However, analysis of the data extracted from samples is still required. Artificial Neural Networks (ANNs) are one of the tools applied to FC data analysis.

Despite several successful applications, ANNs have not been widely adopted by the marine biologist community, as they can not possible to change the number of species in the classification problem without retraining of the full system from scratch. Training is time consuming and requires expertise in ANNs. Moreover, most ANN paradigms cannot cope effectively with unknown data, such as data coming from new phytoplankton species or from species outside the scope of the studies.

This project developed a new ANN technique based on a modular architecture that removes the need for retraining and allows unknowns to be detected and rejected. Furthermore, the Support Vector Machine architecture is applied in this domain for the first time and compared against another ANN paradigm called Radial Basis Function Networks. The results show that the modular architecture is able to effectively deal with new data which can be incorporated into the ANN architecture without fully retraining the system.

Table of contents

Chapter 1: Introduction

1.1 Domain overview

More and more people recognize that our biosphere is fragile and that human activities dramatically affect it. In order to quantify the influence of these activities on the ecosystem, it is necessary to understand it. The oceans have a major effect on the climate and the global ecosystem in general. One of the most important building blocks of the oceans ecology is phytoplankton. Phytoplanktons are a group of microscopic vegetal organisms. However, because of their huge biomass, the phytoplankton influences the chemistry of the seas and is the most important photosynthetic life on earth. The photosynthesis transforms water, carbon dioxide and other material into energetic organic compounds using the sun's energy. Oxygen (O_2) is a waste product of this transformation. Moreover, once in high altitude this oxygen molecule combines to create the ozone layer (O_3) that protects us from the sun's ultra-violet. Phytoplankton is also the lowest and most important link in the oceans food chain, a huge variety of animals ranging from zooplankton to whales feeds on it. Thus, variations in phytoplankton communities affect or jeopardize the survival of many different species. Furthermore, because it is extremely sensitive to the local environment, phytoplankton can be used to detect pollution. Finally, some species of phytoplankton are extremely toxic and a bloom of those species can result in the death of thousands of fish or even poison the local human population.

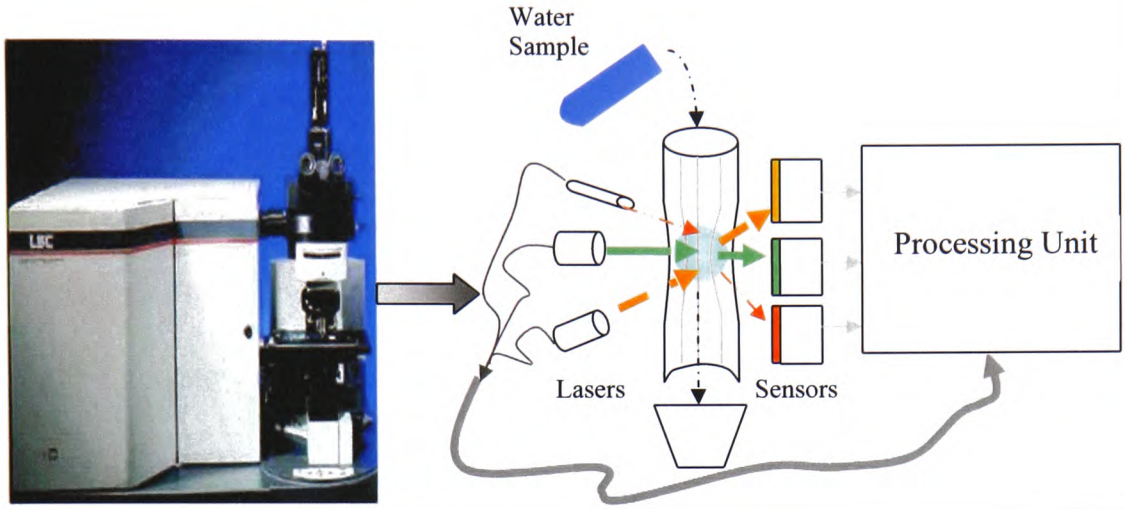
For all these reasons, phytoplanktons are under constant scrutiny. This chapter will describe the popular phytoplankton analysis techniques and their shortcomings. Then it will present the flow cytometry field, which is a research area created to speed up and automate the data analysis. Subsequently, it will describe some previous studies on neural networks in the phytoplankton analysis domain. Finally, it will state the aims of this project.

1.2 Classical phytoplankton analysis and flow cytometry

The classical sample analysis method requires a specialist in phytoplankton to sit in front of a microscope in order to count and identify every cell in the sample. Although this technique has been widely used, it limits sample analysis, as specialists are in short supply. Moreover, it is wasting the intellectual ability of this person on a repetitive and fastidious task. Furthermore, this "manual" analysis can take a long time, e.g. weeks, months. Finally, phytoplankton cells are difficult to identify thus when using an operator, the analysis is subject to human decision. The main problem with human decision is that different operators might make different choices thus the experiments are difficult to reproduce precisely. In order to overcome those issues, microbiologists from many different fields, e.g. medicine, biology, etc. identified the need for a tool that would measure automatically some of the cells parameters.

The Flow Cytometer (FC) was developed in order to fulfil this need. There are some major differences between the manual analysis and the FC analysis. In fact, the FC does not extract visual features during the analysis but optical properties of targeted cells. In order to extract this information, the FC hit the targeted cells using several laser beams (Figure 1.1). Those beams have signatures, i.e. different frequencies, colours, etc... The sensors inside the FC analyse the differences between the beams after and before they hit the target. These differences reveal information such as the size, density, chemical characteristics of the targeted cells. The FC and the manual analysis extract different cell features, thus the FC taxonomy and the manual taxonomy are sometimes different.

Figure 1.1: The drawing on the right illustrates the inside of a simple three beams flow cytometer. The particle flows through a narrow canal. Three lasers hit it, then the sensors extract the distortion information. That electrical information is then transformed into numerical values by the processing unit. On the left, the picture shows the EurOPA flow cytometer.



As the characteristics of different species of phytoplankton can vary dramatically, it requires a purpose-built FC. The first flow cytometers were only able to measure a couple of parameters thus it was sometimes difficult to differentiate the species. The modern FC such as the EurOPA FC (Figure 1.1) can analyse 10^3 cells s^{-1} while extracting seven parameters simultaneously, those parameters are:

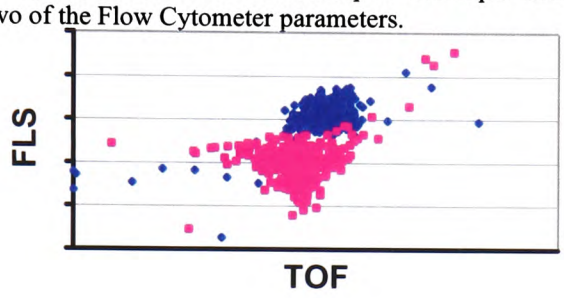
1. Time of Flight (TOF), TOF is the time it took a cell to pass in front of the sensor
2. Forward Light Scatter (FLS)
3. Perpendicular Light Scatter (PLS)
4. Red Fluorescence excited by a 488 nm laser (FBR)
5. Orange Fluorescence exited by a 488 nm laser (FBO)
6. Green Fluorescence exited by a 488 nm laser (FBG)
7. Red Fluorescence exited by a 630 nm laser (FRR)

The following chapter will describe different techniques used to identify the cells based on those parameters.

1.3 Analysis techniques and artificial neural networks

The most popular analysis technique is the scatter plot, using this technique the data are represented on a two dimensions diagram (Figure 1.2).

Figure 1.2: This diagram shows how the scatter plot technique differentiates two species using only two of the Flow Cytometer parameters.

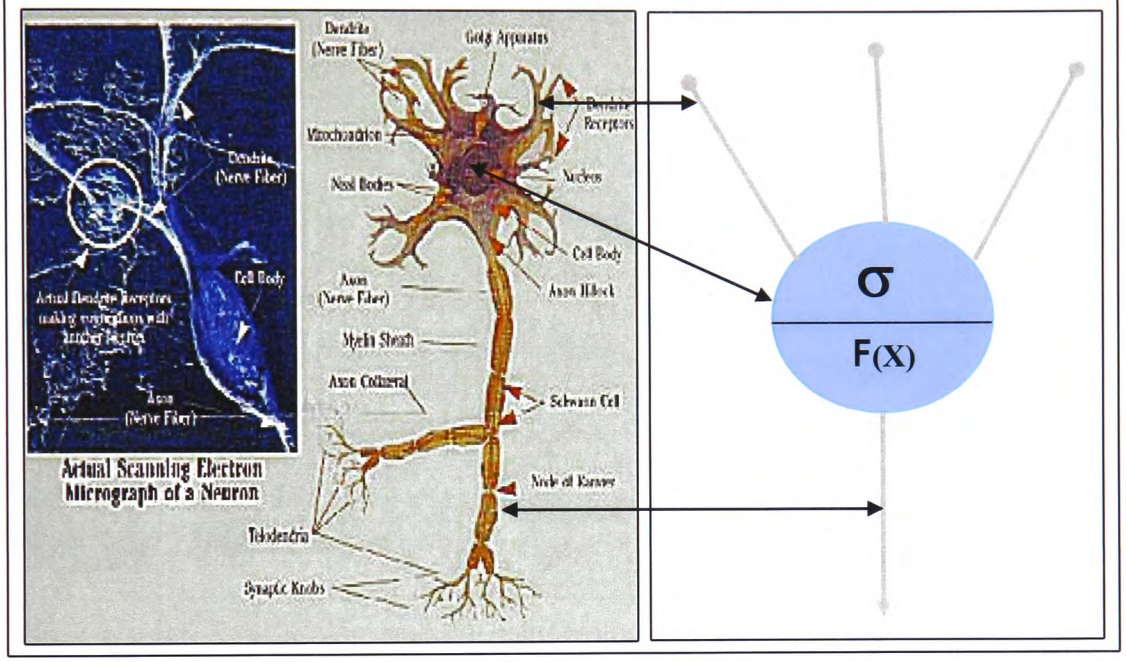


Although, Figure 1.2 shows how to use a scatter plots to differentiate two species, it is not always that simple. In fact, it is sometimes necessary to use several different parameters to identify the cells. As long as the parameter number is smaller than three, it is possible to represent the information on a single diagram. As the number of parameter is often greater e.g., it is seven with the EurOPA FC, representation becomes a problem. In order to overcome this limit, the parameters are displayed in groups of two. For example, a hypothetical study with four parameters would require 6 2D diagrams. The analysis of this ensemble of diagrams can be very difficult and often requires specialists.

Another problem with this type of graphical techniques is that the common species can hide the occurrence of a rare but crucial one, e.g. toxic species. Moreover, it is sometimes not possible to identify the species as no clear cluster can be represented. Finally, this method does not allow a precise count of the number of cells per species. There are many versions of this technique or similar techniques but they all suffer from the same problem. Another approach to analyse the data set is multi-variate statistics e.g. ANOVA, however those techniques requires an advanced knowledge of

statistics and of the data. Moreover, in order to apply those techniques and to reduce the complexity of the problem, it is often necessary to make some assumptions about the data. The Artificial Neural Networks (ANN) are an alternative to those classical techniques. The "Artificial Neural network" story has really started in 1943, with the first publication on the subject by the neurologist Warren McCulloch and the statistician Walter Pits [McCulloch, 1943].

Figure 1.3: Comparison natural (on the left) and artificial neuron:

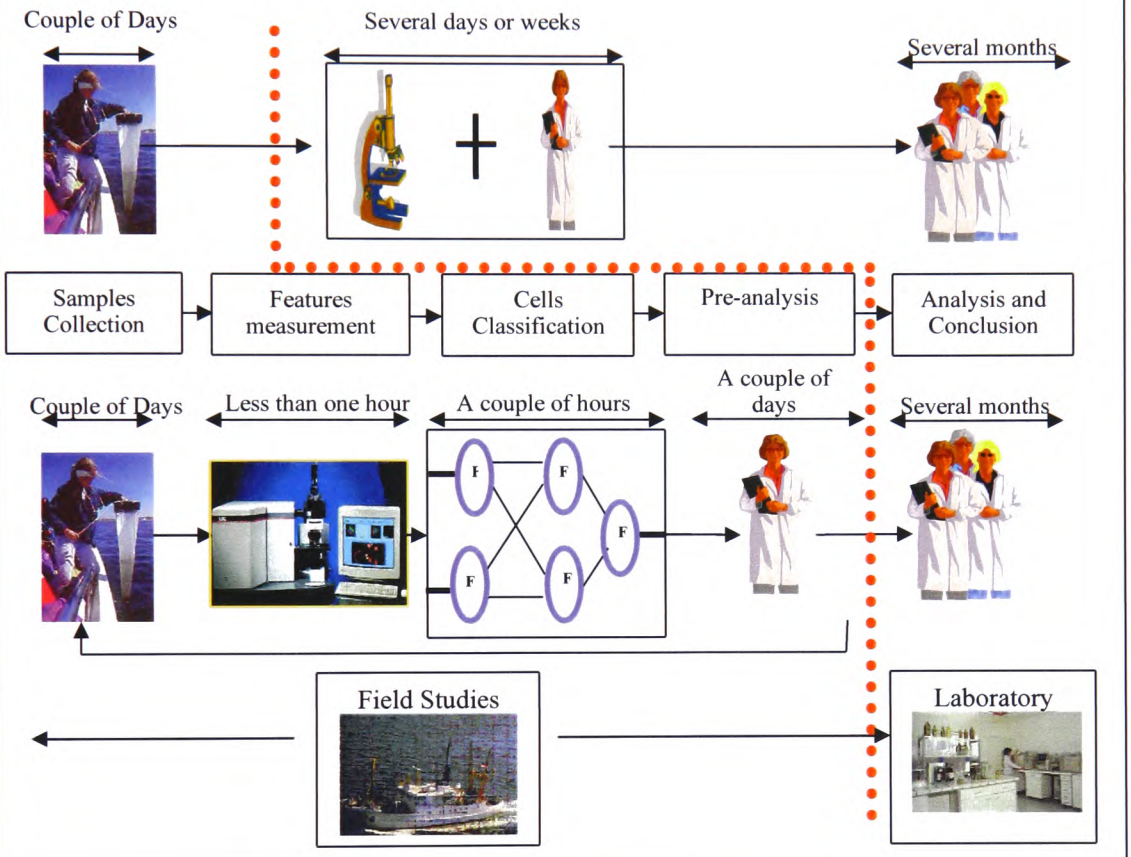


As shown in Figure 1.3, the original ANN attempted to model the brain using mathematics. However because of our limited knowledge of the human brain at a macroscopic level and the relatively small size of the ANN ($\approx 10^{14}$ neurons for the human brain against a maximum of 800 thousand neurons), the ANNs are quite far from the human brain. Despite those limitations, ANNs are able to learn some complex tasks by examples. Moreover, they are capable of a certain level of generalisation. Generalisation is the capacity of the neural network to extrapolate a solution for a problem based on experience.

The Backpropagation Network is the most popular ANN but several studies, e.g. Wilkins (1999) has demonstrated that the Radial Basis Function network was more appropriate for the phytoplankton analysis. The different application of ANN showed that they accurately identify the cells in a sample. Furthermore, as long as they were adequately trained, they are not sensitive to

the rarity of a given species in the sample. The ANNs are faster analysers (2800 cells/hour) than the specialists, and immune to fatigue or subjectivity. They also free the highly trained lab experts from the repetitive tasks Figure 1.4.

Figure 1.4: This diagram shows the difference of procedure between the classical (TOP) and the FLOW CYTOMETRY (Bottom) approaches.



Those specialists are then available to analyse the results of other experiments. Furthermore, as FC and Neural Networks analysis is much faster than the microscope one, it is possible to do some preliminary tests while on field studies; the preliminary tests can be useful to find if more samples are needed.

Despite these obvious advantages, the ANN technology is relatively new in phytoplankton analysis. There are several reasons for the relatively small number of application of ANN in phytoplankton analysis:

1. It is difficult to understand the decision process. Although all the neural network paradigms are mathematical models, it is difficult to understand the path to the decision as those models can be extremely complex.
2. On field experiment, it is often necessary to add/remove some species to the system. In order to do that, ANNs require retraining and the training requires some knowledge of ANNs. Moreover, the full training of the system is time consuming and does not preserve/use the previous knowledge of the system.
3. Phytoplankton study is an unbounded domain, meaning that it is difficult or impossible to predict all the species that will be contained in a sample. Those species can be the target species, other known species or even new undiscovered species. The last cases require that the system should be able to reject the unknown cells (cells from new or untargeted species).

1.4 Objectives of this project

Some previous studies have described a new kind of Modular Neural Networks (MNN) called Highly Partitioned Neural Networks (HPNN) as a possible solution to problem number 2 and 3 of the previous section. This project investigates in depth the use of HPNN and proposes a complete HPNN architecture that allows none ANN specialists to add and remove species to the identification system. It also attempted to create a system that rejects consistently cells from unknown species. Moreover, because of its particular architecture, HPNN made it possible to apply a new type of artificial neural network called Support Vector Machines to phytoplankton analysis. Thus, this project was the opportunity to compare the SVM and the more classical RBF paradigm.

Chapter 2: Literature survey

2.1 Introduction

The Flow Cytometer is a tool that extracts automatically and rapidly some optical properties of the phytoplankton (Chapter 1.2). The Flow Cytometer has mainly been used to limit the microscopic analysis but it generates a huge amount of information (10^3 per second). Thus, microbiologists often use some graphical techniques, chapter 1.3 describes their drawbacks.

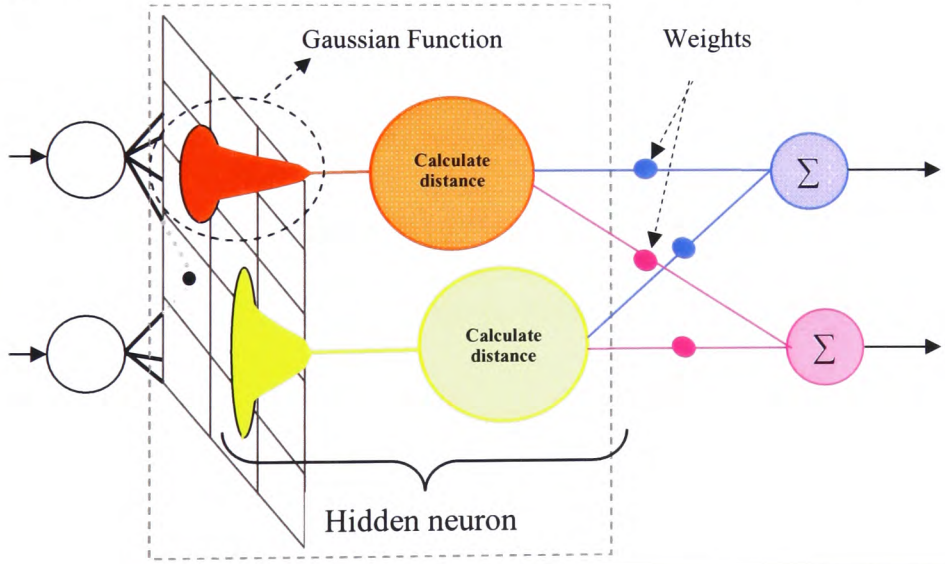
Artificial Neural Networks have been used as an alternative to these somewhat inaccurate methods. In fact, the ANN provides a way to precisely and consistently identify and count the cells in a sample. Moreover, they are a robust and reliable technique to analyse multi-variate data whereas graphical analysis makes it difficult to work with this kind of information. Although most of the studies [Frankel, 1989; Balfoort, 1992] in the literature involve the use of the classical backpropagation paradigm, some recent studies have shown that the Radial Basis Function network is a better choice [Boddy, 1994]. However, those experiments showed that the classical ANN paradigms lack the ability to scale-up without retraining or the capacity to reject unknown species (Chapter 3).

The Modular Neural Network (MNN) concept seemed appropriate for creating such system. In [Morris, 1998], an innovative type of MNN called the HPNN was introduced. This chapter presents two ANN paradigms called the radial basis function (RBF) and the support vector machine (SVM) (although, the full theory behind those paradigms is outside the scope of this thesis). Then, it gives an overview of the most common type of MNN including the HPNN. Finally, it lists the different approaches to novelty detection and rejection.

2.2 The radial basis function network

The Radial Basis Function network (RBF) is a partially connected neural network. It is made of three different layers of neurones, the neurones being the processing units. The input layer is made of simple $y=x$ neurones. The neurones from the middle layer are connected to all the neurones in the previous and next layer but not with each other. It is also called hidden layer is made of non-linear logistic functions, e.g. Gaussian Function (Figure 2.1). Finally, the output layer's neurones are made of a sum function. The output neurones' output is the distance between themselves and the input vector. With the RBF, learning means adjusting the position of the hidden neurones in the input space and of the weights' values (Figure 2.1). The weights are variables attached to the connection between the hidden neurones and the output ones.

Figure 2.1: RBF architecture. The input layer maps the data in the input space. The hidden neurons approximate the data distribution. They also estimate the distances between themselves and the patterns in the input space. The distances are then weighed. Finally, the output neurones sum the weighed distances. The network learns the problem by adjusting the weights



The RBF has proved that it is a powerful ANN paradigm; Hornik (1989) showed that it is **Universal Approximator**. A Universal Approximator is a paradigm that with the appropriate parameters can learn any ANN problem. There are many different kind of learning algorithms for the RBF network but they can be sorted in two distinct groups. The linear algorithms where the number of hidden nodes, size, shape and position stay fixed while

the weights are being adjusted and the non-linear algorithms [Orr, 1995]. The linear RBF learning strategy can be split in four stages:

1. Initialisation of the network parameters: number of input nodes, number of hidden and output neurones.
2. Placement of the hidden kernels to represent as well as possible the training data.
3. Find the hidden kernels width: Automatically or Manually.
4. Adjusting the weights in order to learn the problem based on the position of the hidden neurones.

Because after stage 2, the hidden neurones position does not change and because there is only one hidden layer, the output neurones calculate a linear combination of the distances and the weights. Adjusting the weights is simply solving a linear system. The solution is guaranteed to be the best possible given the current number of hidden nodes and their present positions. It is an advantage over non-linear techniques, e.g. Backpropagation or non-linear RBF, which require iterative procedure such as gradient descent to find the weights. However, it is possible to train the linear RBF weights with gradient descent algorithm [Dayhoff, 1990].

The classical RBF decision function is of the form:

$$F(x_i) = \sum_{j=0}^{m} w_j * h_j(x_i) \tag{1}$$

Where $h_j(x_i)$ represents the j^{th} kernel and w_j, the weights that link the kernels with the hidden layer. The solution vector W minimizes the Least-Square Error function:

$$S = \sum_{i=0}^{p} (y_i - f(x_i))^2 = \sum_{i=0}^{p} \left(y_i - 2 * f(x_i) * y_i + f(x_i)^2 \right) \tag{2}$$

over the training set $\tau \{ x_i, y_i \}_{i=0}^{p}$ that contains p couple of input vector x, output vector y. In order to find "w", it is necessary to differentiate the decision function (1) and the cost function (2) with respect to w.

$$\frac{f(x_i)}{\Delta w} = \sum_{j=0}^{m} h_j(x_i) \tag{3}$$

$$\frac{S}{\Delta w} = \sum_{i=0}^{p} (2 * f(x_i) - 2 * y_i) \left(\frac{f(x_i)}{\Delta w} \right) \tag{4}$$

Now (3) replaces $\frac{f(x_i)}{\Delta w}$ in (5).

$$\frac{s}{\Delta w} = \sum_{i=0}^{p}(2 * f(x_i) - 2 * y_i)(\sum_{j=0}^{m} h_j(x_i))$$
$$\frac{s}{\Delta w} = 2\sum_{i=0}^{p}\left(f(x_i) * \sum_{j=0}^{m} h_j(x_i)\right) - 2\sum_{i=0}^{p}\left(y_i * \sum_{j=0}^{m} h_j(x_i)\right) \quad (5)$$

In order to find the minimum error one has to solve $\frac{S}{\Delta w} = 0$.

$$2\sum_{i=0}^{p}\left(f(x_i) * \sum_{j=0}^{m} h_j(x_i)\right) - 2\sum_{i=0}^{p}\left(y_i * \sum_{j=0}^{m} h_j(x_i)\right) = 0$$
$$\sum_{j=0}^{m}\left[\sum_{i=0}^{p}\left(f(x_i) * h_j(x_i)\right) - \sum_{i=0}^{p}\left(y_i * h_j(x_i)\right)\right] = 0 \quad (6)$$

Let us rewrite (6) and (1) with vectors.

$$f.h_j^T - y.h_j^T \Rightarrow f.h_j^T = y.h_j^T \quad (7)$$

$$f = w.h_i^T \quad (8)$$

If f is replace by (9) in (7), it gives (9).

$$w.h_i^T.h_j^T = y.h_j^T \quad (9)$$

The equations (7 and 8) use two different types of activation, one uses the columns and the other the rows. h_j^T represents the output of every hidden neurones for one pattern and h_i^T, the output of one kernel for every patterns, thus it is easy to merge these two vectors in a matrix, H.

$$H = \begin{matrix} h_1(x_1) & h_2(x_1) & \Lambda & h_m(x_1) \\ h_1(x_2) & \text{O} & & \text{M} \\ \text{M} & & \text{O} & \text{M} \\ h_1(x_p) & \Lambda & \Lambda & h_m(x_p) \end{matrix}$$

$$w.H.H^T = y.H^T \quad (10)$$

From (10), it is now straight forward, after a small transformation to solve (10) using matrix operation.

$$w = y.H^T.(H.H^T)^{-1} \quad (11)$$

There is however a problem. In fact, finding the inverse matrix is not always possible and even when it is possible, it can be time consuming. There are many different techniques in linear algebra that are used in order to overcome

or bypass these issues but it is out of the scope of this thesis. Even when there is no problem with the matrix resolution, the classical RBF performance is limited by the number and characteristics of the hidden kernel. The simplest approach requires the user to define the number of kernels and their width then the network positions them using some clustering algorithms. The most popular clustering algorithms are the K-mean and Learning Vector Quantization (LVQ).

The K mean clustering algorithm aims at finding the best position for k pre-defined vectors to fit a set of data. First, the k vectors are randomly set in the vector space; those randomly set vectors are called "seeds". When each pattern is assigned to the nearest seed, the seed's position is re-calculated with respect to the newly added components; the resulting vector is now called "centroid". The process of adding components and re-computing the position is repeated until all the components are included in one of the centroid.

A simple algorithm can summarise the k-means technique.

1) Set the k initial centroid seeds.
2) Assign the closest component to each seed. Calculate its new mean. If one centroid becomes closer to a component than the centroid in which the component was previously added, then add the component to the closest centroid, remove it from the other, and re-compute the mean.
3) Repeat until no changes are needed.

For further detail see [K mean].

As for the k-mean method, the aim of the LVQ, when applies to RBF, is to find the best representation of the training set using the n kernel. Again as for the k-mean algorithm, the LVQ updates the kernel position with regard to the input vector however the updating method is different. When using LVQ, the kernels are called codebook vectors and the set of codebook vector is one codebook [Kurimo, 1991]. The codebook vectors are updated with regard to one randomly chosen vector x from the training set.

There are several LVQ algorithms; the difference is linked to the learning rules.

LVQ1

If the closest codebook vector v_c is of the same class as the input vector x

$$v_c(t+1) = v_c(t) + \alpha(t)[x(t) - v_c(t)] \quad (13)$$

If the closest codebook vector v_c is of a different class as the input vector x

$$v_c(t+1) = v_c(t) - \alpha(t)[x(t) - v_c(t)] \quad (14)$$

The teaching factor $\alpha(t)$ decreases monotonically with the time.

LVQ2

Same as LVQ1 but update the nearest vector and the second nearest if the sample appears to be near the border between two classes. LVQ2 exist in two releases 2.0 that decrease the distance between the closest vector and x but at the same time does the opposite with the second closest vector. LVD2.1 does the symmetric opposite of LVQ2.0.

One vector is close to one border if the following equation is not true

⇨ $d_c / d_c' > (1-w)/(1+w)$ where $d_c = \| v_c - x \|$ and $d_c' = \| v_c' - x \|$ and $w \in (0,1)$

Other LVQ variants are available but are inspired by these two.

The clustering algorithms presented above try to position the kernels in the best possible way. However once the kernels are placed, the system might still generalise poorly as the user might have selected too many kernels. In order to lessen this problem, one can use a method such as Ridge Regression. In Ridge regression a supplementary term is added to the Least Square Error function S (2), it now gives the Cost function *C*.

$$C = \sum_{i=0}^{p} \left(y_i - f(x_i)^2\right) + \lambda \sum_{j=0}^{m} w_j^{\,2} \qquad (15)$$

"λ" is the regulation factor. The effect of Ridge regression is to nullify the effect of some of the kernels and thus decreasing the size of the network. The λ factor has to be defined using some non-linear optimisation techniques or by testing. The methods presented above are the classical tools used to select the size and shape of the RBF, however there are other techniques. Berougjit (2002) developed a method that attempts to optimise the width of the kernels once they have been positioned. Other studies use Evolutionary programming [Topchy, 1997]. Mark Orr (2000) constructs a decision tree in order to analyse the data set. Once the tree has been completed, its analysis will dictate the width, shape, number, position and size of the hidden kernels. Then using a classical linear resolution, the algorithm finds the weights' values. Aupetit (2000) uses a modified version of the Neural Gas techniques in order to position the kernels and find their size and position. This technique concentrates the hidden kernels in the space where the function is the most complex. Some hidden kernels are also added during the training in the place where the error is the biggest. Finally, Orr (1995) presented a heavily modified RBF paradigm in which a large pool of hidden neurones is positioned in the space using a classical clustering technique. Then, gradually and as long as the system has not properly learnt the problem, those neurones are added one by one with respect to their significance. The difference between the last paradigm and the ones presented before is that in this case, the kernel width is fixed. The network size evolves while the algorithm tries to learn the problem by adjusting the weight values. In order to progressively add the hidden neurones, Orr modified the linear resolution of stage 3. The modified algorithm only requires calculating the new weights and error of the system which is much more efficient than recalculating all the weights every time. Finally, to avoid overtraining, it also includes ridge regression.

2.3 *Support vector machines*

2.3.1 Introduction

In 1995, Vladimir N. Vapnik introduces a new learning algorithm call "The Support Vector Machines". This technique, based on a solid mathematical foundation, aimed at automatically finding the size of the network in order to optimize its learning and generalisation capabilities.

2.3.2 Roots and theory

As with most neural network paradigms, the SV machine (Support Vector Machine) tries to find a mathematical function that produces the smallest error as possible on a given set of examples, the so-called training set. It uses the following function to estimate the error.

$$C = \sum^{n} \left(y - f(x)\right)^2 \tag{16}$$

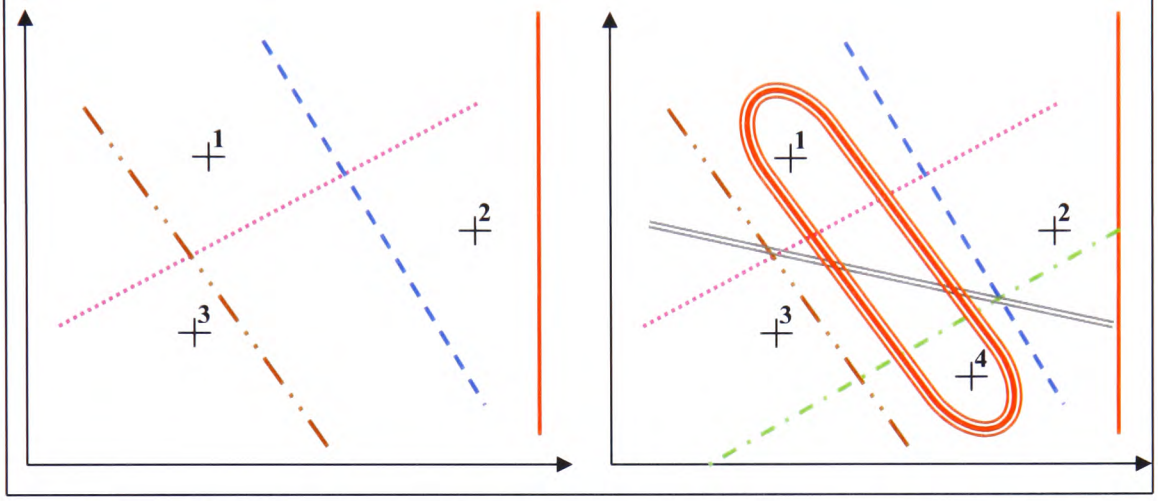
This function estimates the difference between the target answer **y** and the network's one **f (x)** on a set of examples of size **n**, **x** being the network input. It is often called the cost function. In his theory, Vapnik called the result of this equation the **Empirical Risk**. It is the opposite of the **Real Risk**, which is the real error that could be calculated using all the possible inputs. Most neural network techniques assume that as the training set grows the empirical risk and the real risk converge. However, after the training, most techniques cannot guarantee a good generalisation or even estimate that the network has the smallest possible size. The SVM network implements a new approach to learning by which it tries to minimise the error on the training set while choosing the size of the network to obtain a good generalisation.

$$\textbf{Risk}_{\textbf{real}} < \textbf{Risk}_{\textbf{empirical}} + \textbf{Confidence Interval} \tag{17}$$

The Confidence Interval represents an upper bound estimate of the difference between the empirical risk and the real risk; it is called the VC confidence. The VC confidence factor is itself based on an estimate of another value called the VC (Vapnik – Chervonenkis) Dimension. It is an estimate of

the network's learning ability regardless of the free parameters. It is defined as the number of points that can be "shattered" by the network, i.e. one model has dimension 4 if it can achieve all the possible dichotomies of any 4 points (Figure 2.2).

Figure 2.2: These graphs show that a line has a VC dimension of three because it can not obtain all dichotomies of 4 points, i.e. point 1 and 4. It is enough to show that the model can not work with one set of randomly selected points to limit its dimension.



The VC Confidence is an attempt to guarantee good generalisation once the training end; its opposite, the Confidence Interval was introduced in (17).

$$R_{real} \leq R_{emp} + \sqrt{\left(\frac{h\left(\log\left(\frac{2l}{h} \right) + 1 \right) - \log\left(\frac{\eta}{4} \right)}{l} \right)} \qquad (18)$$

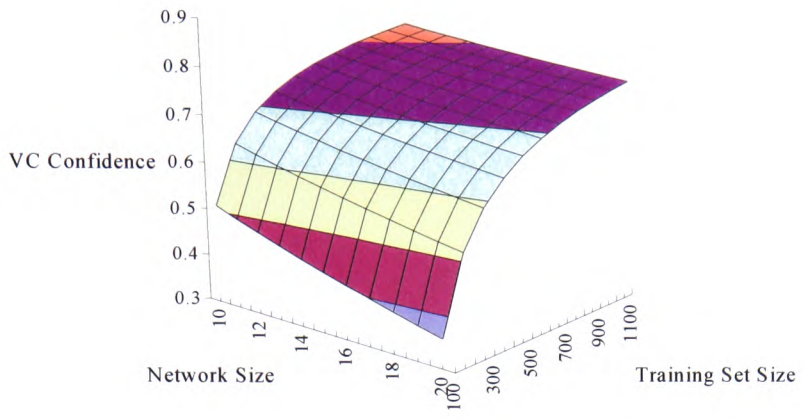
h is the VC dimension,

l is the ANN size,

and η is a constant whose value has to be >0 and <1 for (18) to be true.

As shown in (18) the use of the Confidence Interval (VC Confidence = 1- Confidence Interval) gives a guaranteed upper bound on the real risk; Figure 2.3 shows how the VC Confidence changes as a function of the changes in the network dimension or in the training set size.

Figure 2.3: Evolution of the VC confidence with respect to the network and training



When the confidence interval term is added to the real risk value, this value does not only represent the estimation of the training error but also of the generalisation ability. It is well known that as the network size increases, its learning ability increases but its generalisation aptitude degrades. The VC Confidence Interval is a numerical estimate of the generalisation with respect to the training set/network size ratio. Thus during training, the network will attempt to find the best trade-off between the network size and the VC Confidence to obtain the lowest Risk possible. It has to be compared to the more classical approaches that only minimise the training error and thus often over-fit the training set, giving poor generalisation. Although the previous concepts were theoretically sound [Vapnik, 1995], it was proved in several experiments [Burges et al., 1998] that Support Vectors Machines tend not to respect those theories. However, the theoretical work sets the basis for further works using different networks or on the improvement of the SVM [Cortes et al., 1995]. It also highlights the concept behind SVM.

2.3.3 Implementation

The Support Vector Machines are based on prior studies from Vapnik and Chervonenkis on a specific type of hyperplane which they called the optimal hyperplane. Let us assume that the network is presented with a training set S that contains n pattern (x, y), being either from class –1 or class +1. In a more formal sense, it gives:

$$S=\{(x_1, y_1)\ldots(x_n, y_n)), x \varepsilon R^n, y \varepsilon \{+1, -1\}$$

The separating hyperplane has the following form

$$(w.x_i)+b \geq 1 \qquad \text{If } y = 1 \qquad (19)$$

$$(w.x_i)+b \leq -1 \qquad \text{If } y = -1 \qquad (20)$$

For convenience both equations are compacted in the following one

$$y_i[(w.x_i)+b] \geq 1 \qquad (21)$$

The specificity of the optimal hyperplane is that it tries to maximise the distance between the two classes. From the previous equations, it is possible to see that distance between points that satisfy the two constraints $(w.x_i)+b = -/+1$ (19 and 20) will be the smallest (Figure 2.4).

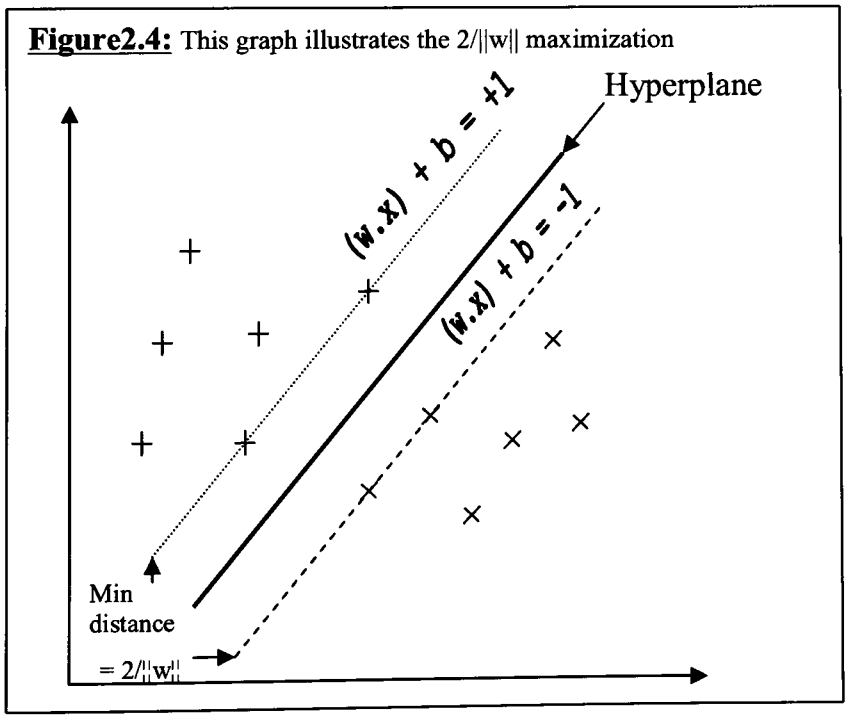


Figure2.4: This graph illustrates the 2/||w|| maximization

This network tries to find a hyperplane than maximises the distance between the two classes given in the examples. It has often been cited as the main factor

in the good generalisation of the SV machines. It can be proved that to maximise the margin between the two classes, one has to minimise ||w|| and that the maximum margin will be 2/||w|| (Annexe 9.2.1). The problem is now set as this algorithm wants to find the support vectors that verify (21) but also that maximise the margin by minimising ||w||, which can be expressed as $\mathbf{w^t w}$. One of the most common technique to find the optimum function given one set of data is call "Dual formulation technique" [Gill, 1991]. This technique is split in two stages. The primal problem tries to find the optimum values for the set of variables, in this case **b** and **w** that satisfies some constraints. In this problem the constraints are that the values should verify (21) and minimise $\mathbf{w^t w}$. First, the constraints are combined in a Lagrange format (22).

$$L_P = \frac{1}{2} w^t w - \sum_{i=1}^{n} \lambda_i \left[y_i \left(w^t x_i + b \right) - 1 \right] \tag{22}$$

This equation introduces λ that represent the "Lagrangian multiplier". This equation has now to achieve two distinct targets; first, it must minimise ***w*** and ***b,*** second it must maximise λ. Following the rules of mathematics to find the extremum, with a multi-variable equation, one has to differentiate (22) with respect to **w** and **b** independently.

$$\mathrm{w} = \sum_{i=1}^{N} \lambda_i y_i x_i \tag{23}$$

$$\sum_{i=1}^{N} \lambda_i y = 0 \tag{24}$$

The problem is convex[1], linear and only the points that respect (24) are different from zero, the Kuhn-Tuckers conditions are satisfied thus **w**, **b** and λ are guaranteed to be a solution to the problem. Because of the convexity and linearity of the function, the optimal solution of the primal problem will be the same as the one for the dual problem. In order to obtain the dual problem, it is necessary to expand the primal problem as followed.

$$L_P = \frac{1}{2} w^t w - \sum_{i=1}^{N} \lambda_i y_i w^t x_i - b \sum_{i=1}^{N} \lambda_i y_i + \sum_{i=1}^{N} \lambda_i \tag{25}$$

By using (23) and (24) (Annexe 9.2.2), it is possible to obtain

[1] Function whose value at the midpoint of every interval in its domain does not exceed the average of its values at the ends of the interval.

$$L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i x_j \tag{26}$$

This equation is subject to the constraints coming from the Kuhn-Tucker Theorem

$$\sum_{i=1}^{N} \lambda_i y_i = 0 \tag{27}$$

$$\lambda_i \geq 0, i \in \{1, N\} \tag{28}$$

Once the Lagrangian multipliers λ have been found it is easy to obtain the other variables by replacing λ by its values in (23) and (19) or (20). The solution to the problem will be found by solving either (22) or (26).

Up to this point, the document only spoke about the separable case. In the nonseparable case, parts of the two classes are overlapping or some of the points fall out of the general pattern (Figure 2.5).

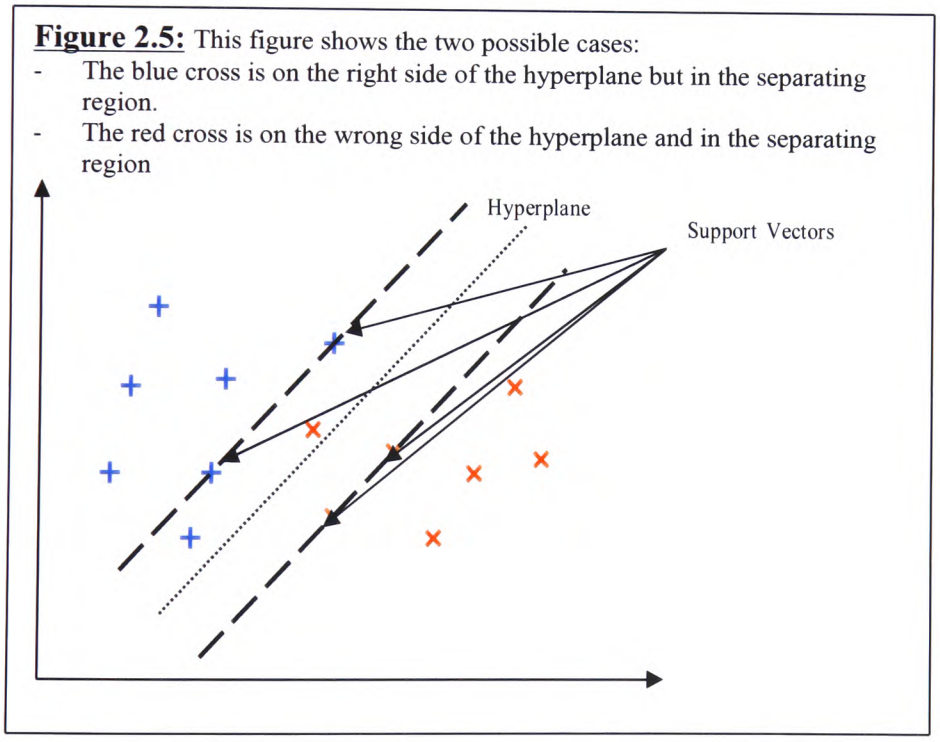


Figure 2.5: This figure shows the two possible cases:
- The blue cross is on the right side of the hyperplane but in the separating region.
- The red cross is on the wrong side of the hyperplane and in the separating region

To treat these problems, one supplementary variable ξ has been introduced in (21), which now became (29).

$$y_i[(w.x_i)+b] \geq 1 - \xi_i \tag{29}$$

The new variable is use to measure the deviation of the data points with respect to the separating plane. In Figure 2.4, the blue crosses are on the right side but

inside the separating zone then the slack variable will have a value between 0 and 1. If as in the red cross case, the data is not nicely behaved, that is the data is not separable, then the value will be greater than 1.

Then the net will have to minimise

$$\frac{1}{2}w^t w + C\sum_{i=1}^{N}\xi_i$$

C controls the trade off between the complexity of the hyperplane and the number of nonseparable points. Using the same method as in the separable case, it is then possible to express the problem in the dual formulation form. It is however interesting to notice that the dual problem will have the same format, except for the constraints (30,31).

$$\sum_{i=1}^{N}\lambda_i y_i = 0 \tag{30}$$

$$0 \le \lambda_i \le C, i \in \{1, N\} \tag{31}$$

As it was stated earlier many times, the SVM is a linear system, thus in its standard format it can only solve linear problem, linear in the feature space. In a non-linear problem, the separating hyperplane will have the following form:

$$\sum_{j=1}^{n} w_j \varphi_j(x) + b \tag{32}$$

The letter φ is a vector of non-linear transformation of length 1 and w, a linear vector that link the feature space to the input space. After applying the same techniques as were used in the linear problem to this case, (33) is obtained (Annexe 9.2.3).

$$\sum_{i=1}^{m}\lambda_i y_i \varphi^T(x_i)\varphi(x) = 0 \tag{33}$$

(33) shows a product of the non-linear transformation vector by its transpose. From the previous equation it is easy to see that the computation time required by the dot product can be enormous, as the feature space dimension can be large. However, Mercer's Theorem (Annexe 9.2.4) states that a so-called Mercer kernel K (x_i, x_j) can replace the inner product in (33). Thus, it is not necessary to consider the feature space any more and the inner product of dimension m does not need to be computed.

$$\sum_{i=1}^{m} \lambda_i y_i K(x_i x_j) = 0 \quad (34)$$

(34) is the resulting equation once the inner product is replaced by one of Mercer's Kernels (Annexe 9.2.4). Using this theory, it is possible to see that to transform a linear SVM into a non-linear one, it is enough to replace the inner product of non-linear transformation in SVM by Neural Network kernel as long as it respects Mercer's theorem.

2.4 Modular neural network

Despite the increasing capacity of today's ANN paradigms, they still have limitations in learning ability, robustness and interpretability. Even the classical ANN applications such as pattern recognition can challenge the most powerful paradigms. For example, pattern recognition can become difficult if the classes significantly overlap a lot or if the input vector has a very high dimension. Moreover, as the complexity of the problem increases, so does the size of the ANN. As the size increases, it becomes rapidly more difficult to assess the robustness of the trained network or to interpret the decision pathways. Furthermore, the training time may increase exponentially with the size of the network. In order to overcome those issues, some scientists suggested that a group of "smaller" ANN would be better than a unique ANN. Two types of network group appeared:

1. ensemble
2. modular

This thesis mainly focuses on the modular approach however it is interesting to briefly speak about the most popular ensemble techniques as they are related to the Modular Neural Network (M.N.N.).

The ensemble theory is that a group of experts will produce a better system than a single one. Bagging is a popular ensemble architecture. A bagging system is made of a group of ANN also called experts or learners. Those experts are trained with the same training set. The experts can be of the same type or of different types but with a different initial state [Opitz, 1999].

Once the experts have been adequately trained, two different kinds of combination strategies can be used:

1. Hard switching, the system choose one active expert as the "winner"
2. Soft switching combines the outputs of the different experts for every pattern.

While using an ANN, the paradigm assumes that the training data are representative of the problem domain, e.g. phytoplankton analysis. A pattern represents the combination of different information on a single entity. It also presumes that the data distribution in the training set is representative of the "real" distribution. Breiman (1994) showed that training multi-class unstable predictors, e.g. Neural Network or Decision Trees, on the subsets of a given training set and then combining them, will produce better result than using a single multi-class expert trained on the full training set. This technique is called bagging. Breiman (1994) implements a hard switching technique called voting. In voting, the class with the highest number of positive identifications is elected as the winner. On the other hand, Hashem (1997) uses a soft switching approach in which, the output of the different experts is linearly combined (35).

$$f(x) = \sum_{j=1}^{p} \alpha_j y_j(x) \tag{35}$$

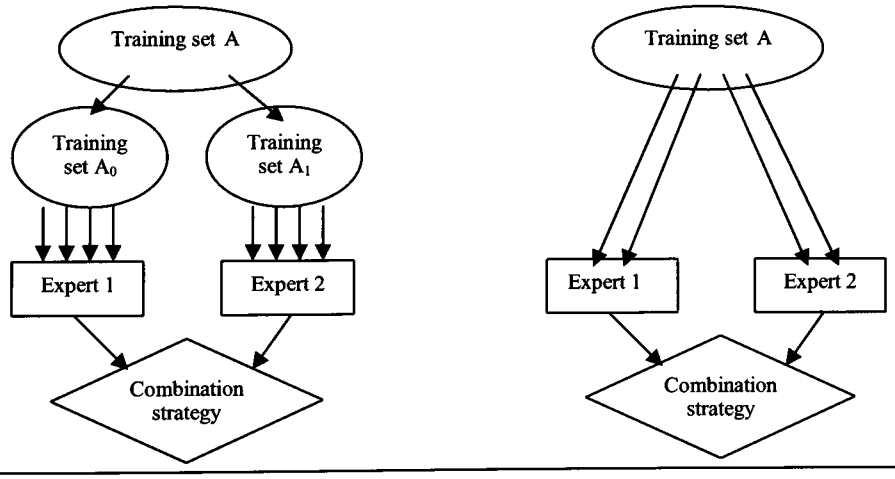
f(x) is the linear combination of the weight vector α and of the neural network's output vector *y(x)*. Those two vectors have p dimension. For every pattern x, every expert y_j returns a value. During training the system learns how to combine the different experts' output by adjusting the α. Hansen(2000) also uses a combination strategy however it not only combines the experts as [Hashen, 1997] but it also takes into account the similarity of the experts in order to improve the system performance. As the bagging performance is directly linked to the correlation of the performance of the pool of experts, a pool in which the experts produce their errors on different patterns will perform better than one in which the miss-identified patterns are the same. Another popular ensemble technique is called boosting. In order to apply boosting, the training set is divided into smaller files. However, this division is not random as for bagging. The n experts are trained sequentially, the first expert trains with the original set. In order to create the training set for the next expert, the

patterns, which were less well identified, will be more likely to be selected [Opitz, 1999].

There is not a clear winner while comparing "Bagging" and "Boosting". Optiz (1999) compared different bagging and boosting techniques and concluded that bagging is a cheap way to improve the system performance. He also suggests that boosting could bring even more performance gain but that the selection is crucial, e.g. a badly built boosting system does not perform as well as a monolithic one whereas good ones outperform bagging.

Although the ensemble methods attempt to improve the identification/regression performance of the ANN systems, they focus on improving the quality of the training sets. Several experts train on sub-sets of the global training set, each the sub sets being a full description of the problem, the experts are then combined in the hope that the group will achieve better performances than any single expert. The modular approach on the other hand, tries to "simplify" the problem itself in order to achieve better results; (Figure 2.6) illustrates the difference between the two approaches.

Figure 2.6: On the left, a typical ensemble architecture and on the right, a possible modular technique in which the input vector is broken in two thus the experts are trained on two sub-tasks.



As shown on (Figure 2.6), a Modular Neural Network (MNN) system fragments the problem/task into sub-tasks; each expert in a MNN does not learn the global problem but a small part of it. Thus, the expert would not be able to "solve" the problem that is the main difference with the ensemble strategies.

There are two main philosophies while breaking down a problem:

1. Horizontal decomposition
2. Vertical decomposition

The horizontal decomposition strategy breaks down the problem in the input space, meaning that the input space is divided into zone/sub-spaces. There are many techniques to divide the space however most of the time those techniques aim at reducing the mathematical complexity of the learning task by localising learning (Figure 2.9 illustrates local learning in a 2D space). In vertical decomposition, the system or the system designer breaks the problem in order to reduce the complexity of the problem in the output space. A vertical decomposition is often decided by a specialist in the field of application of the neural technique, .e.g. In Morris (1994), some species were grouped as a way to increase the identification confidence. The neural network was then used to identify the new group. Although, this was not a MNN application, the task would have been suitable for vertical decomposition. This explicit decomposition was only possible because the system designer knew that it was better to have a less precise identification than a mis-identification. Once the problem has been broken down and the experts trained, they need to be combined in order to fulfil the requirement of the global problem.

As for the ensemble techniques, there are two types of combination strategies that can themselves be implemented into many different ways:

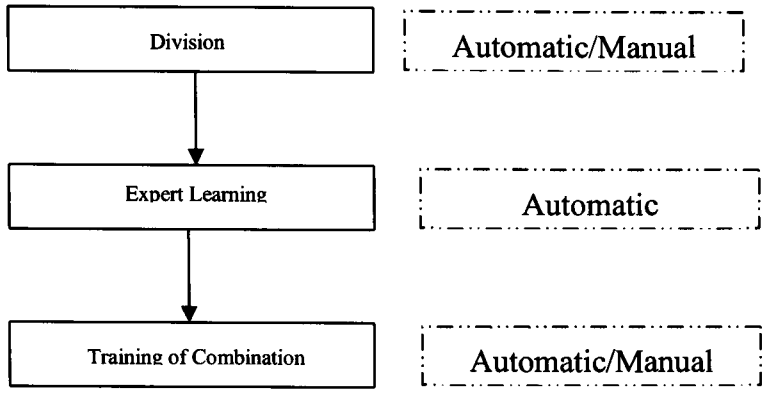
- Hard Switching, it can be applied to cases 1 and 3 of the list on page 32
- Soft switching that can be applied to any division

The combination of switching techniques and division schemes can then be grouped into two categories:

- Automatic, the system learns the division and combination strategies
- Manual, the system uses the prior knowledge of the user

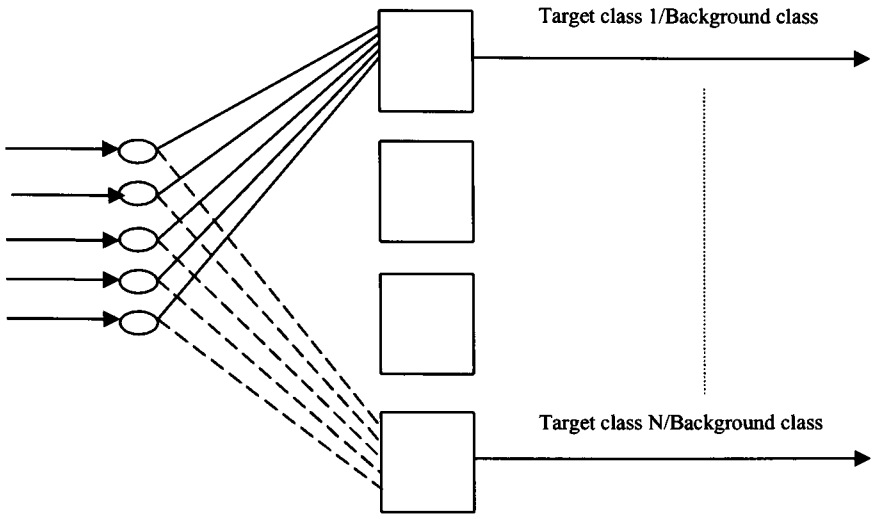
Therefore MNN training can be split into three stages (Figure 2.7)

Figure 2.7: Diagram of the MNN training process



As briefly presented earlier, a vertical division fragments the global problem into sub-tasks based on the problem domain. The most common type of vertical decomposition splits the output space into several sub-space, i.e. a *K* class problem into *N*2* class problems. The aim of the division is to improve the system performance. Masulli (2001) breaks the *K* classes' problem into *L* two categories problems. Every category can be made of many classes. Then the grouped signature of the sub-tasks experts is analysed and a unique "codebook" vector is associated with every class. It presents a method called the Error Correcting Output Codes (ECOC). In order to maximise the performance of the system, the codebook has to be uncorrelated. There are several methods to measure the correlation, i.e. the Hamming distance. The tests show that ECOC can be used to improve the performance of the system and correct learning error. In Lu (1998), a *K* dimensions problem is divided into *K*K* sub-tasks. The system uses three operators (Inv, Min & Max) using a combination of these operators, the size of the pool of experts is limited to C_2^K, i.e. with 4 classes, the pool is made of six experts. Using a different approach, Anand (1995) uses a technique that splits a *K* class problem into *K*2* classes problems. For every two classes problem, the expert is trained to differentiate a "*Target*" class from a "*Background*" class made of *K-1* classes. The output vector is a *K* dimensional vector (Figure 2.8).

Figure 2.8: Each expert differentiates a target class from a background class made of all the other class. The output of every expert is either positive for target class or negative for background class



The architecture in (Figure 2.8) should be able to reject a novel class. In fact, if all the outputs are identifying a background class then the pattern is probably from a novel class. Although that is a powerful ability, this simple architecture would not work adequately if several experts were to identify the same pattern as their target class. The previous studies vertically divided the problem, vertical division is an explicit division of the problem dictated by the system designer in order to fit a specific target, e.g Massulli (2000) splits the output space to improve the performance, as does Lu (1998). Horizontal division also aims at improving the performance of the system by dividing the problem into sub-tasks. However, the division aims at reducing the mathematical complexity of the tasks (Figure 2.9). Finally, as with the ensemble technique and with the vertical decomposition schemes, the sub-task learners have to be combined.

Figure 2.9: The red line is an approximation to the non-linear equation **f(x)** (**black line**) using several linear models. Every rectangle represents the area of influence of a linear model. This drawing would correspond to a Hard Switching combination strategy, as every model is only active in one single area.

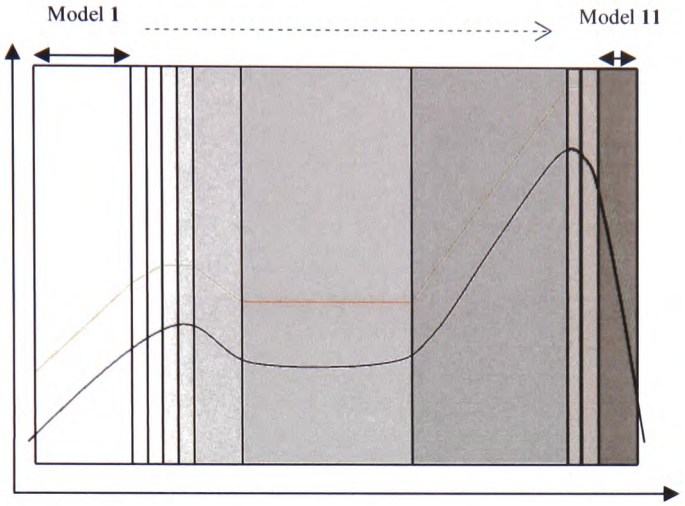
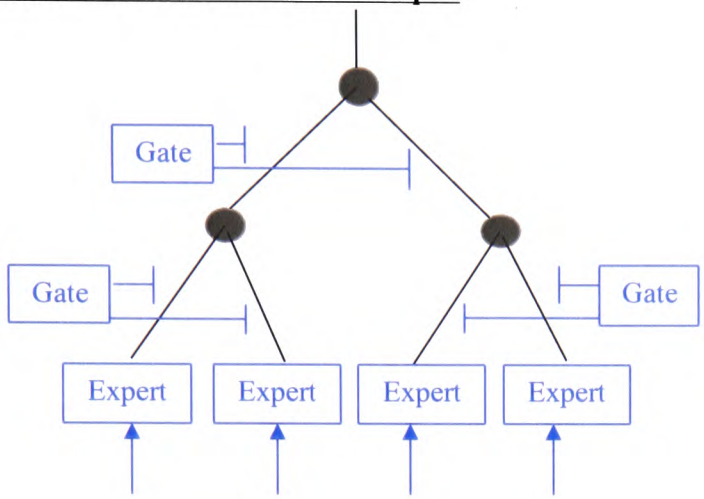


Figure 2.9 shows a horizontal decomposition for a hypothetical two class problem. The target separating hyperplane is non-linear. However, using local learning, it can be approximated if enough linear models are used. A linear model is only active in one part of the feature space thus it is an example of hard switching. In order to improve performance, Sun (1999) suggests that the problem has to be partitioned in the input space. A different ANN learns every partition and the output of those models are then combined. Sun (1999) reviews several strategies used to partition the input space. First, the weighted linear combination of the experts output using for example equation (35) and is adjusted in order to minimize the error function (36).

$$E^c = -\log \sum_i p_i^c e^{-abs(d^c - o_i^c)^2 / 2\sigma^2} \quad \text{[Jacobs, 1991]} \qquad (36)$$

E^c is the error on pattern ^{c}, p_i^c is the output of the gating network for expert I, d^c is the desired output, o_i is the output of the vector i and σ is a constant. This error function generates a hard switching of the system between the experts.

Figure 2.10: HME – Hierarchical Mixture of Expert



As written by Sun (1999), the boosting ensemble method can itself be seen as a soft partitioning method. In Jordan (1994), a new type of Modular Neural Network architecture was presented; it is call the Hierarchical Mixture Experts (Figure 2.10). The HME is an extension of the ME from Jacobs (1991). When the ME had only one level of division the HME allows multiple concatenated divisions (Figure 2.10). This fine division of the input space allows the HME architecture to use simple linear models as experts. Although the HME fragments the input space, it is not implicitly spatial clustering. The association of the model takes into account the ability of the expert to reply correctly to a pattern [Ronco, 1998]. In order to achieve this type of clustering, the HME algorithm submits the new pattern to the experts. If an expert replies incorrectly, the gates learn to minimise the role of this expert for this type of patterns. The experts also adjust its weights in order to improve its performance. It was observed that when this system converges to a stable solution, it only uses one expert for one type of patterns. However, the gates do not spatially cluster the data but learn the capacity of the experts to give the correct output for different types of patterns. Chen (1999) produced an architecture that combined the Ensemble and MNN techniques into a GME (General Mixture of Experts) or GHME (General Hierarchical of Experts). Those architectures are similar to the HME. However, every HME experts is now represented by a bank of experts. The same technique is applied to the

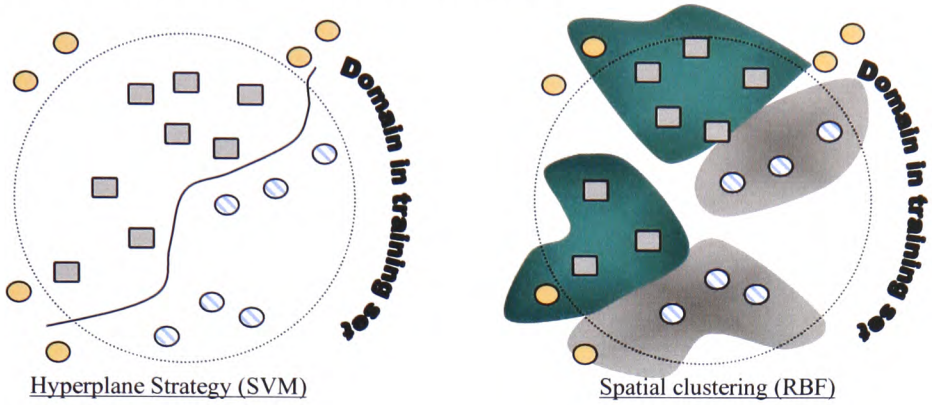
gates. As for Hashem (1997), the experts or gates output are then combined linearly.

Although, there are many other techniques, the ones presented here are the most popular. The other approaches tend to be customised to fit a particular problem.

2.5 Novelty detection

Although, ANNs have been successfully applied in a variety of domains, there are experimentation fields that are not suitable for most paradigms. The principal weakness of the common paradigms in those particular domains is their inability to efficiently and consistently deal with unknown (novel) patterns. If one considers a hypothetical insect study in the middle of the Amazonian forest, this study aims at counting the number of insects of every species in the study in a limited zone. The main problem with this particular problem is that there are many unknown types of insect. Most ANN will put those novel species into the known classes even if they were not known species. However, the scientists would probably be more interested in a system that would detect the novel patterns and store them for analysis. This would allow a further manual examination of those novel patterns. Even though the previous example is specific, it can be applied to many different domains: in the military domain a mis-identification of an unknown radar picture could lead to dramatic events, in the medical domains, unknown patterns could be a new pathology. This weakness of many ANN paradigms comes from their learning strategy. In fact as explained in Chapters 2.1 and 2.2, the NNs learn by approximating a function that splits the training set in sub-regions (Figure 2.11).

Figure 2.11: The problem of ANN is that during the training, the ANN finds separating function(s) for the given domain (the dashed ellipse). However if a pattern lies outside the domain, the behaviour is not guaranteed.

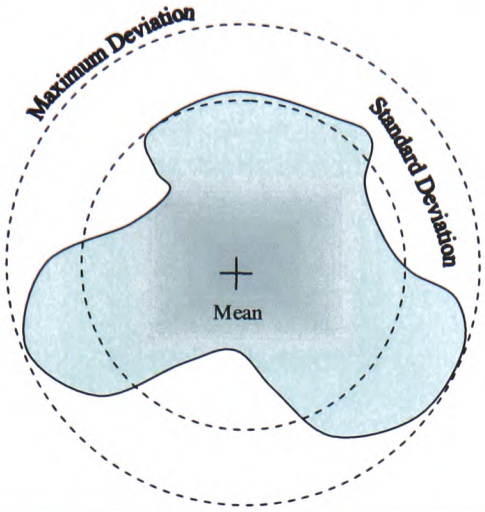


Hyperplane Strategy (SVM) Spatial clustering (RBF)

As illustrated in Figure 2.11, the identification function(s) is/are only relevant inside the domain approximated on the training set. The problem domain is a sub-space inside the infinite space which contains the training data. Some paradigms such as the multi-layer perceptron or the SVM approximate a function; it separates the different classes inside the problem space. When the function is linear, it is called a plane in 3D space, a hyperplane in N dimensional. If the function is not linear, the function is called a curve or hypercurve. For notation convenience, the separating surface will always be called a hyperplane in the rest of this thesis. The output value estimates the distance between the function and a given pattern while the sign identifies the position of the pattern regard to the hyperplane (left or right side).

Because the SVM has only one output, the sign also indicates the class of the data (Figure 2.11: Left). The difficulty with the hyperplane strategies is that the functions are infinite, thus points outside the domain will be on one side of the function however relevant they are to the original task.

Figure 2.12: This drawing represents the drawback of a single value threshold. A single threshold does not take in account the shape of the cluster; it assumes that the cluster is spherical.



Other learning machines such as the RBF or the ART network approximate the landscape inside the domain, the kernels estimate the distance between themselves and a point (Figure 2.11: Right). Although the Distribution Landscape Approximation (Figure 2.11: Right) techniques are better armed, the kernels used for the landscape approximation are made of asymptotic function i.e. Gaussian thus their activation is unlimited. Although, it sounds easy to define a limit threshold, it is not often the case as the multiple dimensional clusters are rarely regular. Thus, every dimension and direction would require a different threshold (Figure 2.12). Choosing multiple thresholds is not an easy task as there is no guarantee that the irregularities will be along the axes (x, y, z...). Moreover, if the unknown data space and the regular data are not clearly separable then this strategy does not work efficiently. Several lines of research arose in order to overcome these difficult issues. If the data are normally distributed then a lot of information is rapidly available, information such as 95% of the data is contained within +/- two deviations of the mean thus anything else could be considered as an outlier/novel. Roberts (1999) assumes that the data are more or less normally distributed thus it becomes possible to use the extreme value theories. Extreme value theories are outside the scope of this thesis. It focuses on an extensive study of the tail distribution. Bishop (1994) takes a radically different approach by not assuming any density distribution. Instead, he uses a non-parametric technique to learn the data density distribution. A non-parametric technique is a

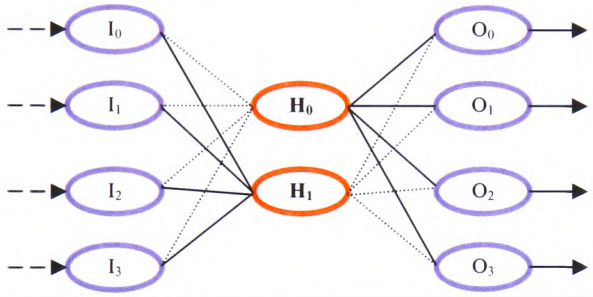
procedure that approximates the density distribution solely based on the data set without requiring any assumptions or prior-statistical analysis. The Parzen-window is a classical non-parametric technique; it replaces every point in a data set by a kernel function, often a Gaussian function. The equation (37) represents a Parzen window using Gaussian kernels. This type of kernel is often used because it generates a smooth distribution function.

$$P(x) = \frac{1}{n(2\pi)^{\frac{d}{2}}} \sum_{q=1}^{n} e^{\left[\frac{\|x-x^q\|^2}{2\sigma^2}\right]} \Rightarrow 1 < q < n \qquad (37)$$

n is the number of Parzen kernel, in this case they are Gaussian Functions, d is the input space dimension. σ is the smoothing parameter, in effect it is the Gaussian Kernel size. The density distribution is then transformed using the Log likelihood method. The log likelihood method simply scales the distribution using a logarithmic function. Bishop (1994) empirically finds the threshold; this rejection threshold is chosen as the value that correctly identifies the data in a test set as being known. An alternative to nonparametric techniques is to use an ANN as a novelty detector. Wilkins (1999) uses the RBFN's hidden layer (Chapter 2.1) as an approximation to the data distribution. In theory, the hidden nodes of the RBF could be used to form a Parzen Windows function. Wilkins (1999) showed that the sum of the nodes activation can be seen as an estimation of the significance of a particular point to the problem. He tested two different strategies to reject the data. In the first strategy, a point whose activation is lower than a learned threshold is rejected. In the second test, if the closest node's activation is lower than a threshold then the point is rejected. Although these approaches are relatively simple Wilkins (1999) was able to successfully reject some novel patterns. Singh (2002) presented another way of using the common ANN paradigms. Singh (2002) generates artificial unknown data. First, it analyses the regular data (as opposed to novel data), measuring the Min, Max, Mean and Standard deviation of the different parameter. Then its algorithm generates random points which are different enough from those data but not too different. It limits the irregularity by assuming that the regular data are roughly normally distributed and so the new data will not be closer than +/- two deviations from the mean. The two

deviations limits are chosen because when the data are normally distributed, 95% of the data lie within two deviations from the centre of the distribution function. Those artificial novel data are then grouped in a class. In a k class problem, the ANN has k outputs, an output "i" returns 1 as a positive identification of class "i" and 0 otherwise. The ANN is then trained to return a vector null $\{0_1,..,0_k\}$ for the Artificial Novel Class. If the outputs' activations are less than 0.5 the pattern is rejected. Japkowicz (1995) uses a multi-layer perceptron to learn to reconstruct the input on the output (Figure 2.13).

Figure 2.13: This drawing shows the principle of an "auto-encoder". The outputs reproduce on the inputs. The reduced dimension of the hidden layer forces the ANN to learn the signature of the global problem. A novel pattern should be badly reconstructed by the ANN thus it should be possible to identify it.



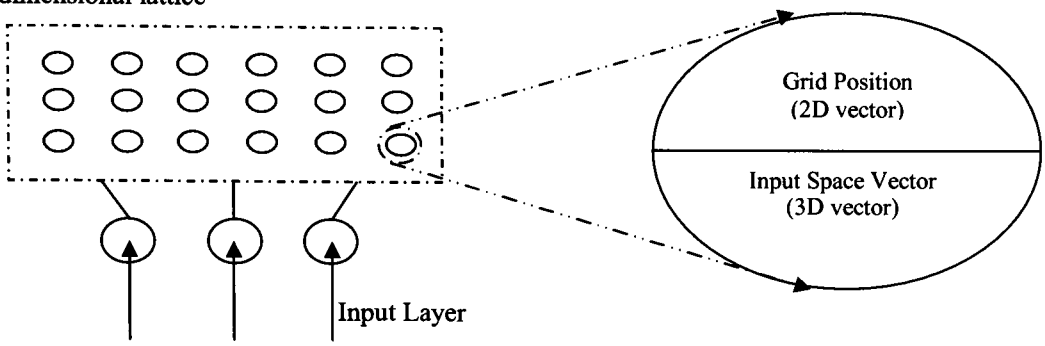
The ANN is trained by backpropagation. Once trained, the system is tested on regular data and on examples of novelty, the threshold is defined as the error value that produce no mis-identification of the regular data. The reconstruction error is computed using the RMS equation (38)

$$E_{RMS} = \sum_{i=1}^{n} (y_i - o_i)^2 \qquad (38)$$

where "n" is the number of output, "y" is the expected value and "o" the ANN output.

In an overview of different novelty detection techniques, Taylor (2000) uses a Self-Organizing Map (SOM) to reject data. The SOM network (Figure 2.14) was first described in Kohonen (2001). The SOM network is made of two layers. The first layer simply contains "n" nodes, where "n" is also the input vector's dimension.

Figure 2.14: Self Organizing Map. This SOM maps a 3 dimensional vector on a 2 dimensional lattice



These input nodes only transmit the information to all the second layer nodes. The second layer is made of a multi-dimension grid (often 2D) composed of logic functions kernels e.g. Gaussians. During the training, many patterns are sequentially presented to the SOM. For each pattern, a node is chosen as the winner. The winner is the node whose Euclidean distance to the pattern is the smallest. As illustrated in Figure 2.14, a SOM node is made of two parts;

1. The Grid vector stores the position of the node on the 2D grid.
2. The input space vector defines the position of the node in that data space.

The Euclidean distance is the distance between the input space vector of the node and the pattern. In order to reinforce the winner's probability to be the winner for a similar pattern, the node's position in the input space is updated (Equation 39).

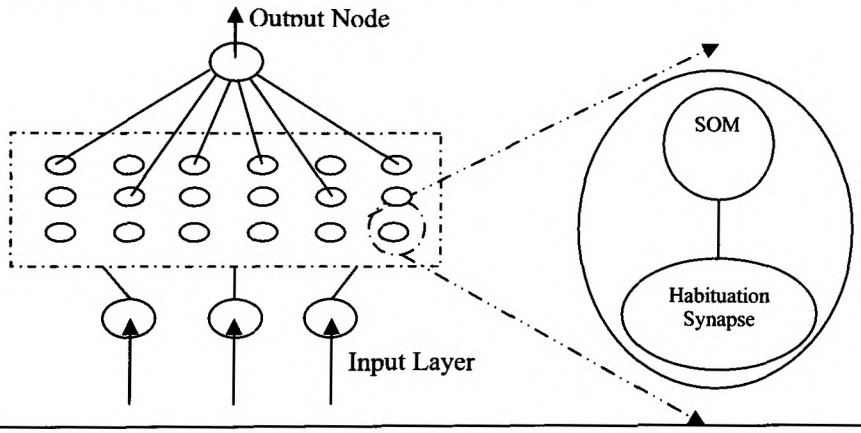
$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)] \tag{39}$$

$$h_{ci}(t) = \alpha(t).e^{-\frac{\|r_c - r_i\|^2_i}{2\sigma^2(t)}} \tag{40}$$

m_i is the i^{th} grid node, x is the input vector and $h(t)$ is the neighbourhood function. Equation (40) defines a Gaussian Neighbourhood whose size is σ. Equation (40) makes sure that the closer a node is to the winner, the more its characteristics will be updated in order to improve its activation for a similar pattern. The nodes that are further than 2σ are also updated but the improvement will be marginal. The learning rate is defined by $\alpha(t)$, $\alpha(t)$ is often a monotonically decreasing function. Updating the neighbour nodes

creates a clustering effect in the input space. As the neighbourhood is defined in the 2D grid space, it also generates a 2D projection of the N dimensional data on the 2D grid that is an interesting feature in order to interpret high dimension information. This is different from a Learning Vector Quantization algorithm that lowers the probability of the neighbours being activated by pushing apart the neighbouring clusters (Chapter 2.1). Taylor (2000) uses this capacity of the SOM algorithms to extract clusters from the data set. Once the grid has been trained (that is to say that the clusters have been defined), the new patterns are presented to the SOM. If the sum distance of a pattern from the SOM kernels is greater than a threshold θ then the pattern is rejected as novel. Marsland (2000) used the ability of the SOM algorithm to detect novelty online. In order for the system to detect novelty, the original SOM (Figure 2.14) was modified. The modified SOM was named the Habituation SOM (HSOM). The HSOM (Figure 2.15) is made of a classical SOM grid to which a "habituation" synapse is twinned.

Figure 2.15: Habitutation SOM. The drawing shows that a classical SOM node is replaced by a SOM node and a Habituation Synapse. The Output Neurones return the sum habituation activation.



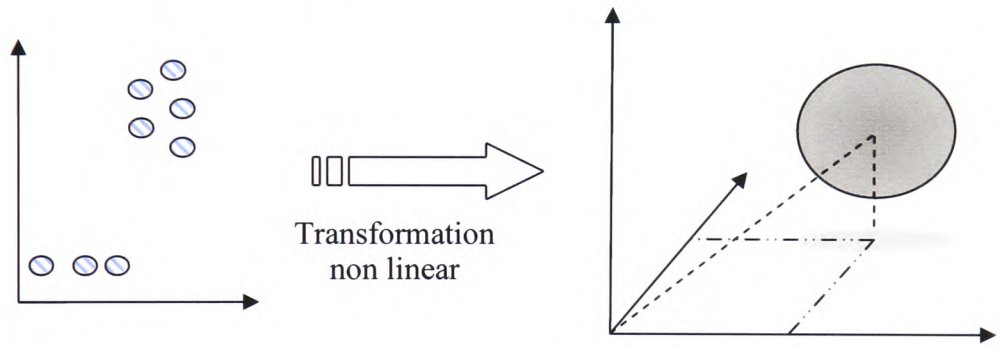
This synapse produces an activation value which is inversely proportional to the number of times its twin node is activated. The synapses habituate using the differential equation (40).

$$\tau \frac{dy(t)}{dt} = \alpha[y_0 - y(t)] - S(t) \qquad (40)$$

"y_0" is the original value before any habituation, $S(t)$ represents the stimulus on the current node, in this case the output of the SOM node. The HSOM is an online system thus there is no learning stage. While in regular use, the map progressively learns. If a given node was not activated before, the output of the

synapse will be high which means that the input vector is new. If the same vector is presented again, equation (40) will give a lower value thus the node gets use to the input vector. The novelty is the sum of all the synapses activation for a given input vector, a high value means that the vector's signature is novel to the HSOM. Finally, Campbell (1999) develops a new ANN paradigm inspired from the Support Vector Machine (Chapter 2.2). As for the SVM, the algorithm transposes the input data from the input space to a higher dimensional space called a Feature Space. In the feature space, the algorithm surrounds the pattern by a function (Figure 2.16).

Figure 2.16: The diagrams illustrates the learning principle behind Campbell (1999)'s idea. In the Feature space, the algorithm finds the function that surround the points while minimizing the distance between the points and the function.



$$w(\alpha,b)=\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_j K(x_i,x_j)+b,\sum_{j=1}^{m}\alpha_j=1 \quad (41)$$

$$K(x_i,x_j)=e^{\frac{-\|x_i-x_j\|^2}{2\sigma^2}} \quad (42)$$

The algorithm minimizes *"w"* over the training set to force the system to fit as closely as possible the function around the data. This function is called a hyper-sphere as the algorithm uses a Gaussian function (Equation 42). As for the SVM (Chapter 2.2), the points on the edge of the cluster are used to approximate the function and are called Support Vectors. Once the training is finished, the function generates a positive identification for the regular data and a negative value for the novel patterns.

2.6 Conclusion

The RBF and SVM neural network paradigms implement two very different learning strategies. Although, the latter seems to have an edge, it has never been applied to the phytoplankton domain. Moreover, this work does not only intend to produce good identification and a high level of flexibility, it also requires a novelty rejection technique. In order to offer a high level of flexibility, this project carried on the investigation of the HPNN architecture, the HPNN architecture is similar to Anand (1995)'s approach. Although, a lot of literature was produced in MNN field, the aim of those techniques was to improve or speed up the learning capacity of ANN. Single monolithic techniques have successfully been applied to the phytoplankton problem. Their pitfall was not their learning capacity but the incapacity of the ANN to grow without retraining. They also require expert knowledge for the training.

For the detection and rejection side, many interesting approaches were presented in the literature however none of them were 100 % appropriate for different reasons. This particular problem cannot use the SV base approach [Campbell, 2000] because it is computationally expensive and it is unproven. Moreover, it depends on the size of the RBF kernels size. The appropriate size has to be found empirically, it works as a rejection threshold. On the other hand, the methods described by Taylor (2000) and C.M. Bishop (1994), implement similar non-parametric methods that could possibly be modified in order to reduce the rejection threshold choice burden.

This project investigated methods to improve the ANN flexibility, to reject unknown without requiring empirical tests while not requiring expertise in ANN.

Chapter 3: Adding species without retraining

3.1 Introduction

Chapter 1 states that a major problem to overcome in order to apply ANN to phytoplankton identification is that it should not be necessary to fully retrain it in order to add new species. Moreover, the architecture should not necessitate neural network expertise in order to be used. This chapter presents a MNN technique that overcomes or minimises these problems. The first part assesses the ability of the RBF and SVM paradigms when applied to the proposed MNN architecture. It compares these results to other comparable studies and tests different combination strategies. In the second part, the ability of the new architecture to accept new species without retraining is tested. The second part of the tests also assesses the scalability of the system.

3.2 Hardware and software environment

The following tests were conducted on a PC equipped with an Athlon Thunderbird 1 GHz, 512 MB of DDR RAM and an ATI Radeon 64 DDR graphic card. The software ran under Windows NT, and was created and compiled using Borland C++ Builder 4.

3.3 The basic architecture and tests

3.3.1 Introduction

In chapter 2.4, several MNN algorithms were presented as possible candidates, however none of those common approaches are suitable to the current problem. In fact, the learning capacity is not a primary problem as several studies showed that the common multi-layer perceptron can successfully learn to identify cytometric data [Frankel, 1989; Balfoort, 1992; Morris, 1994].

In order to allow scaling without full retraining, the problem needs to be decomposed. Moreover, the decomposition should allow different species to be added or removed. In Masulli (2000) and Anand (1995)], a k class problem is divided in k^* 2 class sub-problem. This so-called One-Per-Class decomposition breaks up the problem into a number of 2 class' identification problems. Each sub-problem is made of a **"Target"** class $i \in [1...k]$ that has to be differentiated from a **"Background"** class made of the remaining $k-1$ class (Figure 3.1). A single expert (ANN) learns one sub-problem.

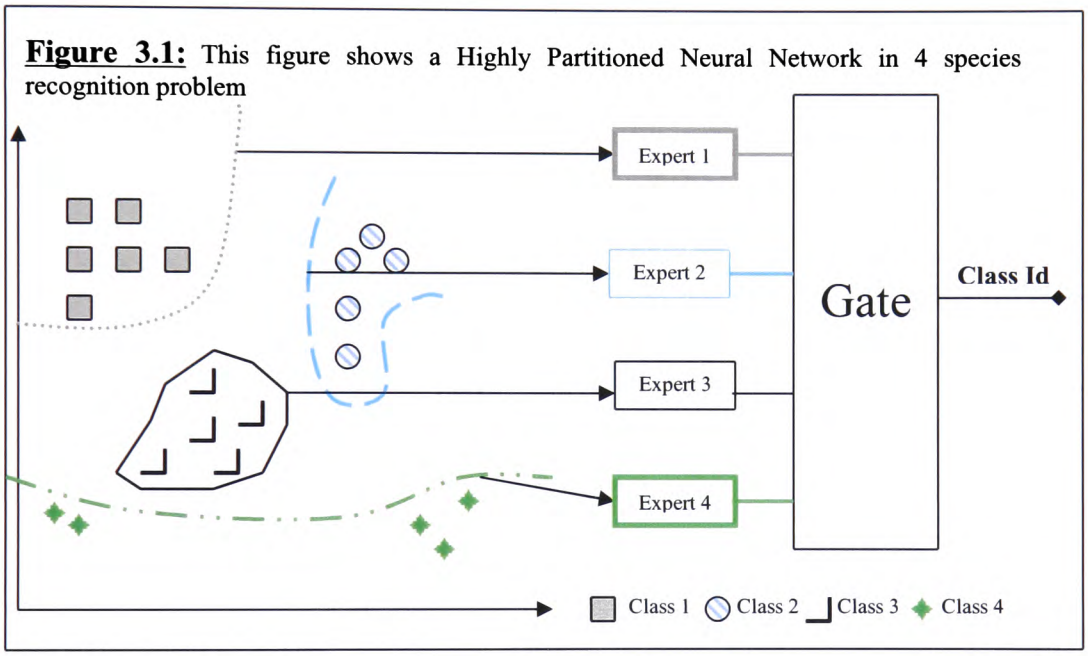


Figure 3.1: This figure shows a Highly Partitioned Neural Network in 4 species recognition problem

Figure 3.1 illustrates the One-Per-Class decomposition architecture. The gate is the mechanism that merges the k sub-problems into a single larger k class problem. It is used to choose between the experts if several of them identify a pattern as their target class. This decomposition method should allow new species to be added to the MNN system without full retraining. Every expert learns to separate a target class from a background class. The background class is a generalised class thus adding new species to the system should not change it greatly if this generalised class is generic enough. New experts will be trained for every new species. The system will then use those new experts for the new species. The existing experts should not need to be retrained. Morris (1998) applied this technique successfully to flow cytometry on a small-scale problem; the sub-tasks were learned using the classical RBFN strategy (Chapter 2.2). When several experts identify a pattern as their target, the gate

applies basic Winner-Takes-All (WTA) to select the right class. This WTA technique selects the expert with the highest output activation as the winner; it is a type of Hard Switching procedure. It will be called WTA throughout this thesis. Morris (1998) called this MNN architecture, the Highly Partitioned Neural Network (HPNN).

3.3.2 Test 1: Preliminary assessment

3.3.2.1 Introduction

This test assesses the potential of the SVM and RBF experts in the HPNN architecture and compares these systems with a monolithic system. These tests were published in Morris (2000).

3.3.2.2 Data set and tests description

The data sets are made of 62 species of phytoplankton typical of those found in the North Sea. However, the cells were grown in a controlled environment in order to give pure species, 1000 cells per species. The cells were then analysed with the EurOPA FC [Wilkins, 1999]. The first test used the classical software and hardware environment common to all the tests However, these tests use a special module (Diffraction Module) of the EurOPA Flow Cytometer; it adds four parameters to the normal 7 parameters (Chapter 1.1). These parameters are:

1. The vertical bar.
2. The horizontal bar.
3. The Outer ring.
4. The Inner ring.

These preliminary tests used three types of ANN:

a) Monolithic RBF (MRBF)

b) HPNN architecture made of RBF experts

c) HPNN network made of SVM experts.

a) The MRBF networks were constructed with eleven and twenty outputs (thus, eleven and twenty species). Networks of 40, 60, 80, 100 and 120 kernels were trained and their outputs analysed by means of a mis-identification matrix. In each case, the kernels were Gaussian functions and the placement strategy used was random. The RBF hidden kernels has a width σ=2.0. This width value was found empirically. Each species was represented by 400 patterns in the training set and 400 new patterns in the test set.

b) Single species RBF networks were constructed and trained with 60 kernels, equally divided between the class of interest and the background class. The kernels were Gaussian functions and placed randomly within the class of interest and the background class. Each network had two outputs corresponding to the class of interest and the background class. The output with the highest value was taken to be indicative of the identification between the two classes. The RBF hidden kernels had a width σ=2.0. This width value was found empirically. Each species was represented by 400 patterns in the training set and 400 new patterns in the test set.

c) Single species SVMs were constructed for each of the species in the eleven and twenty species populations. RBF kernels were used with a kernel width of 316.2 and gamma set at 0.00001. The mis-identification threshold was set at 0.9. Again, the identification was based upon the highest output from the two possible classes. Each species was represented by 400 patterns in the training set and 400 new patterns in the test set.

The single ANNs are combined into a single coherent system using the WTA strategy (Chapter 2.3). With the WTA strategy, the winner is the ANN with the highest output.

3.3.2.3 Results

The MRBFs give average overall identification of between 57 and 69% for the eleven species (Table 3.1) and between 54 and 65% for all twenty species (Table 3.2).

Table 3.1: Eleven species data set - percentage correct identification for single large RBF network with varying number of kernels

	Number of kernels				
Species	**40**	**60**	**80**	**100**	**120**
Chrysochromulina camella	51	53	59	64	77
Emiliania huxleyii	90	92	95	95	98
Gymnodinium simplex	60	67	62	64	57
Gyrodinium aureolum	77	62	60	59	59
Halosphaera russellii	57	57	61	65	67
Heterocapsa triquetra	37	61	61	60	61
Phaeocystis globosa	83	86	85	83	82
Pseudopedinella sp.	20	28	34	36	36
Rhodomonas sp.	89	96	94	97	99
Tetraselmis rubescens	20	22	28	30	40
Thalassiosira sp.	50	69	72	82	84
Mean correct	57	63	65	67	69

Of the eleven species, *Pseudopedinella sp.* and *Tetraselmis rubescens* were the least well identified, with successful identification of less than 40%. All other species were identified at 60% success or better. These same species together with *Gymnodinium simplex* were least well identified (<40% success) in the twenty species network; thirteen species were identified with >60% success.

The single species nets alone gave an average of 64% for the RBF nets and 88% for the SVMs with eleven species (Table 3.3), whilst 63% and 77% correct identification for RBF and SVM respectively were achieved for the twenty species (Table 3.4).

Table 3.2: Twenty species data set - percentage correct identification for single large RBF network with varying number of kernels

	Number of kernels				
Species	**40**	**60**	**80**	**100**	**120**
Alexandrium tamarensis	79	85	75	78	79
Chlorella salina	41	38	38	38	40
Chroomonas	40	36	47	47	55
Chrysochromulina camella	28	38	42	37	41
Cryptomonas calceiformis	94	100	100	92	98
Dunaliella tertiolecta	97	93	95	95	95
Emiliania huxleyii	95	92	92	92	89
Gymnodinium simplex	11	12	17	38	38
Gyrodinium aureolum	11	14	38	42	45
Halosphaera russellii	55	53	59	62	64
Heterocapsa triquetra	68	68	71	69	67
Ochromonas sp.	54	83	90	88	90
Phaeocystis globosa	86	87	86	85	84
Porphyridium pupureum	96	96	98	98	98
Prymnesium parvum	75	84	80	81	78
Pseudopedinella sp.	12	23	20	20	24
Pyrarimonas obovata	59	77	80	81	83
Rhodomonas sp.	71	75	66	77	66
Tetraselmis rubescens	0	3	7	10	10
Thalassiosira	19	13	46	56	63
Mean	54	58	62	64	65

Table 3.3: Eleven species data set - percentage correct identification for individual single species networks (for RBF 30 kernels per class), combined networks using "winner-takes-all" strategy and number of support vectors for individual SVMs

	Single nets		Combination of single nets (WTA)		Number of support vectors
Species	**RBF**	**SVM**	**RBF**	**SVM**	**SVM**
Chrysochromulina camella	60	94	80	92	529
Emiliania huxleyii	94	99	97	99	151
Gymnodinium simplex	46	85	61	85	654
Gyrodinium aureolum	52	80	62	83	529
Halosphaera russellii	50	84	65	83	538
Heterocapsa triquetra	71	91	79	93	441
Phaeocystis globosa	74	91	85	92	373
Pseudopedinella sp.	41	78	56	77	737
Rhodomonas sp.	99	99	99	99	52
Tetraselmis rubescens	43	88	57	85	579
Thalassiosira sp.	73	79	88	90	522
Mean	64	88	75	89	

For the eleven species, the two species *Pseudopedinella* sp. and *T. rubescens* poorly identified in the large RBFs were successfully identified about 40% of the time in the single species RBFs, but 78% and 88% in the SVMs (Table 3.3). Likewise with twenty species, the identification of *Pseudopedinella sp.*, *T. rubescens* and *G. simplex* was much higher (75%, 79% and 82% respectively) with SVMs than single species RBFs (14%, 7% and 25% respectively) (Table 3.4). The SVMs for *Pseudopedinella sp.*, *T. rubescens* and *G. simplex* required the most support vectors of all the species (Tables 3.3 and 3.4).

Table 3.4: Twenty species data set - percentage correct identification for individual single species networks (for RBF 30 kernels per class), combined networks using "winner-takes-all" strategy and number of support vectors for individual SVMs

	Single nets		Combination of single nets (WTA)		Number of support vectors
Species	**RBF**	**SVM**	**RBF**	**SVM**	**SVM**
Alexandrium tamarensis	75	94	87	96	271
Chlorella salina	29	84	57	83	772
Chroomonas	92	99	98	100	105
Chrysochromulina camella	33	89	59	88	673
Cryptomonas calceiformis	98	99	98	99	82
Dunaliella tertiolecta	90	98	97	98	207
Emiliania huxleyii	89	99	98	99	187
Gymnodinium simplex	25	82	52	82	827
Gyrodinium aureolum	49	80	61	84	516
Halosphaera russellii	37	89	64	86	541
Heterocapsa triquetra	60	87	72	88	408
Ochromonas sp.	91	97	95	97	208
Phaeocystis globosa	70	92	85	91	365
Porphyridium pupureum	97	99	98	99	79
Prymnesium parvum	79	97	87	96	234
Pseudopedinella sp.	14	75	34	73	762
Pyrarimonas obovata	63	87	83	86	469
Rhodomonas sp.	92	99	96	99	96
Tetraselmis rubescens	7	79	37	78	847
Thalassiosira	63	81	84	87	547
Mean	63	90	77	90	

Combining the single species nets and using winner-takes-all (WTA) to determine the identification boosted the identification success of *Pseudopedinella* sp. and *T. rubescens* in the eleven species sets of RBF networks to 56 and 57%, and the overall average successful identification rose to 75% (Table 3.2). For the twenty species sets correct identification of *Pseudopedinella* sp., *T. rubescens* and *G. simplex* increased to 34%, 37% and 52% respectively when the single species RBFs were combined with WTA; overall success increased to 77% (Table 3.4). The performance of the SVMs was little changed by the introduction of the WTA strategy in either the twenty or the eleven species data sets.

3.3.2.4 Discussion and Conclusion

The preliminary tests showed that the HPNN architecture gives similar result to other neural network applications [Wilkins, 1999] in phytoplankton identification without requiring as much optimisation. However, it also showed that an HPNN/RBF system does not boost performance. Although the combinations produce superior performance to the MRBFs, the overall HPNN/RBF system does not significantly outperform the MRBF.

However, had the MRBF (Monolithic RBF) and the HPNN/RBF been optimised, the performance of the latter would have been better per species than with the MRBF. With a modular architecture, it is possible to increase the size of the local expert, if its performance on the target species is not satisfactory. On the other hand, a monolithic system (such as the MRBF) only allows improvement on the global problem thus the identification rate can greatly vary from one class to the other. The HPNN/SVM system on the other hand repeatedly out classed the RBF based systems and the scaling of the system had little impact on it, e.g. Tables 3.2 and Table 3.3 show similar identification rate. The HPNN/SVM benefits from the adaptive architecture of the SVM paradigm. The network size and the parameter values adjust as long as the network does not reach the desired success rate. It is more powerful but also takes more training time. Conversely, the SVM pool of experts did not improve with the use of the WTA gating strategy.

3.3.3 Test 2: Larger Problem

3.3.3.1 Introduction

The previous tests showed that the HPNN architecture with either the RBF or SVM have potential. Next, the systems were tried on a much larger problem in order to measure the scalability of the architecture. This sequence of tests was published in Morris (2001).

3.3.3.2 Data set and tests description

The training set contained 62 species representative of the phytoplankton from the north see. Each species is composed of 1000 cells grown under a controlled environment in order to guarantee pure species. The cells were then analysed using the EurOPA FC, this FC extracted seven type of information (Chapter 1.2) [Wilkins, 1999]. *Emeliana huxleyi* was represented by two strains. The training set was made up of the 62 species; each species consisting of 500 randomly selected examples. The testing set was constructed with 500 randomly selected new examples of each of the 62 strains. All the ANNs used the same parameters values as in Chapter 3.2.2.3. This time, the 62 single species ANN were trained to differentiate between one class that contains one species from the background class that is made of the 61 remaining species.

First, the performances of the individual experts were assessed on the testing set then the experts were combined using the WTA strategy and tested on the same testing set.

3.3.3.3 Results

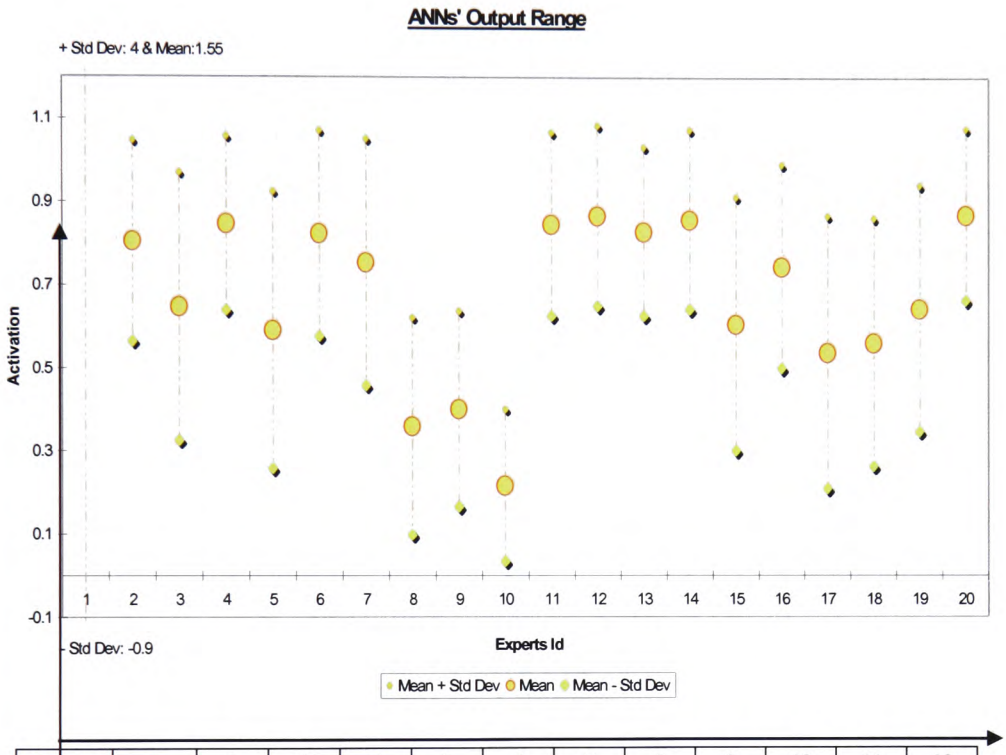
The single RBFs performed extremely badly, the successful identification rate is 16%. The success rate standard deviation varies by 26.02% on the overall k class' problem. 37 experts failed to learn and four experts identified more than 70% of their target species patterns.

The SVM experts identified 77% with deviation of 15.68% of their target class' patterns correctly however nine experts did not distinguish 60% of their target patterns (one of them only identified 26%).

The HPNN/RBF improved the success rate of the single RBF experts to 50%, the standard deviation is 29.04% but 35 species are identified correctly $\leq$ 60% (4% to 60%) of the time. That is an improvement of 34% over the single species experts.

The HPNN/SVM identified 81% of the examples correctly, which is a 4% improvement. The standard deviation is equal to 12.67%. Only three species were identified with less than 60% and 44 of them successfully identified more than 75% of the patterns (examples in the training set).

Figure 3.2: Analysis of the HPNN experts' output



Species	1	2	3	4	5	6	7	8	9	10	11	12
Max	18.95	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.83	1.00	1.00
Min	0.00	0.04	0.01	0.10	0.00	0.01	0.00	0.00	0.01	0.00	0.02	0.02

Species	13	14	15	16	17	18	19	20
Max	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Min	0.01	0.00	0.01	0.02	0.00	0.00	0.01	0.03

The analysis of the experts' signatures, SVM or RBF, show that the activation range for positive identification varies dramatically from one expert to the other, e.g. in Figure 3.2, the range [-Std Dev, Mean, +Std Dev] of the expert 10 is lower than the one of expert 4.

3.3.3.4 Discussion and Conclusion

The RBF experts failed to learn adequately the task (Mean 16%, Std 26.02%). However the HPNN/RBF was still able to categorize 50% of the patterns but the variation is still important (29%). The contrast between the performance of the single experts and the HPNN/RBF shows that even though the single experts are not able to correctly identify the patterns, the pool of experts still learn something about the global k class problem. It means that the pool of experts compensate for the poor performance of the individual experts. It is probable that the small number of kernels (60 kernels) is not appropriate for this problem. In fact, the parameters are identical to the ones in the previous test but with a task more than three times more complex. It is probable that another crucial factor for the poor performance of the single RBF experts is the heavily unbalanced training set. The ratio background/target classes is 31000/500=62, as the RBF algorithm minimizes the error over the training set (Chapter 2.2), the algorithm mostly learned the background class. In order to improve the performance of the RBF expert, it might be necessary to reduce the number of patterns in background class by removing the insignificant ones. It might also be possible to weigh the error on the classes in order to reduce the effect of the unbalanced training set. It could also be possible to repeat the patterns in the target class in order to improve the background/target ratio; a desirable ratio would be 1:1.

The SVM, on the other hand, confirm their robustness as the success rate only dropped by 8% with a more difficult task. The HPNN/SVM only outperformed the single experts' success rate by 4% but it reduced the success deviation by 3%. A smaller deviation means that fewer experts do much worse than the average.

Moreover the average performance improvement on the worst performers (single experts success on their target species: 26 and 73%) was higher than 6%.

As for the HPNN/RBF, the experts that gain more from the group are the poorer learners. It means that the HPNN architecture improves the generalisation performance of the overall system.

The HPNN/SVM also outperformed Boddy's (2000) MRBF by 4% and is more consistent over all the species as its standard deviation is 12.67% against 16.40%. Moreover, the MRBF was not able to learn one species i.e. *Emiliania huxleyii*. This is a significant result as the SVM algorithm automatically learned the task and still outperformed the MRBF despite the fact that it was manually optimized.

Figure 3.1 highlights a common problem to all ANN paradigms, which is that the output of the ANN is not proportional to the confidence in its answer. For example, expert 1 in Figure 3.1 learned poorly its two class problem. Thus the activation range for a target class identification is wide and the average activation value is much higher than the one of the other experts. Despite this poor performance with WTA, this expert would most of the time be the winner if it clashed with another expert.

3.3.4 Conclusion

The tests show that the HPNN architecture is a scalable architecture when it is combined with the Support Vector Machines. They prove that the SVM paradigm is extremely powerful, the system still benefits from the "combination" strategy as it improved the overall performance.

Although the HPNN/RBF performed poorly on the large-scale problem, it did well on the small one, effectively outperforming the MRBF. On the large-scale problem, the performance of the non-adaptive RBF paradigm was poor. However the massive performance improvement (+34%) produced by the HPNN/RBF + WTA indicates that with a better choice of parameter choice the situation may be different.

The analysis of the outputs (Figure 3.1) shows that the WTA is not reliable as the output of the experts is not a confidence estimation. Moreover, the combination strategy should take into account the output range variation between experts. Furthermore, a better method should take into account the reliability of the experts during the decision process.

3.4 Advance HPNN strategy

3.4.1 Introduction

Previous tests showed that the HPNN is scalable but that the gating system is not reliable. Moreover, the RBF experts have performed poorly in the previous tests because of the inadequate parameter choice. This section attempted to test again the RBF with a better choice of RBF parameters. It presents an improved HPNN architecture. Then, the ability of the architecture to add species without any further training of the existing experts were tested. The scalability of the new HPNN architecture was also assessed.

The training speed of the two ANN paradigms was also measured.

This architecture and the following tests were submitted in Autret (2002).

3.4.2 New gate: Feature Based Decision Rule

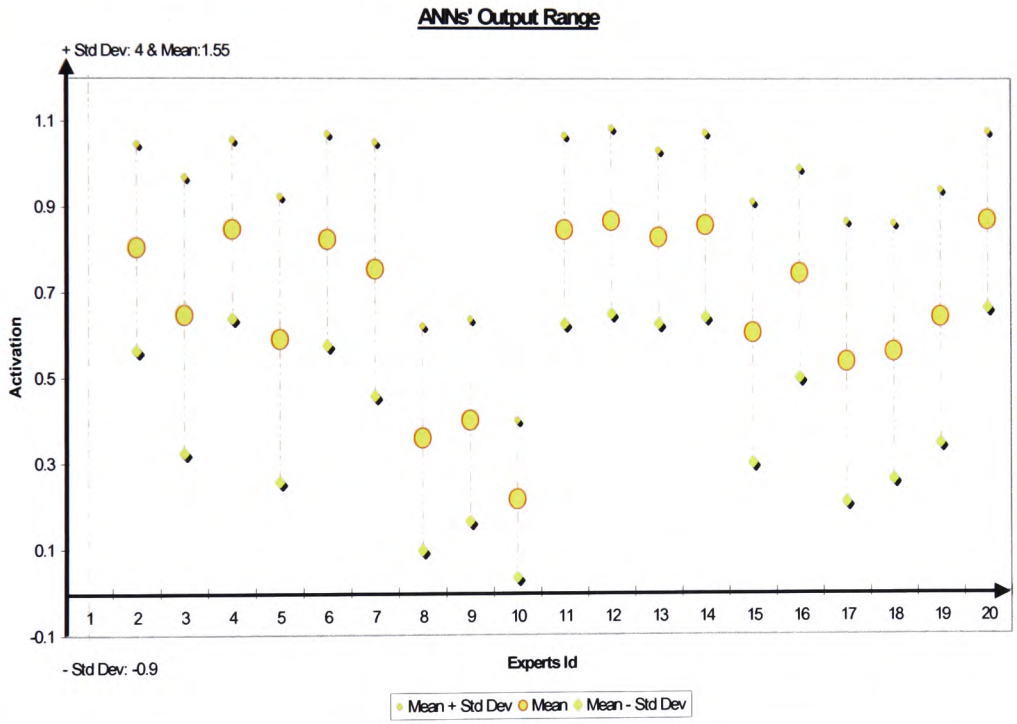
There are two main types of ANN learning. The first type learns to approximate a hyperplane that separate the data e.g. backpropagation or SVM, the second approximates the data distribution of the different classes e.g. RBF. Although, these strategies are different, they rely on the common assumption that is that the different classes (species) are members of different clusters. Sometimes the clusters might overlap but there have to be clusters. The output value is a rough estimate of the distance between an input vector and the centre of the clusters in the case of the RBF type or the distance between the input vector and the hyperplane for the SVM type. Thus, an ANN should generate a consistent and different range of output for every class it was trained to identify. With HPNN, every expert is trained to identify two classes, the target class and the background class. The target class is made of one species of phytoplankton and the background class of all the other species in the study. In order to develop a common system, the RBF's two outputs are merged into a single number. There is one output for the target class and another for the background class, the chosen class is the one with the highest activation.

Thus, it is possible to express the two outputs with one single value as written below

$$O = \left\| O_{background} \right\| - \left\| O_{target} \right\|$$

$O_{background}$ represents the output that identifies the background class and O_{target} represents a positive identification of the target class. Thus, if O is negative, the pattern is from the target class. It is similar to the SVM algorithm that generates a negative output for a target class pattern. As explained in Chapter 1.2, bagging is based on the assumption that the sum distribution of several sub-sets is closer to the real distribution than the distribution of a training set made of all the training examples from those sub-sets.

Figure 3.3: Analysis of the HPNN experts' output



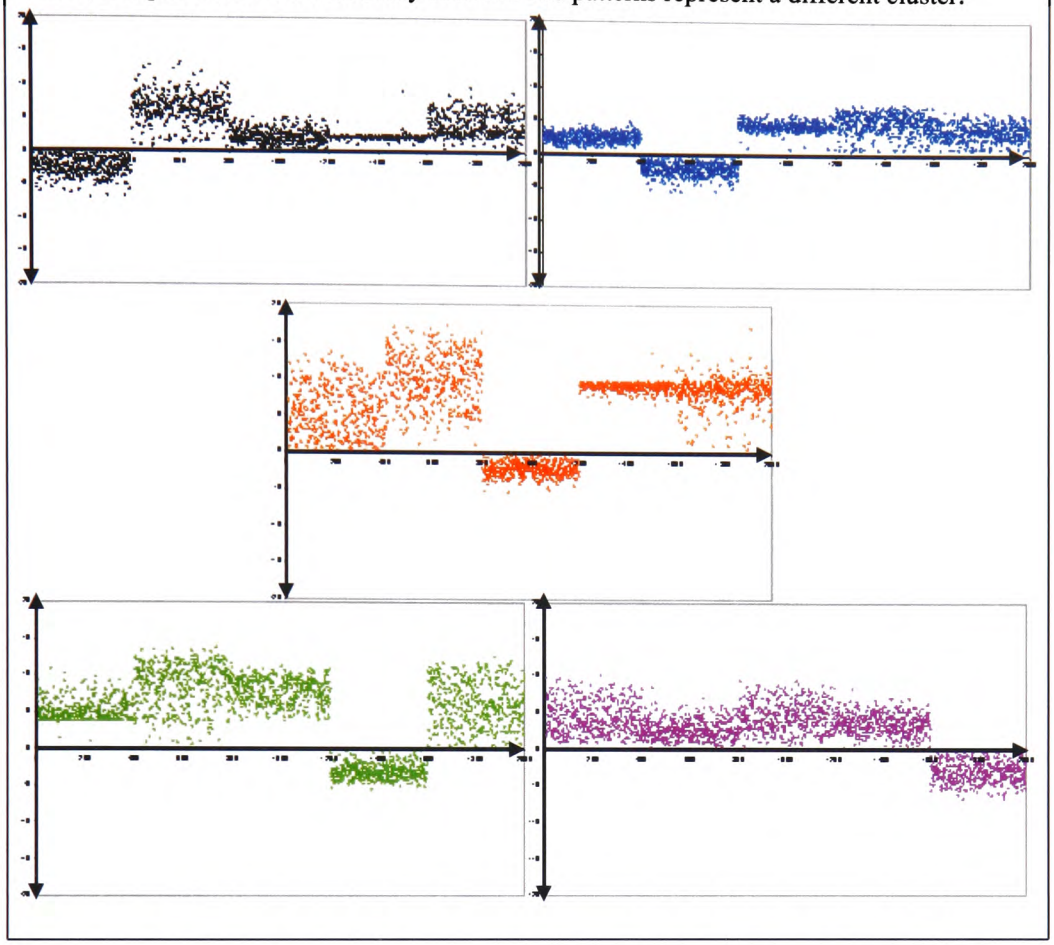
Thus, by creating sub-sets of the training set and then training a different ANN on each of them, every ANN will learn a slightly different identification rule. By grouping the different experts afterwards, the system learns a generalised rule that is closer to the ideal one. Although the experts do not learn exactly the same problem, they use the same data; the difference between them is the species on which they focus. Thus, grouping them should still improve the performance while allowing new target species to be added without retraining. The other type of breakdown techniques described by Anand (1995) would not

have allowed this. In most of those techniques, the target class is made of several classes/species, thus adding something means retraining several experts. Therefore, if an adequate combination system is used the HPNN should be able to allow resizing of the system while giving good performance. Chapter 3.2 showed that the winner-takes-all (WTA) strategy works adequately with small problems but with fails on larger ones. Why did it fail on the larger system? The biggest problem faced by a system designer while combining the multiple experts is that the range of output value is different for every expert (Figure 3.3). In fact the networks approximate a function that separates the two classes. However, the distance between the data and the function (Hyperplane or Area function) depends on the data. If the data are close to each other, the distance will be small. If they are far from each other the distance will be bigger. Furthermore, the range is affected by the shape and size of the cluster; that is well clustered data produce a small range of output. Finally, if an ANN does not train properly its output values are inconsistent, e.g. Expert 1 (Figure 3.3).

For all these reasons, the WTA can only work on small systems because the data tends to be relatively well clustered. The technique that was developed had to be simple and at the same time it had to be able to cope with the difference between output signatures and with the poorly trained ANN experts.

The analysis of the tests in Chapter 3.2 showed that although the ANNs are trained to separate the target class from the background class made of all other species, they have a relatively well-clustered output range for each of the species (Figure 3.4).

Figure 3.4: From top left to bottom right: Expert one to five. For every expert, a group of negative outputs represents the target species. Although the other classes are not as well defined, it is possible to see that every block of 400 patterns represent a different cluster.



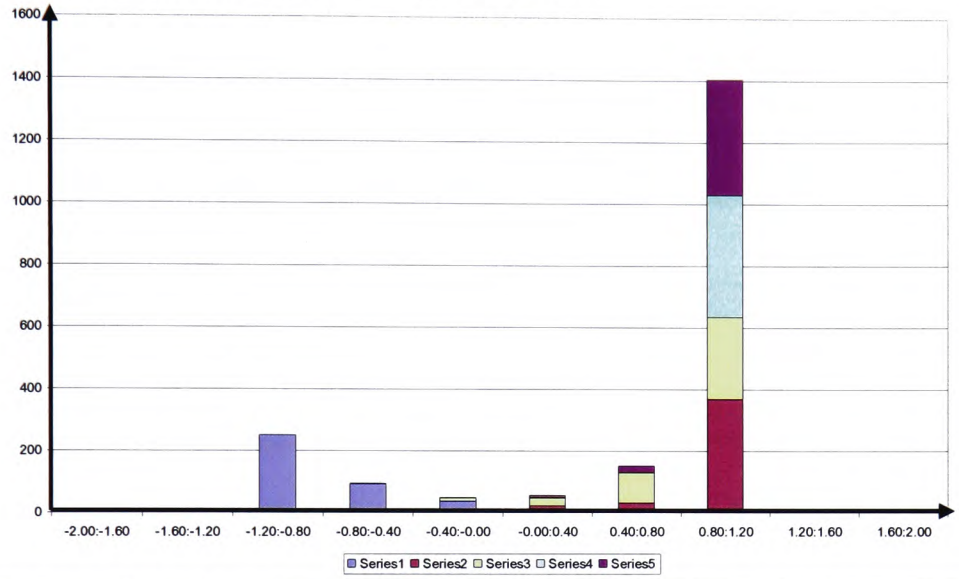
This knowledge could be use to choose between the experts when there is a tie or even to improve the performance of the system. However, it was said earlier that the activation value cannot directly be used so how does the system combine the experts? Well, it is not possible to use the activation values but it should be possible to use the values distribution. One of the most common and reliable techniques to map a data distribution is the histogram. However, the system would need to map the activation distribution of one expert for every species. Thus, the number of histograms for one N species system would be N^2. The system in this research can be made of 62 experts that would be a prohibitive number.

Moreover, the system should estimate the likelihood of having a species A and not a species B or C for a given value. For example, during the training the expert took 100 times value X, 20 times it was "A", 40 times it was a "B" and 60 times it was a "C". From these observations, it is more likely that the

current species is a "C" than an "A" or "B". The solution is to create a histogram in which every interval stores the likelihood of having every species (Figure 3.5).

Figure 3.5 shows that the histogram generates the likelihood of having one

Figure 3.5: This diagram map the behaviour of expert 1 on a hypothetical 5 species system. The distribution is split in 10 ranges of equal size.



species in a given interval but also the likelihood of having this interval. The only user defined parameters in the combination strategy is the number of divisions of the histograms (bins). The system uses the following procedure to create the histograms.

Histograms principle

Variables
nbExperts: the number of experts (Species)
nbIntervals: the user-defined number of intervals in the histograms
nbPatterns : the number of patterns per species in the training set
exArray: contains the nbExperts*nbPatterns examples
min: array that stores the lower boundary of every expert output range
max: array that stores the upper boundary of every expert output range
intInc: array that stores the interval width for every expert
histograms: array that stores the histograms information. It is made of nbIntervals Elt object.
Elt
{
nbHit => stores the total number pattern that generated an output in this range
min => lower boundary
max => upper boundary
hitPerSpecies => Array that stores the number of hits in the interval for every species
}

Comment: Find the limits of the histograms and the width of the intervals
Initialise **min** and **max** to zero
For **i**=0 to **nbExperts** do
    For **j**=0 to **nbExperts*nbPatterns** do
        If **exArray[i][j]** less than **min[i]**, **exArray[i][j]** becomes **min[i]**
        If **exArray[i][j]** more than **max[i]**, **exArray[i][j]** becomes **max[i]**
    End
End
For **i**=0 to **nbExperts** do
    intInc[i]=(max[i]-min[i])/nbIntervals
    Round **intInc[i]** to the closest greater integer
End

Comment: Create the histograms
tmpVal: temporary variable
For **i**=0 to **nbExperts** do
    tmpVal = **min[i]**
    For **j**=0 to **nbIntervals** do
        histograms[i].min[j]=tmpVal
        tmpVal = **tmpVal** + **intInc[i]**
        histograms[i].max[j]=tmpVal
    End
End

Comment: Count the number of hit per species per intervals
For **i**=0 to **nbExperts*nbPatterns** do
    For **j**=0 to **nbExperts** do
        Find the appropriate interval for **histograms[j]**
        Add one to **histograms[j].nbHit**
        Check the origin of the pattern stores pattern species number in **spId**
        Add one to **histograms[j].hitPerSpecies[spId]**
    End
End

Once trained, the system calculates the probability of occurrence of every species based on the information in the histograms. The winner is the species

that is the most likely base on the histograms. In order to calculate this probability, the system computes equation (37).

$$P(t \mid X) = \frac{\sum_{i=1}^{n} P_i(t \mid \mathrm{X_i})}{\sum_{b=1}^{n}\sum_{i=1}^{n} \mathrm{P_i}(b \mid \mathrm{X_i})}, b \neq t \qquad (37)$$

'n' represents the number of species i.e. the number of experts that make the system.

X is the *'n'* dimensional feature vector, X_n being the active feature (the range in which the output falls) for dimension *'n'*.

't' is the target species and 'b' stands for one of the species from the background class.

$P(t \mid X)$ is the probability of having the target class given the feature vector X and $\mathrm{P_i}(b \mid \mathrm{X_i})$, the probability of having species *'b'* for feature X_i with expert 'i'. In effect, the denominator calculates the probability of having anything else but *'t'*.

It gives the following procedure.

Finding the winner

Variables:
nbExperts: number of experts(equal number of species)
validInterval: array that store the position of the valid interval for every expert for one pattern
Histograms: array that stores the histograms information (See Histogram Principle)
inputPattern: Array that stores the output of the experts
probabilities: Array that stores the probability of having the different species
winner: store the id of the winning species

```
//Search the position of the intervals that contains the inputVector values
For i=0 to nbExperts do
   Search valid interval in histogram[i] for inputVector[i], store it position in validInterval[i]
End

//Calculate probabilities of having every single species
Numerator: Temporary variable
Denominator: Temporary variable
For i=0 to nbExperts do Comment: For every species
   For j=0 to nbExperts do
      numerator= numerator + histograms[j]. ].hitPerSpecies[validInterval[i]]   } Eq (37)
      denominator= denominator + histograms[j]. nbHits
   End
   probabilities[i]=numerator/denominator
End

Find the highest probability in probability, store its position in winnerId
```

The process of "finding the winner" (above) based on the histograms is called a feature based decision rule (FBD). The FBD method is an extension of Anand (1995) or of Massulli (2000). In those papers, the experts are associated with flags; the flag is set to 1 if the expert identifies the pattern as its target, to 0 otherwise. Then, every flag combination is stored in a matrix and associated to a class. The winner is selected based on this matrix.

The FBD spawned two different gating techniques:

1. The partial feature based decision rule (PFBD)
2. The full feature based decision rule (FFBD)

The partial feature based decision rule (PFBD) uses the FBD technique as the referee. When only one expert identifies a pattern as the target, it is designated as the winner. If none of the experts identifies the pattern as their target class, the system rejects the pattern. The FBD technique is only required when several experts identify a pattern as their target class. It is assumed that because the experts store knowledge on every species, the sum of this knowledge should be enough to choose the right species. When several species have the same probability, the pattern is rejected. It is better to reject a pattern than to mis-identify it. Moreover, the EurOPA FC can take a picture of the cells that go through so that rejected cells/patterns can be analysed manually later on.

The Full FBD strategy (FFBD) is closer to the method described by Anand (1995) or Massulli (2000) as it always uses the referee to choose the species. Thus, each identification is based on the group knowledge. Of course, as the expert separates in priority their target class from the rest, they store more information on this particular species thus, they have more influence when a pattern comes from their target species. In fact, the histograms strategy gives more influence to well-clustered behaviour and experts generates this type of clustered behaviour on their target species as this is the only one that produces a negative output. When several patterns produce the same probability they are rejected.

It is expected that these FBD strategies will allow the system to produce better identification performances than a single system and that is, they will allow the system to grow with a minimum of training. Moreover, the principle

is based on a common technique that should be easy to understand, interpret and apply even for non-specialist.

3.4.3 Data set

The data sets were made up of phytoplankton from 62 species typical of the northern European seas. They were grown under controlled conditions [Wilkins, 1999] to give pure species that is to guarantee that every individual is correctly labelled. The seven identification parameters were extracted using the Becton Dickinson FACSort™ flow cytometer. The data were then filtered to remove any unwanted noise. Finally, the parameters were linearly scaled such that every one of them had a mean of 0 and a standard deviation of 1.0.

3.4.4 Tests description

Three series of 20 randomly selected species were taken and a training set consisting of 400 randomly selected patterns per species was created (Annexe 9.1.2). Single experts were trained to identify a target class against the background class made of the rest of the 20 species. A test set of 400 new randomly selected patterns per species was also created. Each test involved creating two distinct systems: one with RBF experts and another with SVM experts.

Each RBF expert had 160 Gaussian Kernels in the hidden layer, 80 for the target classes/species and 80 for the background one. Those kernels had a width of 4.0 and were positioned using a modified Learning Vector Quantization (LVQ in Chapter 2.2) strategy. The LVQ strategy was modified to mitigate the effect of the unbalanced training set on the system, i.e. the RBF algorithm minimizes the RMS error on the overall training, thus with a classical LVQ, the background class would use most of the hidden kernel consequently the error minimization would be heavily biased toward it. The modified LVQ makes sure that 80 kernels will map the target class and 80 kernels will map the background class by positioning the two batches of

kernels separately. The rest of the procedure used standard matrix resolution. The appropriate parameter choice was found empirically.

The SVM experts were made of Gaussian Kernels, in order to be able to find a hyper-curve (Chapter 2.3). The penalty term was set to 1000 to force good separation of the classes and the mis-identification threshold to 0.9.

Two basic systems of experts were created, one using RBF experts and one using SVM experts. Each system consisted of 20 experts, each trained to identify one species against the other 19. This was repeated for each of the three sets of species. Once the basic systems (20 experts system) had been trained and tested, they were re-tested on the 42 remaining species for which they had no expertise (The global data set contains 62 species with 1000 patterns per species). These test sets were made of 400 patterns from those 42 species. A mis-identification matrix was created, and the unknown species were sorted by the number of times they had been wrongly identified as a known species. The mis-identification matrices were used to select new species to add to the systems (six at a time as adding the species one-by-one would take too long), starting with the most difficult ones (most mis-identifications).Of course, as the RBF and SVM paradigms are different, the three series of tests produced a different mis-identification matrix for the two ANN. Thus, the three series of tests produces 3*2 mis-identification matrix. The six new experts were trained to separate their individual target species from the already known ones and from the five other new species. The experts were trained with exactly the same setting as the 20 basic ones. Every time new experts were added to the system, the gate (Decision stage) had to be re-trained. When using the PFBD and FFBD methods, the histogram dimension (number of intervals) was set to 10.

A second test was conducted on the gating system alone in order to estimate the architecture sensitivity to the histograms' dimension when using the PFBD and FFBD methods. This test used the three batches of 62 trained RBF and SVM experts from the previous test and only changed the dimension from 10 to 50, 100, 200 and 300. The systems were then retested on the existing test files.

During the system training, the learning time was monitored automatically.

3.4.5 Results

- Single experts RBFs

For the original RBF systems (20 species) and over the three test sets, the individual experts successfully identified 83% (Table 3.5) of their respective target classes' patterns on average. Those same experts identified roughly 93% (Table 3.5) of the background classes' patterns correctly, e.g. expert 1 identified around 93% of the pattern from species 2... 19, 20 as background patterns.

In the final RBF systems with 62 species, the individual experts recognized 71% (Table 3.5) of their target class and identified successfully 82% (Table 3.5) of the background class. There is still a bias of around 10% toward the background class. The identification performance of the system degraded regularly over the target and background classes identification. The difference in identification between the twenty species system and the 62 species system was -12%.

- Single experts SVMs

The original SVM system experts identified around 88% of the target patterns (Table 3.6) and 99% (Table 3.6) of the background patterns. A bias of 11% can be computed from those two results. For 62 species, the single SVM experts correctly categorized 84% of the target class and 98% of the background patterns. Again, there was a bias of 14% towards the background class.

Table 3.5: Three series - RBF (80/80, Gaussian kernel, width 4.0, LVQ)

For each sequence of tests, the system was trained on twenty different species, every species being associated to one ANN expert. New species were added starting with the most difficult. That is the species that were the most frequently incorrectly classified as know species. The order of inclusion of the remaining one is different as the three series start with twenty different species.

For every new addition, the PFBD and FFBD combination strategies are retrained.

Sequence 1	SGLE		WTA	PFBD	FFBD
	Correct Target	Correct Background	Percentage Correct On The Test Set		
Original	83.98	88.75	0.70	75.64	74.99
26	86.13	79.25	2.52	66.14	71.71
32	76.66	78.19	0.17	66.90	71.92
38	77.53	78.88	0.24	68.41	71.66
44	76.11	79.81	0.00	67.35	70.84
50	66.98	82.24	0.00	59.27	62.34
56	65.86	83.54	0.00	58.38	62.00
62	72.33	82.11	0.00	63.88	67.84

Sequence 2	SGLE		WTA	FFBD	FFBD
	Correct Target	Correct Background	Percentage Correct On The Test Set		
Original	77.03	95.84	27.45	76.14	82.23
26	77.27	92.41	1.34	72.64	74.45
32	74.59	91.28	0.00	68.91	69.73
38	73.05	92.49	0.00	67.68	68.50
44	73.74	91.50	0.00	68.03	68.30
50	68.41	88.80	0.00	62.58	68.85
56	67.64	86.72	0.00	61.43	67.51
62	65.15	83.73	0.00	58.09	66.22

Sequence 3	SGLE		WTA	FFBD	FFBD
	Correct Target	Correct Background	Percentage Correct On The Test Set		
Original	86.34	94.26	26.69	85.45	87.51
26	84.12	83.72	0.05	80.24	81.17
32	83.34	81.77	0.00	78.53	78.86
38	81.21	78.99	0.00	73.91	73.86
44	79.20	79.39	0.00	71.18	72.09
50	79.79	78.14	0.00	71.01	72.06
56	77.01	79.27	0.00	68.28	69.60
62	75.33	81.21	0.00	66.86	68.28

Table 3.6: Three series – SVM

For each sequence of tests, the system was trained on twenty different species, every species being associated to one ANN expert. New species were added starting with the most difficult. That is the species that were the most frequently incorrectly classified as know species. The order of inclusion of the remaining one is different as the three series start with twenty different species.

For every new addition, the PFBD and FFBD combination strategies are retrained.

Sequence 1	SGLE		WTA	PFBD	FFBD
	Correct Target	Correct Background	Percentage Correct On The Test Set		
Original	89.10	99.41	87.89	87.53	86.76
26	89.88	98.39	83.11	87.63	84.14
32	89.14	98.05	85.91	98.23	92.12
38	88.22	98.05	72.34	82.72	77.57
44	86.45	98.08	67.07	78.65	72.89
50	84.41	98.12	62.98	76.19	69.94
56	82.55	98.20	60.68	73.86	67.59
62	83.77	98.28	59.71	75.67	69.88

Sequence 2	SGLE		WTA	PFBD	FFBD
	Correct Target	Correct Background	Percentage Correct On The Test Set		
Original	84.88	99.27	83.70	83.38	80.30
26	86.98	98.17	73.13	85.30	81.24
32	86.52	97.91	67.32	83.13	78.37
38	85.48	97.89	62.77	80.07	75.03
44	85.34	97.95	61.10	77.94	72.39
50	84.30	98.05	58.32	76.35	70.65
56	83.79	97.11	55.86	75.01	69.51
62	83.87	97.31	55.08	75.02	69.65

Sequence 3	SGLE		WTA	PFBD	FFBD
	Correct Target	Correct Background	Percentage Correct On The Test Set		
Original	91.05	99.59	90.09	90.05	88.33
26	91.08	98.42	78.36	89.34	86.10
32	90.94	97.95	75.94	87.84	85.75
38	89.47	97.87	69.24	84.32	80.76
44	88.30	97.90	63.90	80.85	76.51
50	87.45	97.99	58.32	79.63	75.62
56	85.37	97.97	57.47	75.79	70.87
62	84.92	97.96	55.85	74.84	69.49

- Combination strategies and RBFs

From the beginning the simple winner-takes-all strategy is rejected as its results are extremely poor (Table 3.6) with less than 20% correct identification on the original systems.

The basic system with PFBD with a 10 intervals histogram gave an average identification percentage of 79% (Table 3.6). The full systems with 62 species identified 63% of the patterns successfully (Table 3.6). The FFBD systems with 20 species recognised 82% of the pattern correctly and the same FFBD system with 62 species correctly identified 67.5% examples. For both systems, the identification performance degraded by 15% as the systems scaled up.

- Combination strategies and SVMs

The SVM systems differ drastically from the RBFs as the results with the simple WTA is not as bad of those of the RBFs (compare Tables 3.5 & 3.6).

On the basic system, the WTA strategy leads with 87% (Table 3.6) of correct identification, closely followed by the PFBD (86.98%). The FFBD method is the worst with 85% correct replies, which is 2% worse than the WTA method.

The full-scale systems give a radically different picture from the basic ones, as the WTA falls to 56.88% (average of the three arrays) correct identification. On the other hand, the FBD strategies maintained their previous identification performance with the PFBD bettering the FFBD strategy by 5.5 (difference between the average success rate of the PFBD and FFBD strategies) percent with a success rate of slightly more than 75% (average of the three arrays).

- Changing the FBD dimension parameter

Both types of system gave similar results in that different sizes can boost the FBD performance by a few points. The FFBD benefits more from this parameter tuning than the PFBD, about 0.5% (Table 3.7) against more than 1%. The SVM systems tend to gain slightly more than the RBF except for the FFBD where the gain is important with a rise of between 1.87% and 5.66%.

Table 3.7: Test on the histogram bins (dimension) influence with a 62 species RBF/SVM systems

Test Series	Histogram Bins	RBF		SVM	
		PFBD	FFBD	PFBD	FFBD
1	10	63.88	67.84	75.67	69.88
	50	64.17	68.77	76.46	72.35
	100	64.62	68.94	76.17	71.57
	200	64.40	67.94	75.96	70.66
	300	64.06	66.67	75.97	70.33
Maximum Improvement		0.74%	1.1%	0.79%	2.47%
2	10	58.09	66.22	75.02	69.65
	50	58.67	67.67	75.60	71.52
	100	58.73	67.61	75.57	70.95
	200	58.29	66.68	75.29	69.99
	300	58.00	65.92	75.10	69.53
Maximum Improvement		0.64%	1.45%	0.58%	1.87%
3	10	66.86	68.28	74.84	69.49
	50	67.46	69.47	76.10	72.15
	100	67.35	69.05	75.98	71.51
	200	67.21	68.42	75.85	70.72
	300	66.80	67.21	75.77	70.23
Maximum Improvement		0.6%	1.19%	1.26%	5.66%

- System's training and testing time

From Table 3.8, it is easy to see that the RBF learning algorithm is much faster than the SVM one. It is also possible to see that the combination strategies training time is negligible compared to the one of the experts.

Table 3.8: Training time for different parts of the system
This table displays the fastest and slowest training time for the two types of experts (SVM and RBF) and for three combination strategies (WTA, PFBD and FFBD) while running the tests.

	Lowest training time	**Highest training time**
RBF (Per expert)	3 Seconds	30 Seconds
SVM (Per expert)	10 Minutes	1 Hour
WTA	No training	
PFBD	(Histogram dim 10) 1 Minutes	(Histogram dim 300) 2 Minutes
FFBD	(Histogram dim 10) 2 Minutes	(Histogram dim 300) 4-5 Minutes

3.4.6 Discussion and conclusion

The Support Vector Machines have shown their superiority over the RBF network, the individual SVM experts being better than the RBF ones by a good 10% (comparison of the average success rate between tables 3.5 and 3.6) on the original system and by 20% on the 62 species network. It is however important to notice that the SVM implements an adaptive algorithm (Chapter 2.3), in which the system automatically scales up depending on the problem complexity, which is radically different from the RBF, which has a fixed size. The side effect of the adaptive algorithm is that the SVM experts are slower to train than the RBF experts (Table 3.8).

It is interesting to note the good performance of both systems, especially the SVM, while identifying the background class. It is also interesting to note that the highly unbalanced data set (up to 61 times more pattern in the background class than in the target class) did not affect greatly the training performance of the individual experts. This was expected with the SVM, as this algorithm is not sensitive to this kind of problem. However, it was surprising that the RBF did not suffer from the unbalanced data sets as several papers had suggested that it might affect training [Anand, 1995]. The small effect of the unbalanced set can be explained by the LVQ algorithm modifications. This LVQ technique optimized the kernel position by class and not on the overall data set, thus it led to an over-representation of the target class data. This in turn increased the influence of the error on target class during the matrix resolution.

As expected when analysing the performance of the individual experts, the HPNN/SVM systems out-performed the HPNN/RBFs. In the HPNN/RBF case, the WTA strategies failed completely, which is without any doubt linked to the fact that many of the RBF experts had generalisation problems and that no attempt was made to optimise them. This is important as it means that the tests represent a worst-case scenario. Thus, with a little bit of care during training, the result could be greatly improved.

Although the WTA strategy performed better with the HPNN/SVM, the performances were still less than adequate when the systems scaled-up. This is probably because as new experts were added the existing ones were not retrained. Thus, their generalisation abilities were not good enough to cope with the new classes. As the WTA method bases its choice on the outputs, the system failed.

On the other hand, the two FBD strategies did not encounter this problem. It is certainly linked to the decision process that is based on the distribution of the output during the training. A network that generates an inconsistent output has less influence than one that is more reliable.

The two system types (HPNN/RBF & HPNN/SVM) gave two different results. The HPNN/RBF networks performed 2 to 3 percent better with the FFBD than with the PFBD. This is due to the Gaussian kernels that make up the hidden layer of the RBF algorithm. In fact, RBFs position the non-linear kernels in the input space to map the problem, because every expert was trained on the same data but with a slightly different focus, the picture they had of the individual class was good. Thus, grouping the knowledge of the different experts improved the generalisation of the overall system on every individual species. It is similar to a bagging technique (Chapter 2.3).

On the other hand, the HPNN/SVM system worked best when combined with the PFBD strategy. It comes from the good generalisation ability of the SVM algorithm, i.e. if an expert rejects a pattern it is best to reject it as it has an accurate idea of what the target class should be. The other reason comes from the adaptive strategy, in fact the SVM map the data in a feature space. The problem for the FFBD is that it does not as for the RBF map the data but finds a separation hyperplane in this feature space. Thus, the output of one expert is not, as for the RBF, an insight into the position of a pattern with respect to the centre of its class but it is an estimation of the distance from the hyperplane.

Moreover, the HPNN offers similar performance to Morris (2001) with the RBF/FFBD (87% successful identification against 81%) and better performance when it uses the SVM/PFBD pair (87 against 87). This is important as in the cited paper, a MRBF was optimized to identify only 11 species and with more flow cytometer parameters (11 against 7 in this case).

The results can also be compared to Boddy (2000) in which another MRBF was optimized to identify 72 species (Table 3.9), although this system gave slightly better results, it did not allow as much flexibility as the current modular approach. In fact, adding new species to this system would require a full retraining of the system. The problem with training a single multi-class network being that it is not possible to tune the network on a particular species, which would have been possible with this system. Thus, in Boddy (2000) some species performed much worse than others (Table 3.9). In these tests, the individual experts were not tweaked to improve the identification because the tests aimed at testing a worse case scenario. However, it could easily have been done with the HPNN modular architecture as the experts are independent.

The last series of tests on the histogram (Table 3.7) dimension showed that it can be useful to try different settings as it can easily and effortlessly bring a few percentage points improvement.

3.5 General Conclusion

This chapter has worked through a series of tests to the development of an entirely novel gating strategy. The partitioning of a multi-class problem using the HPNN strategy and the development of the Feature Base Decision strategies gave birth to a new flexible MNN system. This architecture allows its user to tune the system in order to improve the performance of individual species. Moreover, new species can be added to the system without retraining the existing experts and with only a small degradation of the performance compared to a fully retrained system. Thorough tests proved that the new architecture is highly scalable, e.g. during the tests, the system size was multiplied by three. Furthermore while combined with the RBF, the system gave similar performance to an optimised MRBF that does not offer as much flexibility. When coupled with the SVM paradigm, the new system out-performed any other systems applied to FC analysis in the literature.

The following chapter attempted to develop the new architecture further in order to allow it to reject novel data (unknown species).

Table 3.9:

Comparison of the large RBF network (trained using Mahalanobis distance metric; 135 hidden layer nodes), single species RBF ANNs (trained for each species separately against a background of the rest), combined single species RBF ANNs, single species SVMs and combined single species SVMs. For each species, the percentage correct identification is shown.

				[2]RBF – 1 large net	RBF – single species, individual nets	RBF-single species, combined nets	SVM – single species, individual nets	SVM-single species, combined nets	Number of support vectors
Crypto-phytes	Crypto-phyceae	*Chroomonas sp.*	8-10	95	0	4	96	97	157
	Crypto-phyceae	*Chroomonas salina*	5-12	93	48	88	94	94	460
	"	*Cryptomonas appendiculata*	15-25	98	89	93	99	99	90
	"	*Cryptomonas calceiformis*	10-15	92	0	6	94	94	471
	"	*Cryptomonas maculata*	12-20	92	53	81	92	91	456
	"	*Cryptomonas reticulata*	18-25	94	0	74	95	95	475
	"	*Cryptomonas rostrella*	16-25	99	0	80	99	99	495
	"	*Hemiselmis brunnescens*	5-8	65	0	21	65	69	347
	"	*Hemiselmis rufescens*	4-9	54	0	30	58	59	296
	"	*Hemiselmis virescens*	5-8	96	0	91	94	96	480
	"	*Plagioselmis punctata*	6-9	90	0	94	90	92	459
	"	*Rhodomonas sp.*	8-13	94	0	18	94	95	325
Flagellates	Prasino-phyceae	*Micromonas pusilla*	1-3	99	50	62	97	98	171
Flagellates	Prasino-phyceae	*Nephroselmis pyriformis*	4-7	73	0	20	74	72	860
	"	*Nephroselmis rotunda*	6-8	46	0	9	60	59	293
	"	*Pyramimonas grossii*	5-10	66	1	70	65	71	1230
	"	*Pyramimonas obovata*	4-8	65	0	29	61	69	1149
	"	*Tetraselmis impellucida*	11-19	92	70	78	97	97	89
	"	*Tetraselmis striata*	6-8	67	21	51	75	80	1634
	"	*Tetraselmis suecica*	6-15	86	0	47	86	89	623
	"	*Tetraselmis tetrathele*	10-16	95	59	77	93	95	251
	"	*Tetraselmis verrucosa*	3-11	61	0	33	72	76	1066
	Chloro-phyceae	*Chlamydomonas reginae*	11-20	92	32	76	88	92	634
	Chloro-phyceae	*Chlorella salina*	4-8	60	0	7	66	71	1188
	"	*Dunaliella minuta*	3-12	65	0	18	66	73	1420
	"	*Dunaliella primolecta*	5-12	85	36	76	89	90	623
	"	*Dunaliella tertiolecta*	6-12	85	25	60	83	87	770
	"	*Stichococcus bacillaris*	5-8	65	0	3	70	75	980
	Rhodo-phyceae	*Porphyridium pupureum*	4-6	95	0	95	96	97	660
	Rhodo-phyceae	*Rhodella maculata*	7-24	92	0	1	96	95	190
	Chryso-phyceae	*Ochromonas sp.*	3-12	63	0	39	60	63	1273
	Rhodo-phyceae	*Pelagococcus subviridis*	2-3	89	70	89	88	87	441
	"	*Pseudopedinella sp.*	8-10	71	0	58	65	74	1065
Prymnesio-phytes	Prymnesio-phyceae	*Chrysochromulina camella*	6-12	87	62	83	83	86	576
Prymnesio-phytes	Prymnesio-phyceae	*Chrysochromulina chiton*	5-9	62	0	35	51	64	1391
	"	*Chrysochromulina cymbium*	6-10	43	0	32	26	42	1669
	"	*Chrysochromulina polylepis*	6-8	59	0	48	52	62	1213
	"	*Emiliania huxleyi B11*	5-7	89	93	97	96	96	144
	"	*Emiliania huxleyi 92*	5-6		0	26	80	83	807
	"	*Ochrosphaera neopolitana*	8-10	38	0	40	52	67	1635
	"	*Pavlova lutheri*	4-6	78	0	76	73	76	806
	"	*Phaeocystis pouchetii*	3-6	61	0	26	61	66	1430
	"	*Pleurochrysis carterae*	10-18	87	43	57	93	93	465
	"	*Prymnesium parvum*	8-10	80	0	31	77	81	758
Diatoms	Bacillario-phyceae	*Amphora coffaeformis*	10-20	88	25	69	88	89	614
Diatoms	Bacillario-phyceae	*Chaetoceros calcitrans*	4-6	88	0	83	89	91	417
	"	Phaeodactylum tricornutum	8-35	92	0	18	92	93	247
	"	*Skeletonema costatum*	3-5	72	8	45	75	79	841
	"	*Thalassiosira weissflogii*	12-20	90	64	82	87	89	479
Dino-flagellates	Dino-phyceae	*Amphidinium carterae*	15-20	74	23	43	71	80	913
Dino-flagellates	Dino-phyceae	*Aureodinium pigmentosum*	7-12	86	9	62	81	85	568
	"	*Gymnodinium micrum*	8-15	72	0	22	69	73	1020
	"	*Gymnodinium simplex*	6-10	63	2	50	69	77	935
	"	*Gymnodinium veneficum*	9-16	45	0	14	52	65	1128
	"	*Gymnodinium vitiligo*	7-22	69	1	53	57	70	1047
	"	*Gyrodinium aureolum*	35-45	86	60	79	82	85	434
	"	*Heterocapsa triquetra*	15-27	78	41	81	79	84	808
	"	*Prorocentrum balticum*	9-15	75	16	28	77	78	906
	"	*Prorocentrum micans*	30-40	80	0	65	72	76	790
	"	*Prorocentrum minimum*	16-18	57	0	26	63	68	1071
	"	*Prorocentrum nanum*	8-10	51	0	17	69	74	1358
	"	*Scrippsiella trochoidea*	30-42	49	0	10	73	78	870
Overall				77	16	50	77	81	749

[2] data taken from Boddy et al. (2000)

Chapter 4: Novelty detection and rejection

4.1 Introduction

The previous chapter presented an innovative architecture that enables a problem to grow without any retraining of the existing pool of experts. However before ANNs can be applied to phytoplankton studies, they should be able to reject uncharacteristic signatures. First, this chapter describes a technique that arose from the need to reject unknown data; it is based on the characteristics of the HPNN architecture.

Finally, it discusses preliminary tests on this new rejection technique.

4.2 Problem background and paradigm overview

The problem of detection of the unknowns is probably the most difficult of all. It is hard to define the unknown thus it is even harder to detect it. How do humans detect unknown situations or objects? The dictionary definition is that "A novel event is an incident whose occurrence is unprecedented or rare".

In cognitive science, an unknown or novel object is something that is associated with little or no information. Problems differ with respect to novelty detection. In a well-clustered problem, for which the amount of information is extensive and includes all the variations, the detection of novelty is easy. The quantity of information available on the known species will filter out completely the unknown. For example, let us consider a simple shape recognition problem, the standard objects are square or rectangular, weigh one kilogramme and are either blue or green. In this example, novelty detection is simple as the number of attributes is small and the number of states for each attribute is limited. The difficulty with most application is that the number of attribute is much higher and the number of states that each attribute can take is high, thus programming all those possibilities is difficult or impossible. There are others reasons that can complicate novelty detection. For example, the known data are not clearly clustered thus the novel patterns can easily overlap with them. The amount of information necessary to transform the novelty problem into a separable identification is not available. For example, in a human identification problem the human and monkey clusters are not separable

if the only parameters available are the height and weight. The human and monkey height and weight range overlap, as does the ratio. A supplementary parameter such as the hairiness would make the problem separable, as monkeys are hairier than humans. Finally, in automated novelty detection, there are different kinds of novelty:

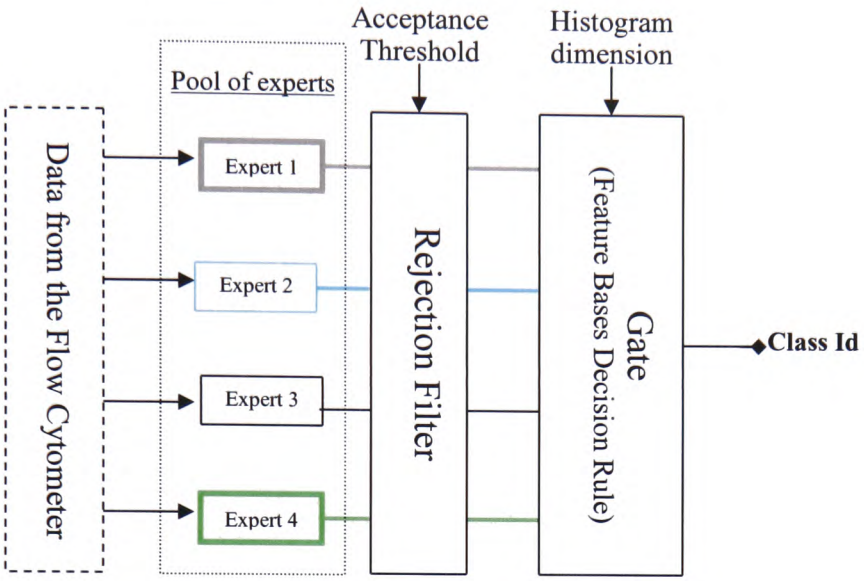
- In an R^N species space, with N being the number of known species. If the system is trained to recognise m species with $m<N$, the novel patterns belong to the N-m remaining species.
- In an R^N species space, with N being the unknown number of species, m is the number of known species, t is the number of species in the study; with $t<m<N$, the unknown is t-N.
- An invalid input vector is also a novel object e.g. sensor failure.

Those different novelties require different detectors. The current rejection system omits the latter kind. Although these two types of unknown detection problems look similar, the second problem is more difficult than the first. In the first type of problem, the novel classes are known thus some information is available. Moreover, the problem domain is finite. The second case is much more difficult as there is no information available on the unknown classes and the domain is unbounded as far as the system designer is concerned. Chapter 2.4 presented two different approaches to novelty detection. The first approach attempts to generate a class of unknowns, then the ANN learns to differentiate this class from the others. In order to generate the novel class' patterns, the system analyses the normal data. Then the system randomly creates those patterns outside the normal data signature spectrum. The success of this approach depends on the quality of the unknowns patterns. Furthermore, it is difficult to apply this technique if the unknown class is not separable from the normal classes. These two criteria are unlikely in this project. The second strategy attempts to detect novelty by finding a common signature within the normal data set. It then rejects the data that do not correspond to the known signature. Some studies such as Wilkins (1999) assume that the ANNs discover the signature while learning how to identify the different classes. Thus, a low sum activation of the output indicates the novel patterns. The auto-encoder (Chapter 2.4) represents another approach. It trains an ANN to reproduce the

input vector on the output vector. Thus, a high reconstruction error indicates a novel pattern. The drawback of this approach is that it requires another ANN, which needs training, which requires expertise. The rejection threshold represents the highest reconstruction error on the known patterns. If a pattern generates a bigger error, the system rejects it. The problem is that this rejection process is not easy to understand. The technique developed for this research should not make any assumption on the unknown data contrary Roberts' (1999) approach. Moreover, this technique should not require an empirical assessment of the rejection threshold as Bishop (1994), Wilkins (1999), Campbell (2000) or Aggarwal (2001)].

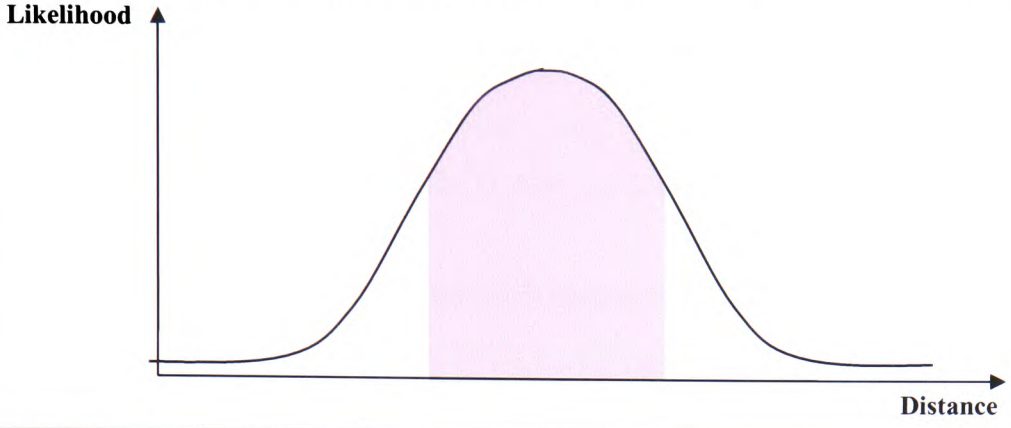
The system that was developed for this project uses a pool of experts to learn a multi-class problem. The proposed rejection technique will use this pool of experts as a pre-processing stage. The rejection filter then uses the output of the experts in order to map the density distribution. The output of one expert for a given pattern is stored in a vector with the output of the other experts for the same pattern. This vector is called the behaviour vector. Once every pattern from the training set has been through the pool of experts, a new data set called the behaviour data set has been created. The rejection filter uses this behaviour data set during its learning stage. This pre-processing stage is important as the pool of experts learns to identify the species thus it clusters the data in the class space. The dimension of this class space is higher than the input space or data space. The density distribution maps the data density distribution in the class space after pre-processing. It is called the behaviour density distribution as the function approximates the normal interactions of the experts with each other. As the rejection filter is added to the previously developed combination stage, the HPNN architecture takes the structure described in Figure 4.1.

Figure 4.1: The rejection filter developed in this chapter works between the pool of experts and the gating system developed in chapter 3. If a pattern does not satisfy the acceptance criteria in the rejection filter, it does not go further. The combination and rejection stages work with the output vector from the pool of experts.



The current technique is also different from Bishop (1994) because the threshold does not control the amount of unknown behaviour that will be rejected but the amount of normal behaviour that will be accepted. Of course, the inclusion of behaviour starts with the most common. The idea is that it is impossible to quantify the amount of unknowns that will be rejected by the system in normal use as nobody can predict the signature of the unknown patterns. However, by rejecting the most unusual behaviour, the system increases the likelihood of rejecting real unknowns (Figure 4.2).

Figure 4.2: The data in the centre of the normal distribution are the most typical and the outliers are the sides. The information available on the normal distribution states that 68% of the most typical behaviour lies within + and - one deviation (pink surface) from the centre. The 42% rejected represent less typical behaviour.



4.3 Implementation and Problems Encountered

The previous paragraph explained the principle and origins of the technique. However, it presented the technique applied to a one-dimensional space and the distribution was normal. The normal distribution has been widely used and studied thus it comes with a vast number of known properties, for example a well known property of the normal distribution is that 68% of the data are within +/- one deviation from the mean.

In real life application the vector space will be multi-dimensional (in this case, up to 62) and the distribution will be unknown. It is also likely that the density function will be very different from the normal distribution. Thus, none of the usual properties will be available and the shape of the curve will be unknown.

Mathematics provides several approaches to approximate the density distribution:

1. Check if one of the common shapes can be applied
2. Transform the distribution
3. Approximate the distribution with a semi-parametric method.

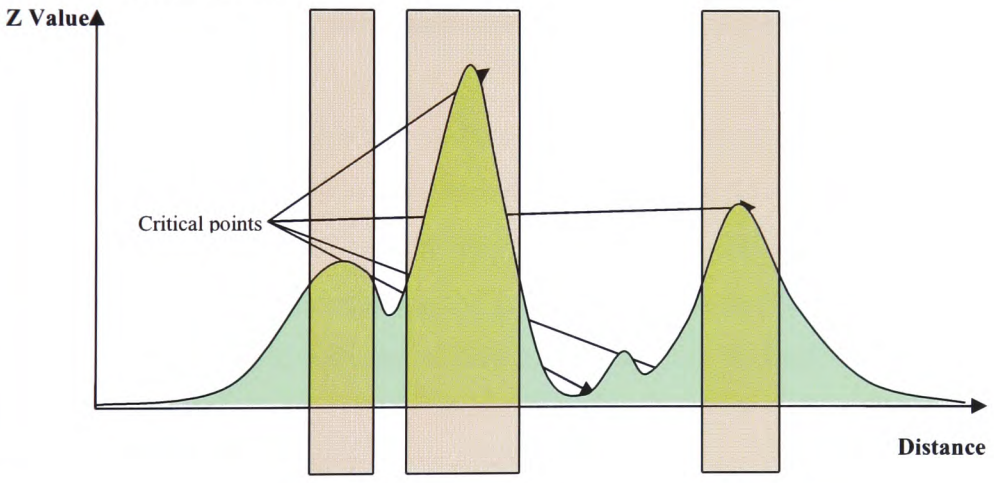
The first two techniques were rejected, as they require the user to study the properties of the data. Solution (1) also requires a highly skilled statistician to apply advanced tools such as MANOVA (Multi-dimension ANOVA) to

analyse multivariable data sets. Moreover, in order to use those techniques, the analysts have to make several assumptions about the data sets. The second approach is not applicable either as it transforms the distribution thus hides the "real" properties of the data.

The semi-parametric methods seem to be the best choice in this particular problem. Furthermore, those techniques, especially the Parzen Window, have already been applied in data rejection [Bishop, 1994; Taylor, 2000].

Once the density function has been adequately "learned", the threshold has to be found. In Figure 4.2, the function was symmetric around the mean and had only one global maximum (no local max) thus the acceptance interval was symmetric around the mean e.g. 68% acceptance equal [- one dev; + one dev]. However, the density distribution of the current data set is highly abnormal, as in Figure 4.3.

Figure 4.3: On the graph below, the red area contains between 70-90% of the data whereas the green one contains 99% of it.



From this observation, it is apparent that the interval I_c for $X\%$ acceptance will be a composite of several sub-intervals, it will have the shape $I_c = I_1 \cup I_2 \cup ... \cup I_n$. Finding the inclusion interval will require two steps:

1. Function Study (Variation, Critical points…)
2. Find the interval using knowledge of the function gained in (1).

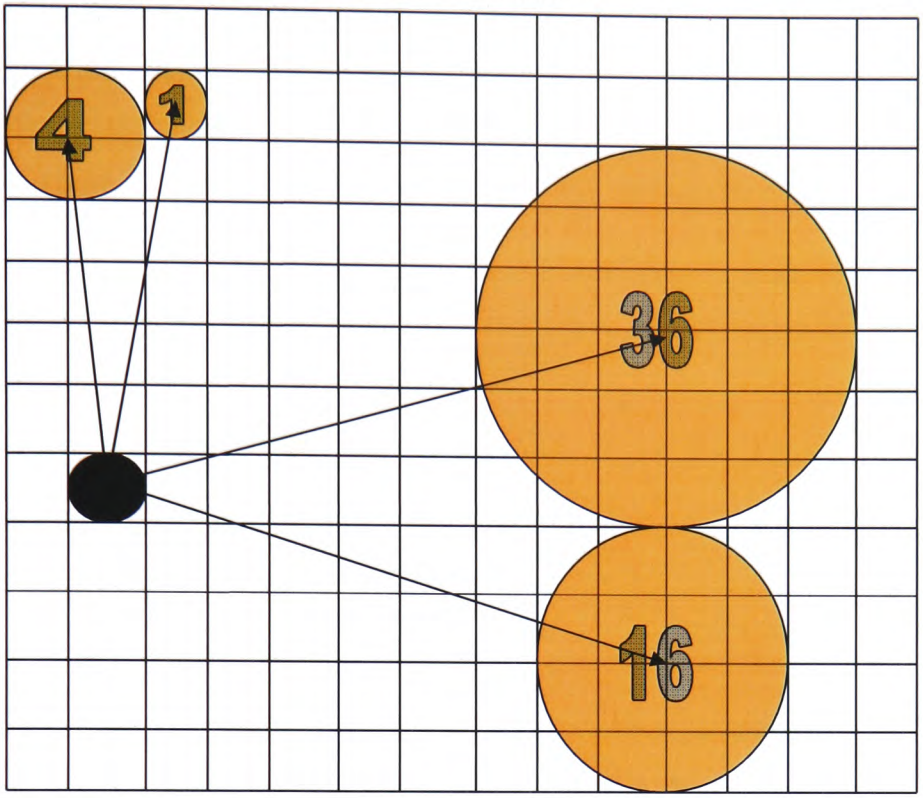
In low dimensional problems, variations and critical points (min, max, saddle points, local min, local max, and inflexion points) are found by differentiating

the functions. Then solving the derivative(s) finds the critical points (first derivative, second derivative...). Then, in order to find the composite interval I_c, the system has first to calculate the total surface below the function (Parzen Window guarantees that the function is convex). Finally, it calculates how much surface corresponds to X% inclusion. The composite interval I_c is calculated using a heuristic procedure.

The problems are numerous with a multi-dimensional vector space. Even though the function is differentiable, its complexity makes analytic resolution difficult or impossible. Furthermore, in order to calculate the surface below the curve, the function needs to have a primitive; the Parzen Window with Gauss's functions does not. Even if it had a primitive, the literature shows that integrating a function in a data space whose dimension is greater than three is difficult. This difficulty increases with the number of dimension, and the dimension of system space ranges from 5 to 62. To overcome those issues, the system has to reduce the dimension of the vector space. In the literature, the Kohonen map or SOM (Chapter 2.4) has been used to cluster the data [Taylor, 2000]. The main property of the technique is that it can detect clusters without any exterior interference.

The current study did not go down this road as using SOM for the data density function require some assumptions on the clusters or an extensive study of the clusters' properties. Then the user has to fix the threshold to merge, split or eliminate the clusters. Although the current technique does not use SOM to map the distribution, it uses the ability of the network to find the important clusters.

Figure 4.4: Example on how the distance is calculated



The relevance of the patterns in the data set is calculated with regard to their distance to the map clusters (Figure 4.4) using equation (1).

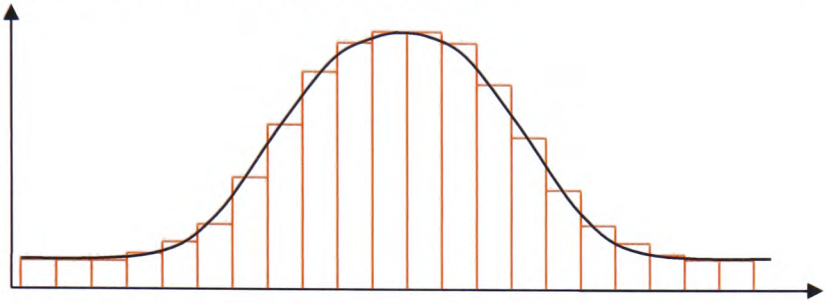
$$d_1 = \sum_{i=1}^{m} \left(n_i * \left\| c_i - x_j \right\| \right) \qquad (1)$$

where *"m"* is the number of clusters, *"n_i"* is the number of patterns in the cluster *"i"* and *"x_j"* is the input pattern. The combination of the Kohonen map and of equation (1) gives an estimation of the similarity of the current behaviour with the one in the training. The data set made up of these values maps the behaviour of the system in a one-dimension space. Although the space is one dimensional, the literature shows that it is difficult to solve the derivative of the Parzen Window density function as it is made up of the sum of many Gaussian kernels. Even a well-known mathematical package such as Maple 7 failed to solve this problem in an acceptable time.

There are heuristic techniques such as Newton's optimization method [Newton] to find the critical points but due to time limitation it was not possible to developing this line of research. Thus, the system uses graphical resolution to find the local maximum.

The remaining problem is to calculate the surface by solving the integral analytically. In many cases, it is impossible to integrate or it is too complex thus, mathematicians use "Numerical Integration". In numerical integration, the function validity range is split in small intervals. Then the integral value is approximated by calculating the surface of simpler shapes below the curve (Figure 4.5).

Figure 4.5: The normal distribution is approximated by the sum of the surface of smaller rectangles. In this case, it is the rectangle rule technique.



There are many different types of numerical integration algorithms, e.g. Figure 4.5 shows the rectangle rule. The trapezoidal rule uses a similar approach to the previous rule but uses trapezoids. Simpson's rule is similar but it also takes into account the interval's mid-point. Other more advanced approaches try to find the best number of intervals, vary their size or use more complex approximation shape. The heuristic algorithm uses those different techniques to search for the best inclusion surface.

The current heuristic search mechanism starts at the global maximum(s) and progressively increases the size of the interval. It only stops when the desired surface is included, with a predefined deviation. The pseudo-code below illustrates the search principle.

Limits search algorithm:

Comment: Initialize system
Input Critical Points
Input Maximum Interval of integration
Input Parzen's windows size
Calculate total integration surface, **S**

Comment: Training
Input distribution inclusion percentage, **p**
Calculate target surface **TgS** = **S*p**
Find tempo interval, **TmpI**, from global maximum (Critical Points) using initial increment value, **Inc.** The final inclusion interval is stored in **PermI**
Calculate tempo surface in tempo interval, **TmpS**
While **TmpS** different from **TgS**
  If **TmpS** equal **TgS** then
    TmpI become permanent surface, **PermI**
  Else
    If **TmpS** less than **TgS** then
      Include Critical Points higher inside **TmpI** (Tempo Interval)
      TmpI becomes **PermI**
        Find new **TmpI** using **PermI** and Inc
    Else
        Reduce **Inc** value
        Find new **TmpI** using **PermI** or global maximum
    End if
    Calculate new temporary surface, **TmpS**
  End if
End of while

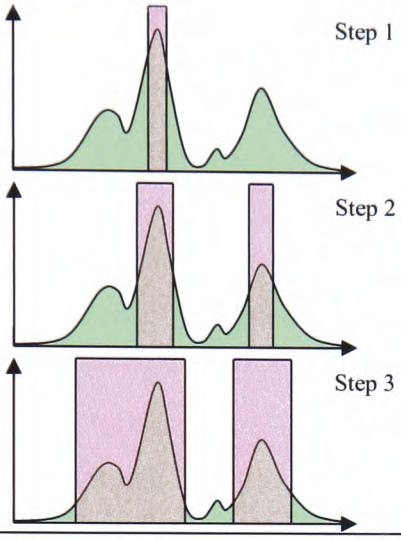
Comment: After the filter training
For each pattern in the file
  Pass pattern through Experts layer
  Scale Expert Output Vector with the Kohonen map
  If Result inside **PermI** then
    Identify it
  Else
    Reject pattern
  End if
End of for

Figure 4.6 illustrates the interval search process. Once the appropriate composite interval has been computed, the filter training is finished. Once the rejection filter has been trained, a pattern is presented to the pool of experts. Then the normality distance is calculated using the SOM grid learned during the training. Finally, the system checks if the value is inside the composite interval. If it is in the interval, it is accepted otherwise it is rejected as novel.

The advantage that the technique brings over others is that it does not require any assumptions on the novel data. The threshold choice is only linked to the level of certainty that the user desire. Thus, the choice of threshold is

linked to the application, e.g. in phytoplankton analysis, 75% acceptance may be good enough. On the other hand, 90% might be just enough for a medical application. (Figure 4.6 illustrates the progressive inclusion process, implemented by the heuristic procedure.

Figure 4.6: Evolution of the inclusion interval limits during the search process



As the search algorithm should take into account the likelihood of the behaviour while searching for the desired limits of inclusion, it has to start from the highest peak or peaks (global maximums). In theory, it should be possible to find those peaks automatically. However as explained earlier the current system uses a graphical resolution. The desired included surface corresponds to a percentage of the "total" surface below the function. The area included increases iteratively around the highest peak until the limits are as high as other peak(s). As illustrated in Figure 4.6, the included surface will then expand simultaneously around the included peak(s). By following this strategy, the algorithm makes sure that the search includes the most likely behaviour first. As the process continues, less likely behaviours are included. The process only stops when the surface limited by the inclusion intervals is equal to the desired portion of the total surface defined by the user. Chapter 6 presents some critical improvements that could be brought in, in order to improve the mathematical foundation of the current algorithm. Once again, those critical features were not implemented for time reasons. The following sections assess the usability of this approach.

4.4 Hardware and software environment

The following tests were conducted on a PC equipped with an Athlon Thunderbird 1 GHz, 512 MB of DDR RAM and an ATI Radeon 64 DDR graphic card. This software ran under Windows NT, and was created and compiled using Borland C++ Builder 4.

The Parzen Windows Size, the local maximums and the total interval of integration were visualized and selected using Maple 7.0 from Waterloo Maple Inc. First, the Parzen Window was drawn on a diagram using Maple then the min and max limits of integration were found graphically.

The SVM and RBF paradigms were the same as the ones used in the test on combination. The numerical integration techniques algorithms were found in the Numerical Recipes in C [1992]. The Kohonen Self-Organizing Map was created specifically for this project using C++ Builder 4.0. The numerical integration techniques and the Kohonen Self-Organizing Map were integrated to the test environment presented earlier.

4.5 Test data

Ten species were randomly selected out of a 62 species pool (Annexe 9.1.1), those species are typical of the northern European seas. They were grown in controlled conditions [Boddy, 2000] in order to obtain pure species, it guarantees that every "cell" is correctly labelled. The FC Parameters were extracted using the Becton Dickinson FACSort™ flow cytometer and the data then filtered to remove any unwanted noise. Finally, the parameters were linearly scaled such that every one of them had a mean of 0.0 and a standard deviation of 1.0.

4.6 Tests Procedure

Each test involved two distinct systems: One with SVM experts and the other with the RBF's. The experts were trained with 400 patterns per class during the training and tested with 400 different patterns per species in the testing stage. The Parzen Window weakness is its dependence on the number

of kernels and even more importantly on their size. Therefore the experiments assess three questions:

1. In test 1 and 2, the effect of the Parzen Window parameters is measured.
2. Test 3 tries the rejection ability of the novel architecture.
3. Test 3 also compares the rejection ability of the architecture with the RBF or SVM paradigms once adequate Parzen parameters are used.

None of the tests analysed the Kohonen map. It was setup using the Kohonen guideline (Annexe 9.3) and the grid was oversized to make sure that it would not limit the Kohonen search for clusters. The network trained over 10000 cycles, the learning speed was set to slow down at 2000 cycles (*Half-life*). The initial learning rate before "Half-life" was set to 0.9 and the minimum rate was set to 0.0. After Half-life, the learning rate varied from 0.02 to 0.01. The neighbourhood size decreased from 1 to 0 before Half-life and from 1 to 0.1 after. The decay function was a non-linear function of shape (2).

$$p = p_{init} * (\frac{p_{final}}{p_{init}})^{\frac{i}{n}} \quad (2)$$

The letter *"p"* is a generic letter that represents the learning rate or the neighbourhood size.

The initial value is *"p_{init}"*.

The minimum value is represented by *"p_{final}"*.

"n" is maximum number of cycle in a given stage.

"i" is the cycle counter.

Before and after half-life, the learning parameters initial and final values are different.

4.6.1 Test 1

This part of the experiment attempts to assess the importance of the kernel size on the Parzen function therefore on the rejection ability of the system. The number of kernels was set to 3000, which represents 75% of the training set. The system is made of ten experts. It only involves the Feature Base Decision strategies, in order to assess the joint rejection and combination

capacity of the HPNN architecture. The test set is made of 10 randomly selected species for the known species; those species were extracted from the usual data set (Section 4.5). The 52 unknown ones were made of the remaining species from the same data set. The dimension of the FBD strategies (Chapter 3) was set to 100. The distribution inclusion was set to 50% as an arbitrary choice. However, it had to be big enough to limit the influence of the weaknesses of the current limits search algorithm.

This test aims to estimate the influence of the window's size in the rejection process thus the window's size range was chosen to represent the transition from a highly noisy curve to a normal distribution. The range of the windows' sizes has to be appropriate for the distance range. This last case is unusable, as it would generate a normal distribution, which the current data do not follow.

RBF:

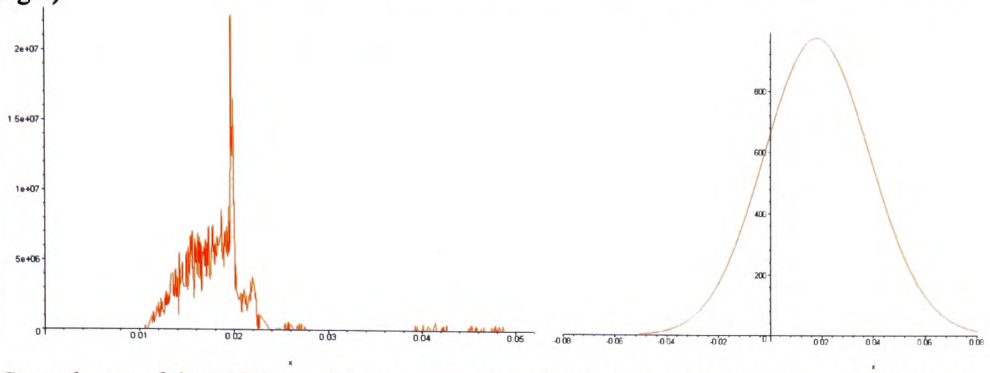
The RBF were made of 80 kernels (40 for target class/40 for background class), the distance is calculated using the Euclidean rule. The adapted K-means algorithm (Chapter 2.1) positions the RBF hidden kernels, the kernel activation function is exponential and finally the kernel width was set to 4.0.

The window's size sequentially takes the following values:

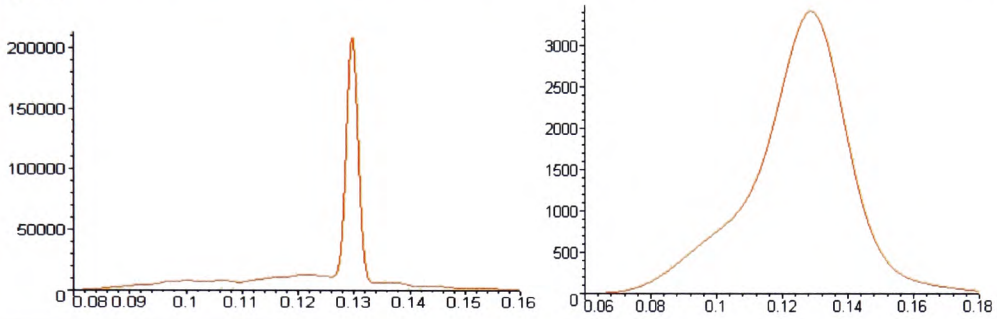
- 0.00002
- 0.00008
- 0.00014
- 0.00020
- 0.00060

The different window sizes were found empirically to be adequate for the given data set. Adequate means that the values represent the full spectrum of alteration of the curve smoothness from an extremely noisy function to a normal distribution (Figure 4.7).

Figure 4.7: Evolution of the density function smoothness with the increase of the Parzen window size.
Smoothness of the RBF density function with w = 0.00002 (On the left) and w= 0.006 (On the right)



Smoothness of the SVM density function with w=0.001 (On the left) and w= 0.009 (On the right)



SVM:

The SVM experts used Gaussian Kernels; allowing the SVM experts to find hyper-curve although the resolution algorithm in the feature space is still based on linear principles [Vapnik, 1995; Gunn, 1998; Burges, 1998]. The penalty term was set to 1000 to force good separation of the classes and the mis-identification threshold was set to 0.9.As for the RBF tests, the window sizes was set to those five different values as they represent the smoothness variation of the curve with different window sizes. They are different from the one in RBF tests, the data set that is used is made of vectors that represents the output of the experts for a given pattern. The SVM is different from the RBF thus the characteristic of the behaviour data set is different (Section 4.3). Therefore, the Parzen Window sizes have to be different.

The following values were found empirically to be adequate:

- 0.001
- 0.003
- 0.005
- 0.007
- 0.009

4.6.2 Test 2

This test targets another side of the system that is "How important is the number of kernels in the Parzen system?" The test data is made of the same 10 experts that were trained in test 1. The unknown data are once again made of the remaining 52 species. This time, the window size is fixed. The value was found empirically during the test session number 1. The number of kernels takes three different values: 1000, 2000 and 3000. Once again, the FBD dimension is set to 100 and the inclusion threshold to 50%.

RBF:

The RBF experts are the same as the one in Test 1 but the Parzen window's size was set to 0.0002.

SVM:

Similarly, the SVM experts in this test are the same as those in Test 1 and the window's size is 0.002.

4.6.3 Test 3

In this test, the experiments investigate the relation between the inclusion threshold, the percentage of rejection of unknown and the percentage of correct identification. The number of experts is 10 and the Parzen function is made of 75% of the training data set, i.e. 3000. Once again, the FBD dimension is 100. The two systems were then tested with several percentage of inclusion, ranging from 20% to 95%. The RBF and SVM parameters are set to the same values as in test 1 and 2.

4.7 Tests Analysis:

4.7.1 Test 1:

RBF (Table 4.1)

When no acceptance threshold is set, the PFBD system identifies 82.4% of the known patterns correctly with a deviation of ≈9 percent. Out of the remaining data, 11.4 % are rejected. Simultaneously 60% of the unknowns were discarded. However, the rejection deviation is ≈ 31 percent.

The picture is slightly different with the FFBD, 89.75% of the known patterns are rejected and 1.18% of the remaining ones are rejected. Only 10% of the unidentified patterns are rejected.

The effect of the windows size is dramatic e.g. with a window size of 0.00002, the curve is unusable as the curve passes by all the Parzen window points.

With a size of 0.0002 or 0.002, the results are similar although the second has an advantage of roughly 7% on the known patterns' identification. The error on the known patterns and the rejection of unknown on both FBD systems and with the two windows size is similar.

The last window size is clearly inappropriate; the results are unrelated to the acceptance threshold. It is important to stress that the system with rejection produce ≈5 percent less error than the one without.

Table 4.1: The HPNN and Radial Basis Function system

PFBD	No Rej			w 0.00002		
	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back
Average	82.40	11.40	60.40			
Std Dev	9.31	7.81	31.01			
FFBD	**No Rej**			**w 0.00002**		
	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back
Average	89.75	1.68	9.96			
Std Dev	3.56	1.11	5.82			

PFBD	w 0.0002			w 0.002			w 0.02		
	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back
Average	46.50	51.60	87.76	53.63	44.60	87.36	79.88	13.93	61.27
Std Dev	19.16	19.13	13.78	20.29	20.20	14.44	10.98	9.64	30.19
FFBD	**w 0.0002**			**w 0.002**			**w 0.02**		
	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back
Average	46.83	51.58	87.56	54.00	44.48	86.84	87.23	4.20	10.83
Std Dev	19.12	18.94	13.62	20.11	19.92	14.43	8.74	8.26	6.00

The table displays the variation of the identification of the known species (Targ), the percentage of those species patterns that were rejected (Rej Targ) and finally the percentage of unknown that were successfully discarded. The standard deviation measures the variation around the average identification and rejection for the individual species. It also shows the difference of performance between the Partial Feature Based Combination strategy and Full Feature Based Combination strategy when those strategies are combined with the rejection filter.

SVM (Table 4.2)

The SVM system identifies 95% of the data correctly when the acceptance threshold is set to 100%. The identification performances change from 5% to 32% as the window grows from 0.001 to 0.009. Simultaneously the unknown's rejection decreases by 10% with the PFBD and by 20% with the FFBD. As for the RBF, the sum rejection/identification is around 95%, which is good.

Table 4.2: The HPNN and SVM system

PFBD	No Rejection			w 0.001			w 0.003		
	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back
Average	95.75	2.05	29.47	5.58	94.35	96.91	12.10	87.63	92.30
Std Dev	3.21	1.47	32.45	3.97	4.02	3.87	8.18	8.34	9.24
FFBD	No Rejection			w 0.001			w 0.003		
	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back
Average	95.35	0.65	5.76	5.53	94.33	93.72	12.15	87.43	85.00
Std Dev	2.45	0.41	3.25	3.93	3.95	8.28	8.20	8.30	18.22

PFBD	w 0.005			w 0.007			w 0.009		
	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back
Average	22.28	77.25	87.12	25.35	74.18	85.73	32.93	66.35	82.29
Std Dev	16.68	16.56	16.87	16.50	16.46	16.08	22.17	22.09	20.60
FFBD	w 0.005			w 0.007			w 0.009		
	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back
Average	22.13	77.10	78.64	25.23	73.95	74.63	32.68	66.13	70.69
Std Dev	16.41	16.42	23.29	16.40	16.41	26.29	21.96	21.91	28.83

The table displays the variation of the identification of the known species (Targ), the percentage of those species patterns that were rejected (Rej Targ) and finally the percentage of unknown that were successfully discarded. The standard deviation measures the variation around the average identification and rejection for the individual species. It also shows the difference of performance between the Partial Feature Based Combination strategy and Full Feature Based Combination strategy when those strategies are combined with the rejection filter.

4.7.2 Test 2

In this case, the windows size is fixed (RBF: 0.0002 SVM: 0.002) and the number of points in the sample varied.

RBF (Table 4.3)

The changes in sample size affect the identification by about 2% at around 46% for both FBD strategies. The rejection of unknown is also affected, with a variation of 2%. The average value is around 88%. This time, the sum identification/rejection performance is still around 98%.

Test 4.3: RBF

PFBD	No Rejection			1000 Points			2000 Points			3000 Points		
Species	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back
Average	82.40	11.40	60.40	45.95	52.85	91.01	46.03	52.13	88.01	48.35	50.48	89.90
Std Dev	9.31	7.81	31.01	20.01	19.98	11.18	19.35	19.28	13.58	17.41	17.11	11.92
FFBD	No Rejection			1000 Points			2000 Points			3000 Points		
Species	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back
Average	89.75	1.68	9.96	46.30	52.70	90.58	46.35	52.10	87.83	48.83	50.20	89.33
Std Dev	3.56	1.11	5.82	19.83	19.75	11.24	19.31	19.09	13.43	17.21	16.88	12.01

This table shows the evolution of the identification success on the known species (Targ), the rejection of known species (Rej Targ) and the successful rejection of the unknown (Rej Back) evolve as the number of kernels increases in the Parzen Window density function

SVM (Table 4.4)

With the variation of sample size, the identification percentage changes from 29 to 38%; on the other hand, the rejection performances are stable at 80% for the PFBD and 69% for the FFBD. As for the other tests, identification/rejection still gives 92%.

Test 4.4: SVM

PFBD	No Rejection			1000 Points			2000 Points			3000 Points		
	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back
Average	95.75	2.05	29.47	29.30	70.15	80.45	29.30	70.15	80.45	38.55	60.65	82.04
Std Dev	3.21	1.47	32.45	13.64	13.55	19.39	13.64	13.55	19.39	26.27	26.10	21.07
FFBD	No Rejection			1000 Points			2000 Points			3000 Points		
	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back	Targ	Rej Targ	Rej Back
Average	95.35	0.65	5.76	29.25	69.68	68.69	29.25	69.68	68.69	38.23	60.48	71.94
Std Dev	2.45	0.41	3.25	13.61	13.50	25.66	13.61	13.50	25.66	25.93	25.93	26.63

This table shows the evolution of the identification success on the known species (Targ), the rejection of known species (Rej Targ) and the successful rejection of the unknown (Rej Back) evolve as the number of kernels increases in the Parzen Window density function

4.7.3 Test 3

In this stage, the tests from the previous stage are combined to obtain a more optimised system. The following tests analyse the relation between the distribution threshold and the rejection ability of the system.

RBF (Table 4.5)

The link between the rejection of known patterns and the inclusion threshold is apparent for both the PFBD and FFBD strategies. With the PFBD, the unknowns' rejection decreases from 97% to 61.79%. Although, there is a similar trend with the FFBD, the effect of the threshold is not as beneficial; the rejection percentage varies from 97% to 21% as the inclusion percentage increases.

Once more, the sum of the Rejection/Identification reduces the error when compared to the system without rejection.

The gap between the predicted acceptance and the real one widens as the acceptance threshold increases with both combination strategies.

Test 4.5: RBF

	Acceptance Threshold	20%				35%				50%			
		Targ	Rej Targ	Err on Targ[1]	Rej Back	Targ	Rej Targ	Err on Targ[1]	Rej Back	Targ	Rej Targ	Err on Targ[1]	Rej Back
PFBD	**Mean**	20.78	78.93	0.29	97.20	35.73	63.05	1.22	92.83	49.38	49.45	1.17	89.50
	Diff T/R[2]	0.78				0.73				-0.62			
	StdD	13.82	13.83		5.33	18.36	18.25		9.18	17.59	17.28		12.25
FFBD	**Mean**	20.88	78.85	0.27	96.99	35.98	63.03	1.89	92.70	49.83	49.18	0.99	88.92
	Diff T/R[2]	0.88				0.98				-0.17			
	StdD	13.77	13.77		5.34	18.26	18.12		9.15	17.41	17.04		12.34

	Acceptance Threshold	65%				80%				95%			
		Targ	Rej Targ	Err on Targ[1]	Rej Back	Targ	Rej Targ	Err on Targ[1]	Rej Back	Targ	Rej Targ	Err on Targ[1]	Rej Back
PFBD	**Mean**	61.73	35.53	2.74	81.62	73.65	21.50	4.85	71.13	79.03	14.78	6.19	61.79
	Diff T/R[2]	-3.27				-6.35				-15.97			
	StdD	17.97	17.62		18.31	14.35	12.93		25.06	12.32	11.14		29.78
FFBD	**Mean**	62.48	35.23	2.29	80.82	75.35	20.45	4.2	68.20	86.00	5.65	8.35	20.94
	Diff T/R[2]	-2.52				-4.65				-9			
	StdD	17.59	17.12		18.23	13.66	12.51		25.08	10.69	10.30		16.07[34]

This table displays the percentage of known species that were correctly identified (Targ). The number of known species that were rejected (Rej Targ). The error on the target (Err on Targ) species which is the % of patterns that remains after the identification and rejection. The percentage of unknown that were rejected and the difference between the theoretical number of known patterns and the real number of patterns that should have been rejected (Diff T/R). It also shows the standard variation from the average value, this value indicate if the system did consistently well or not

[3] *Err on Targ = Error on the known species = 100% -% of known patterns successfully classified (Targ) - % of known species rejected (Rej Targ)*

[4] Diff T/R = It is the difference between the theoretical number of known patterns that should have been rejected and the real number of rejection

SVM (Table 4.6)

The rejection percentage decreases from 94% to 37% for the unknown patterns as the inclusion threshold varies from 20% to 95% with the PFBD strategy. Simultaneously the identification percentage increases from around 10 % to 89%. With the FFBD, the trade-off is similar as the identification performance increases from 10% to 89% as the inclusion percentage increases. The rejection of unknown follows an opposite path as it decreases from 89% to 17%. The analysis of the raw data set showed that the existing system rounds the output value at three digits.

The difference between prediction and real acceptance does not seem to be linearly related to the acceptance threshold.

Test 4.6: SVM

	Acceptance Threshold	20%				35%				50%			
		Targ	Rej Targ	Err on Targ[1]	Rej Back	Targ	Rej Targ	Err on Targ[1]	Rej Back	Targ	Rej Targ	Err on Targ[1]	Rej Back
PFBD	Mean	9.88	89.90	0.22	94.17	22.15	77.48	0.97	88.71	32.68	66.60	1.02	83.28
	Diff T/R[2]	-10.12				-12.85				-17.32			
	StdD	6.63	6.72		7.19	14.77	14.70		13.71	21.85	21.76		19.51
FFBD	Mean	9.90	89.73	0.37	88.34	22.00	77.30	0.7	79.00	32.48	66.35	1.17	72.24
	Diff T/R[2]	-10.10				-13				-17.52			
	StdD	6.61	6.64		14.77	14.53	14.54	[5]	23.94	21.56	21.59		27.56

	Acceptance Threshold	65%				80%				95%			
		Targ	Rej Targ	Err on Targ[1]	Rej Back	Targ	Rej Targ	Err on Targ[1]	Rej Back	Targ	Rej Targ	Err on Targ[1]	Rej Back
PFBD	Mean	40.55	58.68	0.77	74.27	66.00	32.30	1.7	60.13	89.53	8.38	2.09	37.82
	Diff T/R[2]	-24.45				-14				-5.47			
	StdD	22.67	22.66		26.42	25.64	25.42		31.28	10.29	10.38		32.60
FFBD	Mean	40.40	58.23	1.37	60.32	65.95	31.20	2.85	44.66	89.28	7.08	3.64	15.75
	Diff T/R[2]	-24.6				-14.05				-5.72			
	StdD	22.72	22.75		31.34	25.23	25.18		30.58	10.29	10.21		15.92[6]

This table displays the percentage of known species that were correctly identified (Targ). The number of known species that were rejected (Rej Targ). The error on the target (Err on Targ) species which is the % of patterns that remains after the identification and rejection. The percentage of unknown that were rejected and the difference between the theoretical number of known patterns and the real number of patterns that should have been rejected (Diff T/R). It also shows the standard variation from the average value, this value indicate if the system did consistently well or not.

[5] *Err on Targ = Error on the known species = 100% -% of known patterns successfully classified (Targ) - % of known species rejected (Rej Targ)*
[6] *Diff T/R = It is the difference between the theoretical number of known patterns that should have been rejected and the real number of rejection*

4.7.4 Analysis

The SVM showed its identification superiority over the classical RBF. However, it also showed its limitations in this research, in fact the SVM system is not able to reject the data from the unknown classes by itself (Test 1). The RBF's main advantage in this domain is that it does not find a separating hyperplane as the SVM (Chapter 2.2). Instead, it approximates the space in which the target and background classes are contained. Thus, the output is an estimation of the position of a pattern with regard to the target class spatial coordinates.

When the experts are combined using the FBD strategies the picture is the same as in (chapter 3). Tests 1 and 2 prove the dependence of the rejection strategy on the Parzen parameters. The diagrams illustrate the importance of the window size on the curve smoothness, i.e. too much noise gives an unusable function and too little tend to hide the real distribution. However, it also showed that when an adequate setting is chosen, the combination of Kohonen Map and Parzen Window works well. Test 3 showed that with a proper parameter configuration, rejecting the data based on the pool behaviour is quiet efficient. There is however a major difference between the RBF and the SVM systems. In fact, the RBF's system identification performance follows better the acceptance threshold variation than the SVM's. It might mean that the system approximates its behaviour distribution better. Once again it is certainly linked to the difference between the identification strategies. Moreover, when the rejection architecture is coupled with an RBF system, it is able to reject more unknowns than the SVM system while accepting more known patterns.

The difference between the FBD strategies is negligible as the rejection stage happens before the identification thus the advantage of one strategy over the other fades away (Test 3).

Although the RBF system does better than the SVM one, especially when the acceptance threshold is low, the SVM system produces less error on the target class (Test 3). The reason being that the SVM identification algorithm is extremely powerful thus if the data are not rejected it will produce

less errors than the RBF one. The SVM performs exceptionally well considering that its behaviour analysis is affected by a rounding error that probably hides the real distribution; the RBF performed better in this sequence of tests. In fact, it is able to take full advantage of the Rejection scheme. When combined with it, it more than makes up for its identification performance with its rejection ability.

After the tests, it is also apparent that the rejection strategy discards efficiently the unknown data based on the acceptance threshold but also that it improves the reliability of the system on the known species. In Test 3, the error on the known species decreases with the acceptance threshold. Thus, a high rejection threshold (Low Acceptance Threshold) will correspond to casting off a lot of unknown and simultaneously will improve the reliability of the known patterns identification.

The expected drawback of the current implementation is that the system does not reject all the unknown species with the same consistency (Table 4.7). It does not consistently reject the known species either. The reason is that not all of the species have the same degree of similarity. As some species are more similar than others are, they will not be identified with the same success rate. Similarly, some of the unknown species are more different from the others. Thus, they will be discarded more successfully. It is an unavoidable problem as the unknowns will always be unknown thus it is impossible to say how the system will deal with them. However, by reducing the acceptance threshold, the confidence that the data are properly identified increases, as test 3 showed. The RBF results in Test 3 show that as the acceptance threshold increases so does the difference between the predicted result and the real ones. It is certainly linked to the current implementation of the limit search algorithm (Chapter 6).

Table 4.7: Error and Rejection numbers when there is no combination or filtering technique.

The greyed are represent the number of identification error on the known species. The other part represents the number of times the unknown species were wrongly identified as known species when they should have been rejected.

Species	RBF		SVM	
	% Class	Nb Error	% Class	Nb Error
0	94	14	98	9
1	89	16	98	7
2	88	67	97	6
3	93	21	99	4
4	75	56	89	32
5	69	47	99	8
6	86	43	97	10
7	82	31	97	14
8	84	57	95	15
9	70	25	95	11
10		242		461
11		216		404
12		313		414
13		84		365
14		159		452
15		335		335
16		0		113
17		7		346
18		200		372
19		258		426
20		118		361
21		272		408
22		0		2
23		3		6
24		12		11
25		1		5
26		2		3
27		322		406
28		263		378
29		375		393
30		344		286
31		49		85
32		298		414

	Species	RBF		SVM	
		% Class	Nb Error	% Class	Nb Error
	33		347		400
	34		359		414
	35		266		367
	36		376		369
	37		15		493
	38		287		299
	39		0		428
	40		9		377
	41		59		350
	42		207		423
	43		176		441
	44		325		401
	45		84		418
	46		159		396
	47		18		360
	48		15		16
	49		263		376
	50		80		415
	51		288		457
	52		295		367
	53		290		388
	54		27		35
	55		1		3
	56		149		414
	57		356		359
	58		170		580
	59		234		369
	60		15		390
	61		36		636
Target	Mean	83	38	96	12
	StdD	9	19	3	8
Unknown	Mean		168		280
	StdD		137		173

4.8 Conclusion

This chapter tested the rejection method while comparing two neural network paradigms. The new method has the advantage of making use of the HPNN architecture (Chapter 3) and of two robust neural network architectures, which is better technique than using new unproven ANN methods. Moreover because it is based on the distribution curve, its mathematical basis is more robust and user friendly than those tried in previous studies. The preliminary tests demonstrate that it is possible with the HPNN to reject efficiently and consistently unknown species based on the behaviour of the pool of experts. The tests showed that the Radial Basis Function is especially suitable to this approach.

Some unexpected pitfalls in the implementation of the technique were discovered:

- Limits of the classical integration technique, .e.g. finding primitives
- Finding the critical points for the complex mathematical functions

After having brought to light those issues, a viable solution to these issues is offered with a Kohonen map in order to reduce the dimension of the behaviour space. It showed that numerical integration is a solution to the integration problem. This technique proves that despite its simplicity, it achieves a respectable and consistent level of success. Despite those positive results, the preliminary tests highlight the need for further studies in order to try a better distribution approximation technique than the Parzen Window. In fact, the tests show that this technique is highly dependant on the number of Gaussian kernels and on the window size. Thus, it would be preferable to use a more advanced method such as the General Mixture of Experts.

Finally, the tests confirm that the limits algorithm needs to be improved to take more account of the distribution shape. Further experiments should attempt to improve on what was already done but also to test the architecture on bigger problems and with other neural network paradigms.

Chapter 5: Conclusion

5.1 Conclusion

In order to achieve an extensive study of the biosphere, new automated tools were needed. Although the Flow Cytometer (FC) is a fast and reliable information extractor, it does not solve the analysis problem. The huge quantity of information generated by the FC still requires pain staking human analysis before any interpretation can be done. Several techniques including artificial neural networks (ANNs) have been successfully applied in the past to transfer the analysis burden to an automated system. However none of those approaches successfully satisfied all the criteria that were required in the ideal automated analysis tool. The main weaknesses of the classical ANN were that:

- The decision path decision path is difficult to follow.
- ANN cannot grow without retraining the system, retraining looses the existing knowledge.
- It cannot efficiently reject the novel data.

This research aimed at creating an innovative architecture that would combine the classical identification ability of the ANN paradigms with the capacity to grow without retraiming. It also intended to integrate the capacity to reject novel data, which is an important ability in phytoplankton analysis, as the species domain is unbounded, due to many "unknown" species of phytoplankton. These species can be new species or species on which the system was not trained. In a biological study, the analyst focuses on a limited number of target species, thus the other species are discarded as unknown/novel; this is something that the common ANN paradigms cannot do. Moreover, as this research targets users that are not ANN specialists, the system had to be simple to interpret and understand.

In order to achieve this goal, this research developed two innovative and complementary architectures based on the highly partitioned neural network paradigm (HPNN). The HPNN divides a multi-class problem into smaller two class ones. A mechanism is required to merge those smaller problems in order to offer the same characteristics as the original multi-class one.

This research led to several important contributions to the applications of ANNs in FC data analysis:

1) A new combination architecture called HPNN/FBD was developed. In fact, the tests (Tables 3.1 and 3.2 from Chapter 3) showed that the simple WTA strategy works on small problems. However, further tests (Tables 3.5 and 3.6) highlighted that this simple technique was not appropriate for larger, growing problems. The FBD technique (Chapter 3, Section 3) was demonstrated in a thorough sequence of tests (Tables 3.5 and 3.6) to be applicable to large problems. These tests also showed that the FBD architecture was suitable for growing problems.

2) The same sequences of tests (Tables 3.1 to 3.6) demonstrate that the performances of the HPNN/FBD with RBF were on par with MRBF. But that the HPNN/FBD with SVM is superior to the other techniques (Monolithic RBF or HPNN/FBD and RBF).

3) Chapter 4 (Sections 2 and 3) presented a new strategy that allows detection and rejection of novel data. Its advantage over other analysis techniques is that it does not make any assumptions on the data distribution or on the novel data. The threshold value is not empirically found based on tests on "pseudo novel data" but on the expert knowledge the user has of the problem, i.e. a biologist does not only know how critical the results are, but also has a clear idea of the problem domain. Thus, the threshold is based solely on the knowledge that is available in every experiment. The tests (Tables 4.1 to 4.6) showed that this new architecture was a possible answer to the novelty detection and rejection question.

4) From the tests on the rejection architecture (Tables 4.5 and 4.6, Chapter 4), it seems that the RBF is more appropriate for novelty detection than the SVM.

5.2 Future work

Although, the goals of this research have been achieved, several possible improvements could be applied to the new architecture:

1) The RBF came out as the overall winner in this project as it offers the best compromise between the classification and the rejection abilities (Chapter 3 and 4). However, the classical RBF showed that it struggles to learn the complexity of the sub-tasks. Moreover in order to learn this task, it requires a certain level of testing in order to adjust the learning parameters. As a potential user of the system would be a non-specialist, removing the need to tune parameters should be of high priority. Furthermore, automating parameter tuning would improve the reproducibility of the test, as the neural networks would always be tuned with the same method. In Chapter 2, Section 1, several automated tuning techniques were presented. The technique in [Orr, 2000] looks more promising as it uses a tree to automatically find the size of the RBF and to tune the parameters. Moreover, once the network size and parameters have been chosen, the learning remains linear.

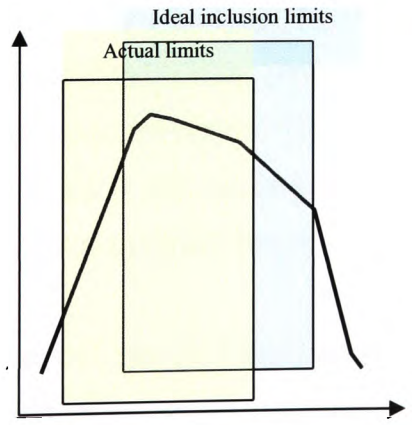
2) The SVM was the best learning paradigm of the two as it consistently learnt any task that was given to it (Tables 3.1 to 3.6) but it was much slower that the RBFs (Table 3.8). However, the SVM model used in this project in no longer the state-of-the-art as Joachims Thorsten has developed a new version of his software [SVM Light]. Moreover, several other SVM simulators exist. Those new SVM simulators implement new mathematical methods to speed up the learning process. Then, the SVM did not reject the unknowns well in the experiment presented in Chapter 4. Although it is reasonable to think that the rounding problem identified late in this research limited the SVM ability, new SVM models [Gammerman, 1998; Saunders, 1999] that

attempt to build a confidence estimator in the SVM learning paradigm could improve even further its rejection ability.

3) The current combination has successfully achieves its role but it relies on histograms. The accuracy of which relies on the number of intervals (bins) that makes it. It would be interesting to investigate methods to automate the choice of the number of intervals or to use alternative methods to map the behaviour. For example, the semi-parametric methods such as the Parzen Windows could be suitable to replace the histogram.

4) The current research rejection technique uses the Parzen window technique in order to approximate the density distribution. Tables 4.1 to 4.4 showed that the Parzen window relies heavily on parameters in order to limit the smoothness of the function. It would be interesting to try more advanced methods such as the Gaussian Mixture of Experts as those methods do not rely on empirical tests to choose the parameters. Moreover, due to time limitation, it has not been possible to develop an automatic method to find the local maximums. This line of research should be a priority, as it would dramatically improve the usability of the architecture. Finally, the interval search algorithm at this stage contains a weakness that seriously needs to be addressed (Figure 6.1).

Figure 6.1: The current limits problem



As illustrated in (Figure 6.1), the current implementation of the search algorithm does not take into account the asymmetry of the curve. That

is, in Figure 6.1, the slope on the left of the local maximum is very different from the one on the right. The data on the right are more likely than those on the left. Thus the data on the right should be included first. The current version of the algorithm is not able to take this shape information into account. In order to use this information, the limit increment should take into account the shape of the curve. In order to implement this more advanced technique, the system would require information such as inflection, saddle points and local minimum. The development of a search algorithm that would take into account this information would improve further the mathematical robustness of the algorithm. It would also be interesting to test the architecture on a larger range of problems.

5) Finally, this research did not attempt to investigate techniques that explain the ANN decision path or estimate the identification confidence. Those two issues should be investigated as they would allow a wider spread of ANN use in common applications.

5.3 Final remarks

The aim of this research was to create a new ANN architecture that would be better suited for the needs of microbiologists. It led to the development of a new type of ANN architecture that should allow a wider application of ANN for phytoplankton analysis. For the first time, a neural network architecture offers good classification ability while allowing a none-specialist to add or remove species to the system. This ANN did not only give a more flexible classification system, it also resulted in a system that can reject unknown species without requiring empirical test in order to find the rejection criteria.

The combination of those two powerful features should make it easier to use artificial neural network not only on field experiment in microbiology but also in a wider range of real life applications.

Chapter 6: References

[Anand, 1995]
R. Anand, K. Mehrotra, C.K. Mohan, S. Ranka. Efficient Classification for Multiclass Problems Using Modular Neural Networks. IEEE Transaction on Neural Networks. Vol 6. p 117-124. 1995.

[Aupetit, 2000]
M. Aupetit, P. Couturier. A "Recruiting Neural Gas" for function approximation. In Proceedings of International Joint Conference on Neural Networks, Como, Italy, 2000.

[Autret, 2002]
A. Autret, C.W. Morris, P. Angel. 2002
The HPNN solution: adding new classes without retraining
Submitted to Neural Network Applications

[Balfoort, 1992]
H. W. Balfoort, J. Snoek, J. R. M. Smits, L.W. Breedveld, J. W. Hofstraat, J. Ringelberg. Automatic identification of algae: neural network analysis of flow cytometric data. In J. Plankton res. Vol 14. p 575-589. 1992.

[Beroughit, 2002]
N. Benoudjit, C. Archambeau, A. Lendasse, J. Lee, M. Verleysen. Width optimization of the Gaussian kernels in Radial Basis Function Networks. In Proceedings of European Symposium on Artificial Neural Networks, d-side. p 425-432. 2002.

[Bishop, 1994]
C.M. Bishop. Neural Networks for Pattern Recognition. Oxford Press. 1995.

[Blanzieri, 1998]
E. Blanzieri. Learning Algorithms for Radial Basis Function Networks: Synthesis, Experiments and Cognitive Modelling. Thesis. University and Polytechnic of Turin. 1998.

[Boddy, 1994]
L. Boddy, C. W. Morris, M. F. Wilkins. Neural network analysis of flow cytometric data for 40 marine phytoplankton species. In Cytometry. Vol 15. p 283-293. 1994.

[Boddy, 2000]
L. Boddy, C.W. Morris, M.F. Wilkins, L. Al-Haddad, G.A. Tarran, R.R. Jonker, P.H. Burkill. Identification of 72 phytoplankton species by radial basis function neural network analysis of flow cytometric data. In Marine ecology-Progress series. Inter-reseach, Oldendorf luhe. Vol 195. p 47-59. 2000.

[Breiman, 1994]
L. Breiman. Bagging Predictors. Machine Learning. Vol 24. p123-140. 1994.

[Burges, 1998]
C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. Journal of Data mining and Knowledge recovery. Kluwer Academic Publishers. Vol 22. p 161-167. 1998.

[Campbell, 1999]
C. Campbell, N. Cristianini. Simple Learning Algorithms for Training Support Vector Machines. Technical Report. Dept. of Engineering Mathematics. University of. Bristol, 1999.

[Chen, 1999]
K. Chen, H. Chi. A modular neural network architecture for pattern classification based on different feature sets. In International Journal of Neural Systems. World Scientific Publishing Company. Vol 9. p 563-581.

[Cortes, 1995]
C. Cortes. Prediction of Generalization Ability in Learning Machines. Department of Computer Sciences, University of Rochester and N. York. 1995.

[Dayhoff, 1990]
J. Dayhoff. Neural Network Architectures, An introduction. Van Nostrand Reinhold. New York. Chapter 1. 1990.

[Frankel, 1989]
R. J. Olson, S. L. Frankel, S. W. Chisholm. Use of neural net computer system for of flow cytometry data of phytoplankton populations. In Cytometry. Vol 10. p 540-550.. 1989.

[Gammerman, 1998]
A. Gammerman, V.Vovk, V.Vapnik. Learning by Transduction. Uncertainty in Artificial Intelligence, Procs of the Fourteenth Conference. Morgan Kaufmann. 1998.

[Gill, 1991]
Philipp E Gill, Walter Murray, Margaret H Wright. Numerical linear algebra and optimisation. Addison-Wesley Publishing Company, Redwood city, Calif. Vol 1. p317. 1991.

[Gunn, 1998]
S. Gunn. Support Vector Machines for Classification and Regression. Technical Report. University of Southampton. ISIS. 10 May 1998.

[Hansen, 2000]
J.V Hansen, A Krogh. A General Method for Combining Predictors Tested on Protein Secondary Structure Prediction. In ANNIMAB-1, 2000.

[Hashem, 1997]
S. Hashem. Optimal linear combinations of neural networks. In Neural Networks. Vol 10. p 599-614. 1997.

[Hornik, 1989]
K. Hornik, M. Stinchcombe, H. White. Multilayer feed-forward networks are universal Approximators. In Neural Networks, 1989. Chapter 2. 9 359-366. 1989

[Jacobs, 1991]
R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton. Adaptive Mixture of Local Experts. In Neural Computation. MIT Press, Massachussetts, p 79-87. 1991.

[Japkowicz, 1995]
N. Japkowicz, C. Myers, M. Gluck. A novelty detection approach to classification. In IJCAI. p 518-523. 1995.

[Jordon, 1994]
M.I. Jordan, R.A. Jacobs. Hierarchical mixtures of experts and EM algorithm. In Neural Computation. Vol 6. p 181-214. 1994.

[K mean]
Chapter: Non-hierarchical clustering.
http://www.mathworks.com/access/helpdesk/help/toolbox/stats/multiv16.shtml
21 April 2003

[Kohonen, 2001]
T. Kohonen. Self-Organizing Maps. Springer-Verlag, New York. Second Edition. 2001.

[Kuhn-Tucker conditions].
Power Learn – Iowa State University
http://www.chass.utoronto.ca/~osborne/MathTutorial/KTCF.HTM
21 April 2003

[Kurimo, 1994]
M. Kurimo. Application of Learning Vector Quantization and Self-Organising Maps for training continuous density and semi-continuous Markov models. Technical Report. Helsinki University Of Technology. February 1, 1994.

[Lu, 1998]
B-L. Lu, M. Ito. Task Decomposition and Modular Combination Base on Class Relations: A Modular Neural Network for Pattern Classification. IEEE-Neural Network. Vol 10:5. p 1244. 1998.

[McCulloch, 1943]
W.S. McCulloch, W. Pitts, A Logical Calculus of the Ideas Immanent in Nervous Activity, Bull. Math. Biophysics. Vol. 5. p 113-133. 1943.

[Marsland, 2000]
S. Marsland, U. Nehmzow, J. Shapiro. Detecting novel features of an environment using habituation. In From Animals to animats: Proceedings of the 6th international conference on simulation of adaptive behaviour, SAB 2000. MIT Press. p 189-198. 2000.

[Masulli, 2000]
F. Massuli, G. Valenti. Comparing Decomposition Methos for Classification. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies KES'2000. R.J. Howlett and L.C. Jain, editors. Piscataway, NJ. p 788-791. 2000.

[Masulli, 2001]
F. Masulli, G. Valentini. Dependence among codeword bit error in ECOC learning machines: An experimental analysis. Proceeding of the second international workshop on multiple classifiers systems. MCS 2001. Cambridge, UK. 2001.

[Morris, 1998]
C.W. Morris, L. Boddy. Partitioned RBF networks for identification of biological taxa: Discrimination of phytoplankton from flow cytometry data. In Intelligent Engineering System Through Artificial Neural Net, Smart Engineering System, ASME Press. Vol 8. p 637-642. 1998.

[Morris, 2000]
C.W. Morris, A. Autret and L. Boddy 2000
A comparison of Strongly Partitioned RBF Networks and Support Vector Machines for Identification of Biological Taxa. Intellident Engineering Through Artificial Neural Networks Eds. C.H. Dagli, A.L. Buczak, J. Ghosh, M.J. Embrechts, O. Ersoy, S. Kercel. Vol 10, pp 783-788.

[Morris, 2001]
C.W Morris, A. Autret, L. Boddy. Support vector machines for identifying organisms - a comparison with strongly partitioned radial basis function networks. Ecological Modelling. Vol 146, pp 57-68

[Newton]
Newton's optimization technique
http://www.mapleapps.com/categories/mathematics/Operations%20Research/html/NewtonsMethod.html
19 Mach 2003

[Numerical recipes in C, 1992]
Numerical recipes in C: the art of scientific computing. William H Press.2nd ed.- Cambridge : Cambridge University Press. 1992.

[Opitz, 1999]
D. Opitz, R Maclin. Popular ensemble methods: An empirical study. In Journal of Artifical Intelligence Research. Vol 11, p 169-198. 1999.

[Orr, 1995]
Mark J. L. Orr. Regularisation in the Selection of Radial Basis Function Centres. In Neural computation. In Neural computation, p 606-623. 1995.

[Orr, 1996]
M. J.L. Orr. Introduction to Radial Basis Function Networks. Technical Report. Centre for cognitive sciences. University of Edinburgh.1996.

[Orr, 2000]
M. Orr, J. Hallam, K. Takezawa, A. Murray, S. Ninomiya, M. Oide, T. Leonard. Combining Regression Trees and Radial Basis Function Network. In International Journal of Neural Systems. Vol 10:6. p 453-465. 2000.

[Parzen, 1962]
E. Parzen. On estimation of a probability density function and mode. Ann. Math. Stat. Vol 33. p 1065-1076. 1962.

[Roberts, 1999]
S.J. Roberts. Novelty detection using extreme value statistics. IEE Proceedings on Vision, Image and Signal Processing. Vol 146:3. p 124-129. February 1999.

[Ronco, 1998]
E. Ronco, P.J. Gawthrop, D.J. Hill. Gated Modular Neural Networks for Control Oriented Modelling. Technical Report. 1999.

[Singh, 2002]
S. Singh, M. Markou. An approach to novelty detection applied to the classification of image regions. IEEE Transactions on Knowledge and Data Engineering. Accepted Revision. 20002.

[Sun, 1999]
R. Son. Multi-Agent Reinforcement Learning weighting and partitioning. Neural Networks. Vol 12:4-5. p 127-153. 7 April 1999.

[SVM Light]
SVM Light software home page. Joachims Thorsten. Cornell University http://svmlight.joachims.org. 20 March 2003

[Taylor, 2000]
O. Taylor, D. Addison. Novelty detection using neural networks technology. 13th international congress and exhibition on condition monitoring and diagnostic management COMADEM. 2000.

[Topchy, 1997]
A. P. Topchy, O. A. Lebedko, V. V. Miagkikh, N. K. Kasabov. An approach to radial basis function networks training based on cooperative evolution and evolutionary programming. In proceeding of the international conference on Neural Information Processing ICONIP'97. Syngapore, Springer Verlag. p 253-258. 1997.

[Vapnik, 1995]
V. N. Vapnik. The Nature of Statistical Learning theory. Springer. 1995.

[Wilkins, 1999]
M.F. Wilkins,L. Boddy, C.W. Morris, R.R. Jonker. Identification of Phytoplankton from Flow Cytometry Data by Using Radial Basis Function Neural Networks. In Applied and Environmental Microbiology. American Society for Microbiology. Vol 65: 10. p 4404-4410. 1999.

Chapter 7: Annexes

7.1 Data sets

7.1.1 The complete species list

This is the complete list of the phytoplankton used in this project

Phytoplankton Species Names	
Amphidinium carterae	Pavlova lutheri
Amphora coffaeformis	Pelagococcus subviridis
Aureodinium pigmentosum	Phaeocystis pouchetii
Chaetoceros calcitrans	Phaeodactylum tricornutum
Chlamydomonas reginae	Plagioselmis punctata
Chlorella salina	Pleurochrysis carterae
Chroomonas salina	Porphyridium pupureum
Chroomonas sp	Prorocentrum balticum
Chrysochromulina camella	Prorocentrum micans
Chrysochromulina chiton	Prorocentrum minimum
Chrysochromulina cymbium	Prorocentrum nanum
Chrysochromulina polylepis	Prymnesium parvum
Cryptomonas appendiculata	Pseudopedinella sp
Cryptomonas calceiformis	Pyramimonas grossii
Cryptomonas maculate	Pyramimonas obovata
Cryptomonas reticulate	Rhodella maculata
Cryptomonas rostrella	Rhodomonas sp
Dunaliella minuta	Scrippsiella trochoidea
Dunaliella primolecta	Skeletonema costatum
Dunaliella tertiolecta	Stichococcus bacillaris
Emiliania huxleyi 92	Tetraselmis impellucida
Emiliania huxleyi B11	Tetraselmis striata
Gymnodinium micrum	Tetraselmis suecica
Gymnodinium simplex	Tetraselmis tetrathele
Gymnodinium veneficum	Tetraselmis verrucosa
Gymnodinium vitiligo	Thalassiosira weissflogii
Gyrodinium aureolum	Pavlova lutheri
Hemiselmis brunnescens	Pelagococcus subviridis
Hemiselmis rufescens	Phaeocystis pouchetii
Hemiselmis virescens	Phaeodactylum tricornutum
Heterocapsa triquetra	Plagioselmis punctata
Micromonas pusilla	Pleurochrysis carterae
Nephroselmis pyriformis	Porphyridium pupureum
Nephroselmis rotunda	Prorocentrum balticum
Ochromonas sp	Prorocentrum micans
Ochrosphaera neopolitana	Prorocentrum minimum

7.1.2 Species list in tests chap 3.3.4

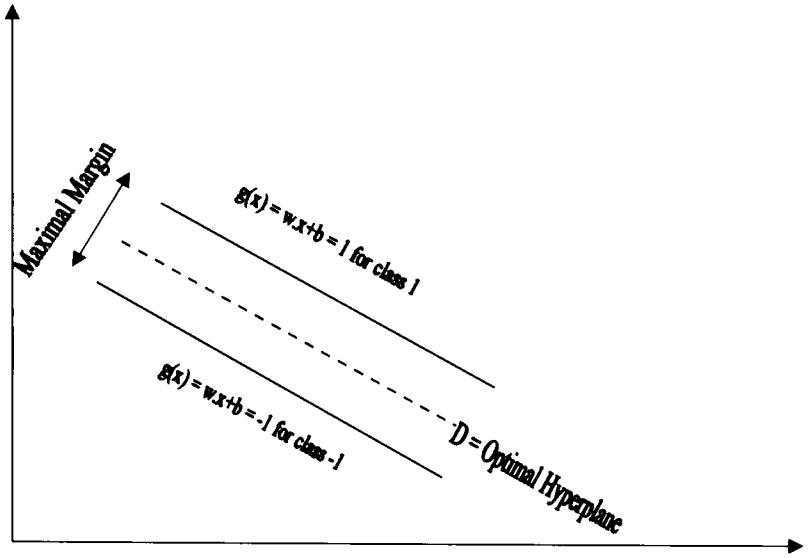
The original 20 species for the three series			
	Test 1	Test 2	Test 3
0	Chlorella salina.nna	Amphidinium carterae.nna	Amphidinium carterae.nna
1	Chrysochromulina camella.nna	Amphora coffaeformis.nna	Aureodinium pigmentosum.nna
2	Cryptomonas calceiformis.nna	Chroomonas salina.nna	Chlorella salina.nna
3	Cryptomonas reticulata.nna	Chrysochromulina chiton.nna	Chroomonas sp.nna
4	Cryptomonas rostrella.nna	Chrysochromulina cymbium.nna	Chrysochromulina chiton.nna
5	Dunaliella minuta.nna	Cryptomonas maculata.nna	Chrysochromulina cymbium.nna
6	Emiliania huxleyi 92.nna	Emiliania huxleyi 92.nna	Cryptomonas appendiculata.nna
7	Emiliania huxleyi B11.nna	Gymnodinium vitiligo.nna	Cryptomonas calceiformis.nna
8	Gymnodinium veneficum.nna	Hemiselmis brunnescens.nna	Cryptomonas maculata.nna
9	Hemiselmis brunnescens.nna	Hemiselmis rufescens.nna	Emiliania huxleyi 92.nna
10	Heterocapsa triquetra.nna	Nephroselmis pyriformis.nna	Emiliania huxleyi B11.nna
11	Nephroselmis pyriformis.nna	Pavlova lutheri.nna	Gymnodinium simplex.nna
12	Nephroselmis rotunda.nna	Pelagococcus subviridis.nna	Gymnodinium vitiligo.nna
13	Ochromonas sp.nna	Phaeocystis pouchetii.nna	Micromonas pusilla.nna
14	Pelagococcus subviridis.nna	Prorocentrum micans.nna	Nephroselmis rotunda.nna
15	Pleurochrysis carterae.nna	Pseudopedinella sp.nna	Porphyridium pupureum.nna
16	Pyramimonas grossii.nna	Pyramimonas obovata.nna	Pseudopedinella sp.nna
17	Pyramimonas obovata.nna	Stichococcus bacillaris .nna	Rhodella maculata.nna
18	Scrippsiella trochoidea.nna	Tetraselmis verrucosa.nna	Rhodomonas sp.nna
19	Tetraselmis striata.nna	Thalassiosira weissflogii.nna	Thalassiosira weissflogii.nna

7.1.3 Species list in tests chap 4

	Species
0	Gymnodinium vitiligo.nna
1	Gyrodinium aureolum.nna
2	Phaeocystis pouchetii.nna
3	Pleurochrysis carterae.nna
4	Prorocentrum minimum.nna
5	Prorocentrum nanum.nna
6	Prymnesium parvum.nna
7	Scrippsiella trochoidea.nna
8	Tetraselmis suecica.nna
9	Tetraselmis verrucosa.nna

7.2 Support Vector Machines (SVM)

7.2.1 - Maximum margin -



The formula that gives the distance between one point and one plane is:

$r = \frac{Ax_1 + By_1 + Cz_1 + b}{\sqrt{A^2 + B^2 + C^2}}$, P (x_1, y1, z1) and Plane = Ax+By+Cz+b

Now let define vector $w = \begin{cases} A \\ B, x = \begin{cases} x_1 \\ y_1 \end{cases} and \parallel w \parallel = \sqrt{A^2 + B^2 + C^2} \\ C \end{cases}$ then

$r = \frac{wx + b}{\parallel w \parallel}$

It is now easy to see that the numerator correspond to the equation of g (x). In this case, it is looking for the distance to the optimal hyperplane whose vector director is w_0 thus w has to be replaced. It gives $r = \frac{g(x)}{\parallel w_0 \parallel}$.

When x is on the optimal hyperplane g (x) equal 0, when x is in class 1 then g (x) is equal to 1 and the distance is $r = \frac{1}{\parallel w_0 \parallel}$ and when x is in class –1, g (x)

equal −1 then $r = \frac{1}{\| w_0 \|}$. Thus the maximum margin between the two classes is $2r = \frac{2}{\| w_0 \|}$.

7.2.2 - Dual problem transformation -

$$L_P = \frac{1}{2} w^t w - \sum_{i=1}^{n} \lambda_i \left[y_i \left(w^t x_i + b \right) - 1 \right] \quad (22)$$

The next step in the process is to differentiate (22) regard to w and b, which gives (23) for w and (24) for b.

$$\mathrm{w} = \sum_{i=1}^{N} \lambda_i y_i x_i \quad (23)$$

$$\sum_{i=1}^{N} \lambda_i y = 0 \quad (24)$$

Then (22), (23) and (24) are merged. Before merging, it is necessary to develop (22), which gives (25).

$$L_P = \frac{1}{2} w^t w - \sum_{i=1}^{N} \lambda_i y_i w^t x_i - b \sum_{i=1}^{N} \lambda_i y_i + \sum_{i=1}^{N} \lambda_i \quad (25)$$

From (24), it is known that the third term from the left is equal to 0 thus it gives (25.1).

$$L_P = \frac{1}{2} w^t w - \sum_{i=1}^{N} \lambda_i y_i w^t x_i + \sum_{i=1}^{N} \lambda_i \quad (25.1)$$

The most left term can then be transformed using (23) which gives (25.2).

$$\frac{1}{2} w^T w = \frac{1}{2} \sum_{i=1}^{N} \lambda_i y_i w^t x_i = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_j y_j x_j \lambda_i y_i x \quad (25.2)$$

It is also possible to transform the second term from the left which gives (25.3).

$$\sum_{i=1}^{N} \lambda_i y_i w^t x_i = \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_j y_j x_j \lambda_i y_i x_i \quad (25.3)$$

Finally by replacing (25.2) and (25.3) in (25.1), the equation becomes equation (26).

$$L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i x_j \quad (26)$$

(26) is call the "**Dual problem**", it is only defined from the training set and does not need w or b. Once the lagrangian multipliers are known, it is easy to obtain w and b be replacing them in (23) and (19).

7.2.3 - Non-linear Problem -

$$\sum_{j=1}^{n} w_j \varphi_j(x) + b \tag{32}$$

The difference between (32) and (19) from the linear problem is φ. This new parameter represents the non-linear transformation that link x from the linear space to its image in the non-linear space. Using (32) and by adapting the linear process to the new parameter, it is possible to obtain (33). (33) is part of the "Primal problem".

$$\sum_{i=1}^{m} \lambda_i y_i \varphi^T(x_i)\varphi(x) = 0 \tag{33}$$

"m" is the dimension of the feature space. Now that the operation to obtain (33) was explained, the next step will be to clarify what its structure brings. First the inner product of the non-linear transformation vector can be replace by $\varphi^T(x_i)\varphi(x) = \sum_{j=1}^{m} \varphi_j(x)\varphi_j(x_i)$. This is a symmetric function; thus Mercer's theorem (Annexe 9.2.4) says that it can be replaced be an inner-product kernel which gives (34).

$$\sum_{i=1}^{m} \lambda_i y_i K(x_i x_j) = 0 \tag{34}$$

7.2.4 The Mercer theorem

Mercer's theorem states that a continuous and symmetric kernel k that gives a positive integral, it can be expanded in a uniformly convergent series in term of Eigen functions ψ_j and positive Eigen values λ_j, $n \leq \infty$.

$$k(x,y) = k(y,x) = \sum_{j=1}^{n} \lambda_j \psi_j(x) \psi_j(y)$$

It is possible to see that (33) has the required shape thus it can be replace by a Mercer's kernel.

Some of Mercer's kernels	
Polynomial kernel	$(x^T x + 1)^P$
RBF kernel	$e^{\left(-\frac{\|x-x_i\|^2}{2\sigma^2}\right)}$

7.3 Kohonen Guidelines

T. Kohonen (1995) gave the following recommendations when he wrote "Self-Organizing Maps".

1) Use of a fast initial learning rate, Alpha = 0.9. Alpha Half life should be around 1000 cycles. He also recommended linear decay but admitted in the book that in general it does not matter much.
2) Initial neighbourhood size should equal half the grid size *n*. Gaussian size = *n*/4. Gaussian half-life=1000, with linear decay.
3) After half-life, Alpha initial value = 0.01. Alpha half-life = 20000. Exponential decay. Neighbourhood size = 1.
4) Train for further 50000-100000 cycles. Total number of cycles should be around 500 times the number of nodes in the grid.

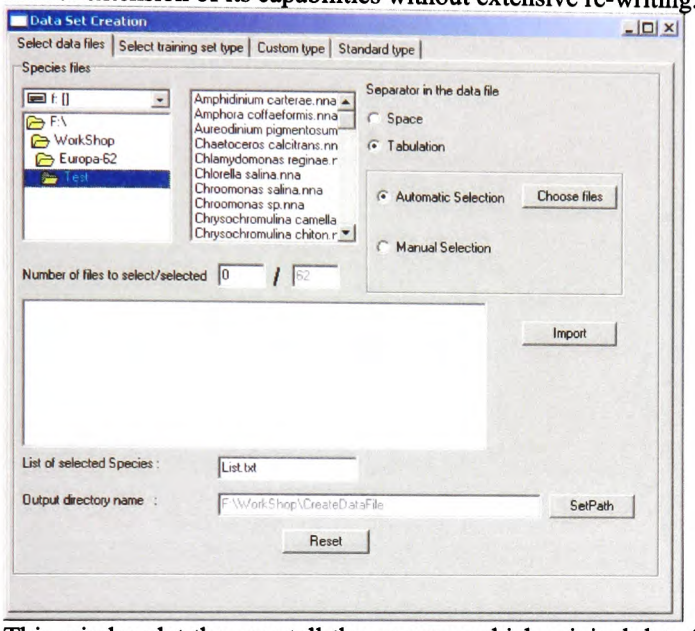
7.4 List of acronyms

ANN	Artificial Neural Network
NN	Neural Network often for ANN
MNN	Modular Neural Network
HPNN	Highly-Partitioned artificial Neural Network
RBF	Radial Basis Function Network
RMS	Root Mean Square error
LVQ	Learning Vector Quantization
SVM	Support Vector Machine
FC	Flow Cytometer
MLP	Multi Layer Perceptron => Backpropagation ANN
WTA	Winner-Takes-All
AWTA	Adaptive Winner-Takes-All
FBD	Feature Base Decision rule
PFBD	Partial Feature Base Decision rule
FFBD	Full Feature Base Decision rule

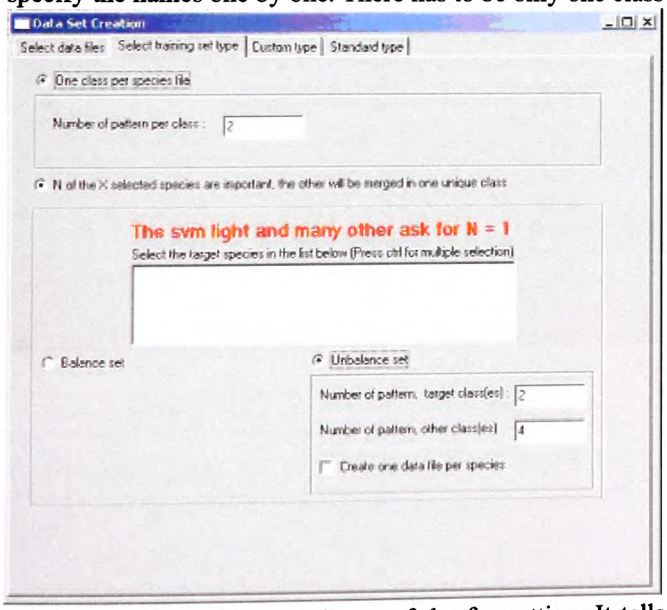
Chapter 8: Software manuals

8.1 Format Software

As there are many different formats of Flow Cytometer output files and as many format of the artificial neural networks' training file, a conversion program was created. The software has been written in a modular fashion that is, it is possible to add options without having too much rewriting to do. This software was used to transform the EurOPA FC output files into useable and practical data files for the SVM and RBF paradigms. The two paradigms do not use the same training file format. The other objective of this program was to allow further extension of its capabilities without extensive re-writing.

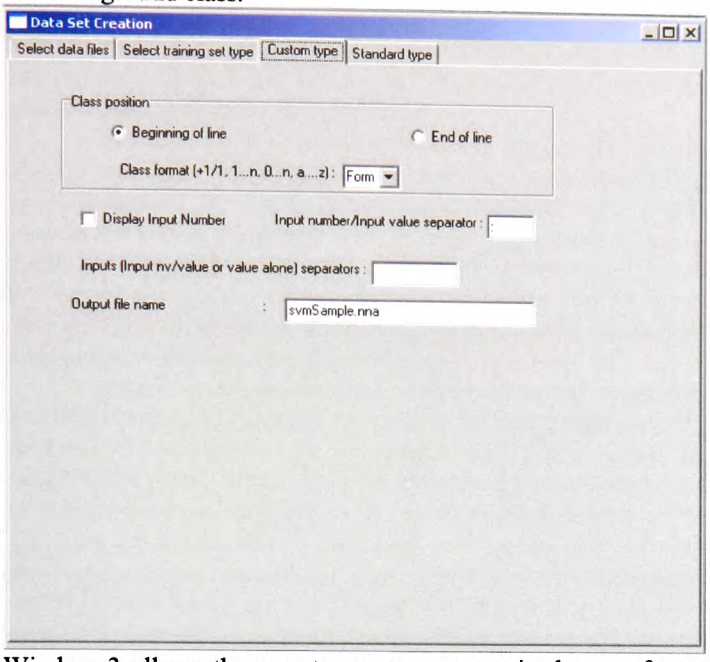


This window let the user tell the program which original data files/files will be used to create the new data file(s). The user needs to indicate if the original files use Tabulation or Space between the different parameters inside a vector (7 FC parameters make a 7 dimensions vector). If there is a list of files that exists, it is possible to use it so that it is not necessary to specify the names one by one. There has to be only one class per file.

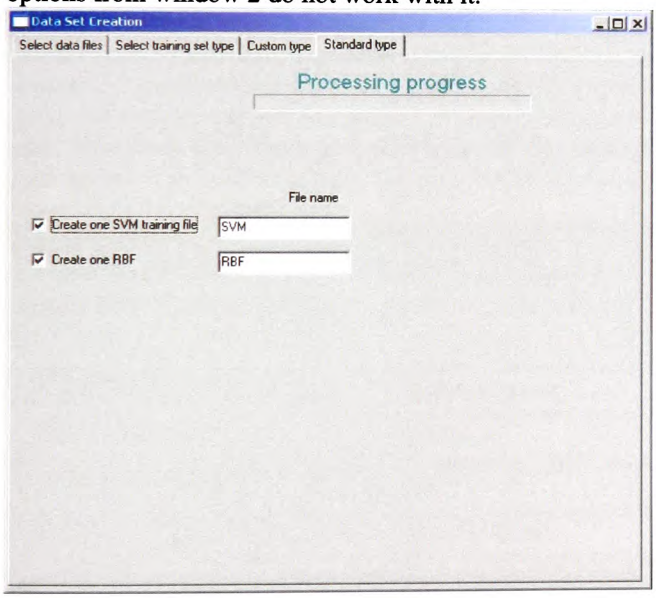


Window 2 shows the second step of the formatting. It tells the program how to generate the output file. The one class per species option tells the program that every class from the original files becomes one class in the single output file. It is possible to select the number of pattern per species.

The second option is more complex. It allows the user to put one or more species in the target class and the rest into a background class. More important for our project, it allows the user to create several files, each of them having a different class as the target and all the others in the background class. It also allows the user to select a different number of patterns for the target and background class.



Window 3 allows the user to create a customized type of output file; however, some of the options from window 2 do not work with it.



This is the final window, it contains predefined output format. The RBF type that is adapted to the original RBF software developed by Wilkins[1999] and the SVM type adapted to the original SVM Light format from Joachims Thorsten. The RBF type is the only one used by the actual test bed system, as the test bed does the conversion internally and on-the-fly to save file space.

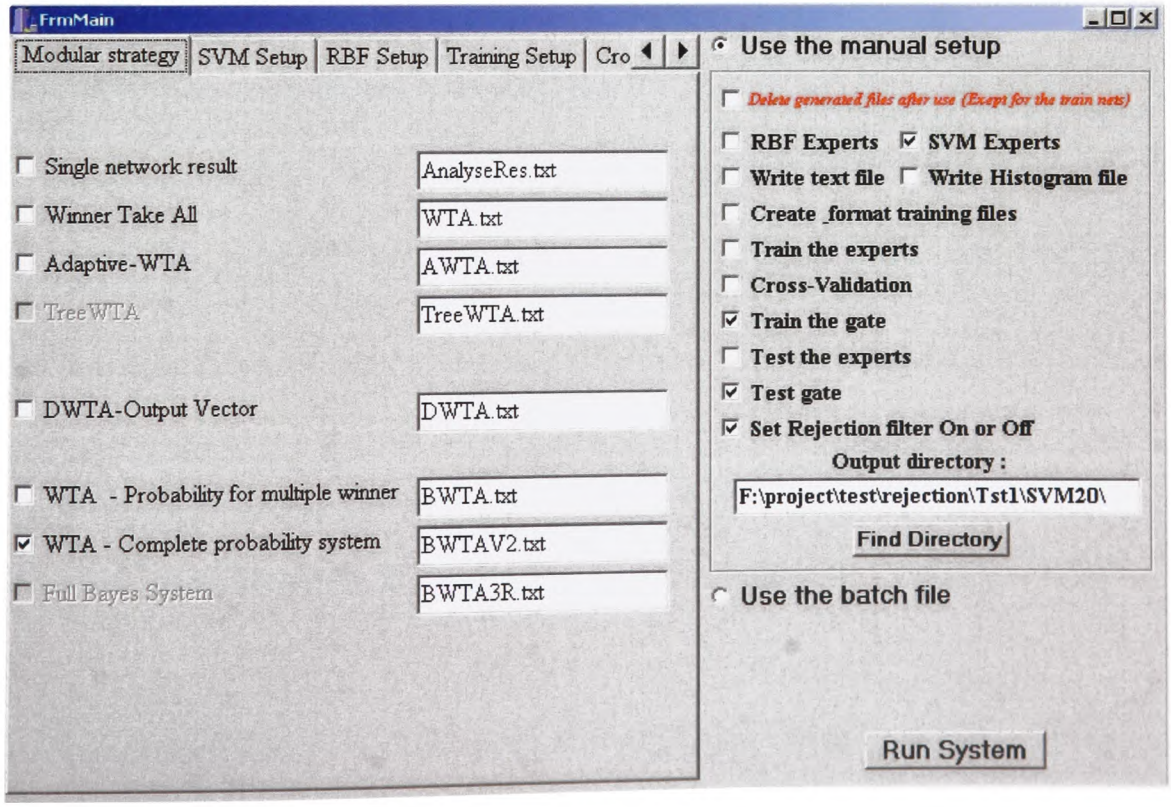
8.2 Test-Bed Manual

Early in the project, it was decided to use two different paradigms: RBF and SVM. As several tests and been successfully conducted with Malcom Wilkins[1999]'RBF, it was selected as the RBF model. The SVM Light from Joachim Thorsten was selected because it was one of the few SVM model that was freely available with the source code and at this early stage of the project, I had to think that the user might have to modify the SVM model to fit the specific requirements of this project.

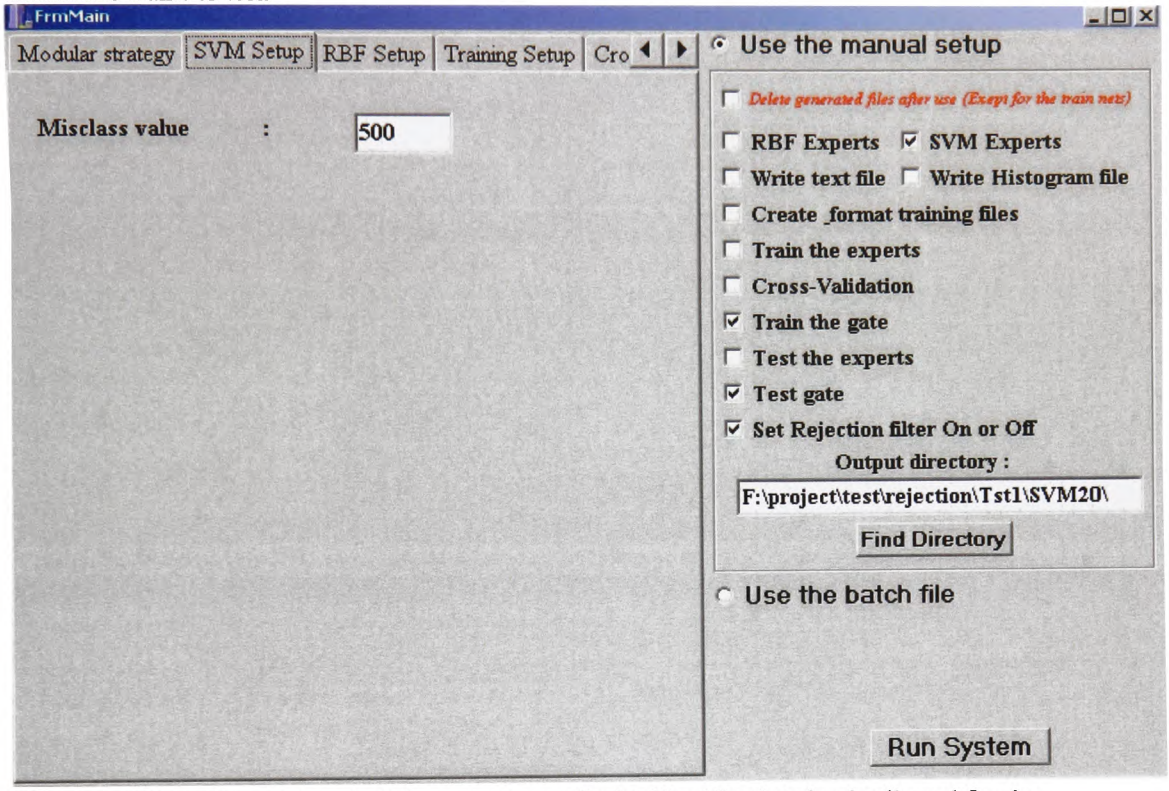
However, those two paradigms had been written for two different environments. The SVM was a Linux based software written in C and the RBF paradigm had been written in an old C standard for DOS. Thus, these two softwares were ported from their respective environment to a Windows C++ Builder environment. As, the project required, the ANN models to run sequentially and even in the future simultaneously on a single machine, after porting the models to a common environment, the models were encapsulated into C++ objects. Coming from different environment, it required some alteration to the original source code and even partial re-writing of several functions.

Once encapsulated into objects, the two paradigms were merged into a single graphical program. In order to be able to test, extend and modify the test-bed program as required, the system's task i.e. train experts, train gates, test experts... was broken into independent parts. That is that provided that the necessary data are available, each task can run be itself. Every task uses a data file as information source and save its results into a destination file. This destination file is then used as the source for the next step. Thus, providing that the previous stage ran normally, it is possible to run a given step on its own. It was a necessary and useful feature during the development but also during tests. For example in Chapter 4, the tests involve trying different set of combinations parameters, because the system allows a stage to run by-itself, it was not necessary to re-train the experts for every combinations.

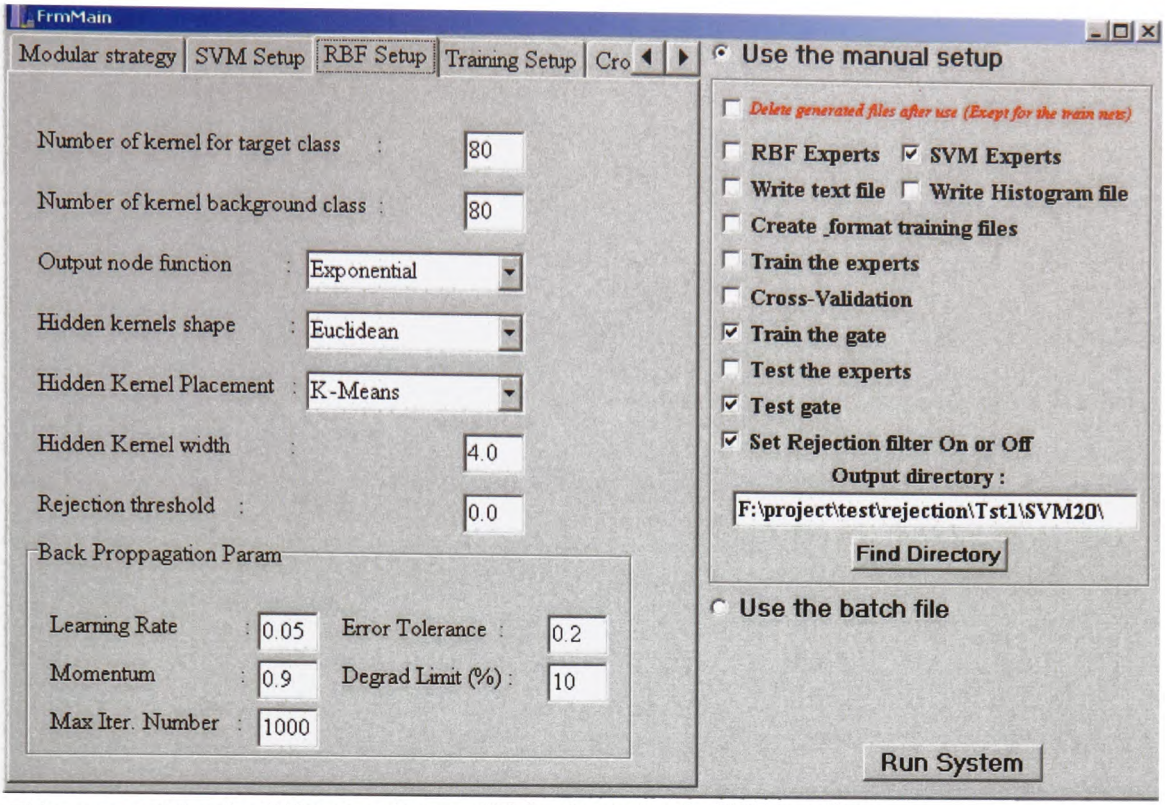
As training ANN can take a long time (minutes, hours), the system had to be able to run several trained experts sequentially without intervention. Moreover, it had to be able to run several tests sequentially and automatically. In fact, one series of test can such as the one in chapter 3.3 takes several hours so it was necessary to run these tests sequentially over night. It led to the development of two modes: A manual stage and a batch mode. In manual mode, the user supervises everything and manually set the options. In batch mode, a file contains a description of the different tests, the program reads through the file, run one test, finish it and move on to the next test.



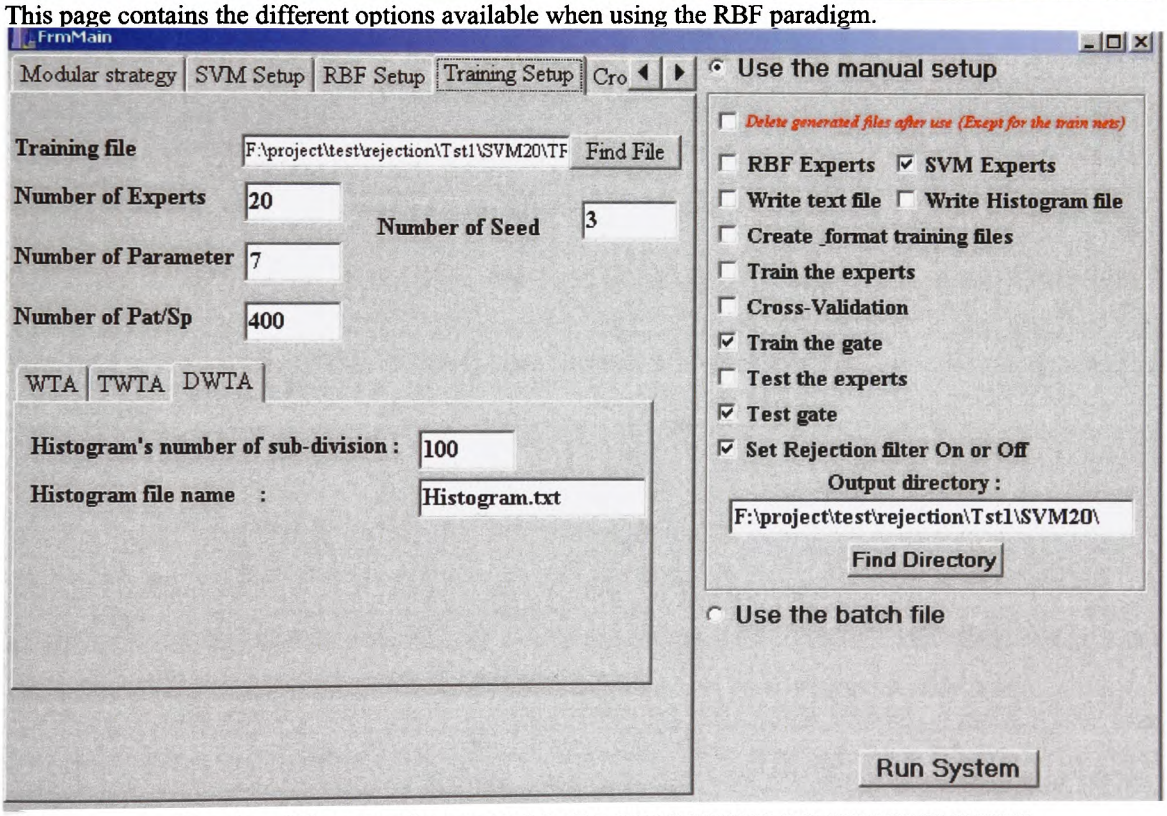
The test bed window is split in two parts. The top right side lists the different stages of the system use. Each stage generates a file that is necessary for the next one. Thus, providing that the necessary file is available the different stage can run one by one.
The bottom right side is use to indicate where the batch file is. If the batch mode is selected, the batch file contains all the options that need to be checked thus the left side is not necessary.
One the left, there is a portfolio; each page corresponds to options for a different stage.
The first picture shows the modular page, it allows the user to select the combination strategies that he/she wants to test.



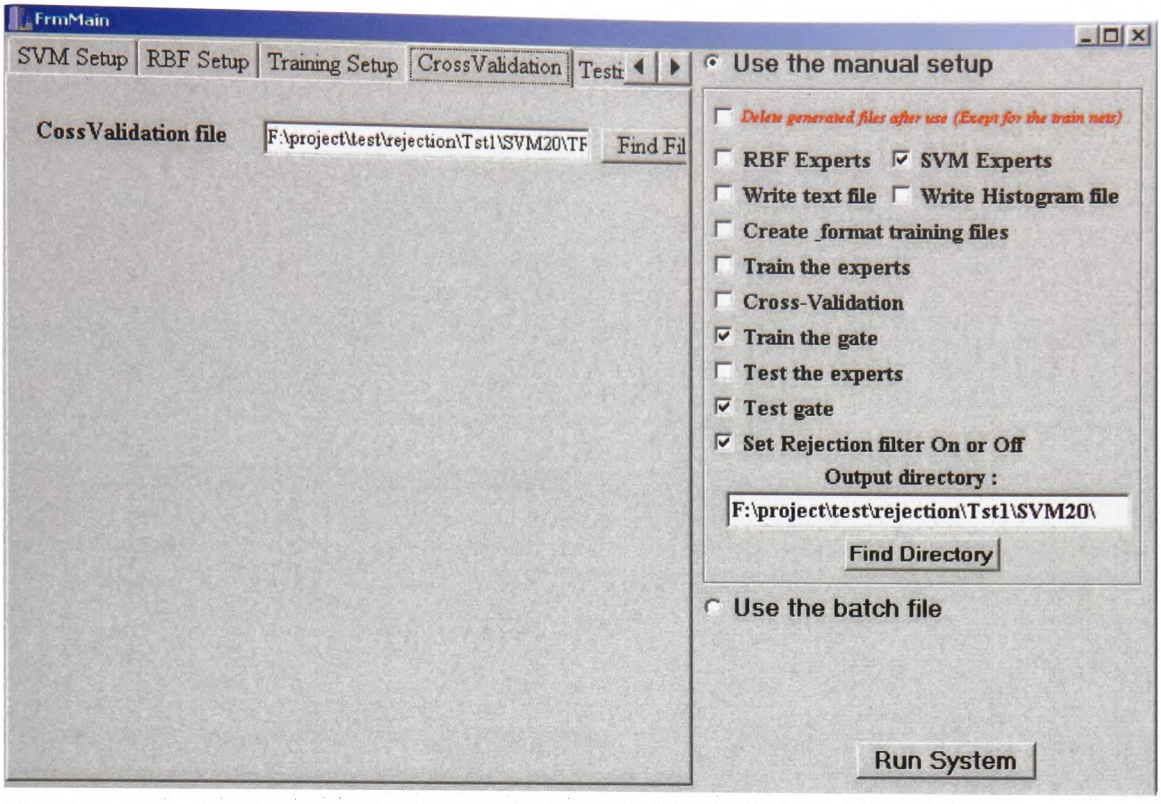
This page allows the user to select the percentage of miss-identification that is allowed for the SVM paradigm.



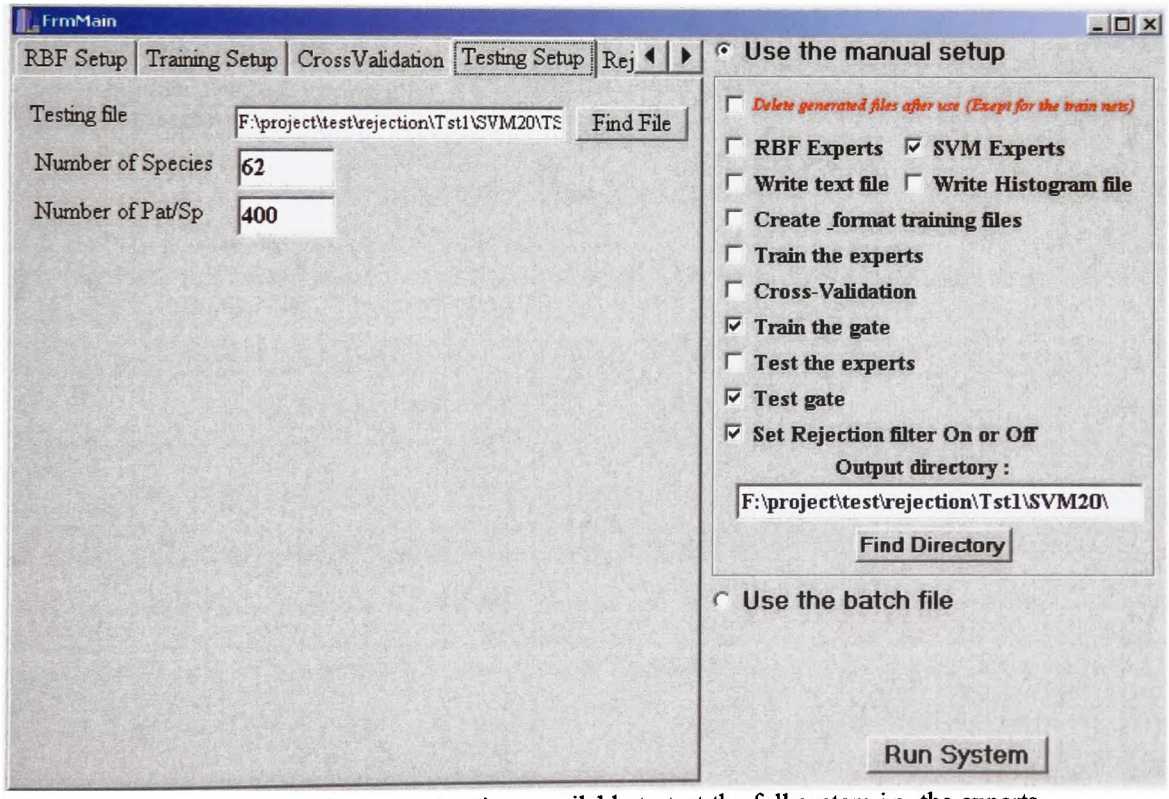
This page contains the different options available when using the RBF paradigm.



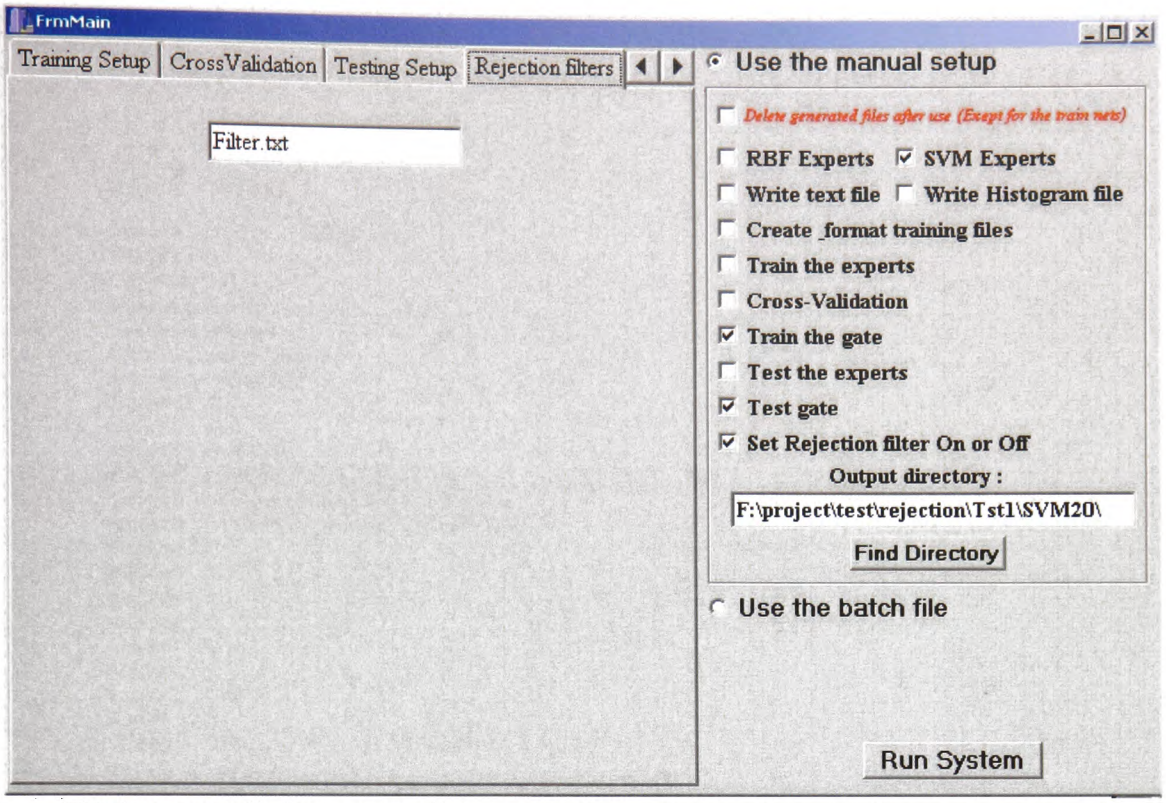
This page specifies the different options regard to the expert training e.g. number of experts, number of patterns per species, number of FC parameters… It also set the options for the combination strategies that requires user intervention e.g. number of interval per histograms and it of course set the training file name.



This page allows the user to use a different file to check if the ANNs are well trained before training the combination strategies on the resulting files.



This is the test page, this page contains options available to test the full system i.e. the experts when combined using HPNN and any of the combination strategies. It is possible to use a file that contains more species than the one the system was trained on to observe the system behaviour with unknown data.



This last page tells the program where to store the results once they went through the rejection stage.

N.B:

- The rejection options are not yet available through the batch or the manual mode as this stage is not fully integrated.
- In order to create the batch file it is possible to write a text file using a word processor but there is also a program to do that.

8.3 Sample Batch file

```
!
!       Test 5 Species
!
!G      namd2win        dwta2.res
!G      hisdwta 5
!G      hisdwta 5
!G      namdwin dwta.res
!G      namawin awta.res
!G      namsing analyse.res
!G      namtwin twta.res
!G      nbpattr 400
!G      nbpatts 400
!G      namwin  wta.res
!G      outdir  c:\test\test\t5\
!G      trpath  c:\test\test\t5\trt5.nna
!G      tspath  c:\test\test\t5\tst10.nna
!G      twtsee  5
!G      twtsee  5
!G      twtsee  5
!G      wthres  0
!G      nbexp   5
!G      nbpar   7
!G      trexp   OFF
!G      trgat   ON
!G      tsexp   OFF
!G      dwta2   OFF
!G      tsgat   ON
!G      awta    ON
!G      form    ON
!G      hist    5
!G      nbsp    10
!G      text    ON
!G      dwta    OFF
!G      twta    OFF
!G      rbf     ON
!G      sgle    ON
!G      svm     OFF
!G      wta     OFF
!
!Radial Basis Function Network
!
!R      kerwid  2
!R      thresh  0
!R      tgtcla  30
!R      bckcla  30
!R      strat   LVQ
!R      metr    EUCL
!R      funct   EXP
!R      lrn     0.05
!R      mom     0.9
!R      maxit   1000

!R      tol     0.2
!R      degrad  10
!
!Support Vector Machines
!
!S      penal   1000
!
!
!       Test 20 Species
!
!G      namd2win        dwta2.res
!G      hisdwta 5
!G      hisdwta 5
!G      namdwin dwta.res
!G      namawin awta.res
!G      namsing analyse.res
!G      namtwin twta.res
!G      nbpattr 400
!G      nbpatts 400
```

```
!G      namwin  wta.res
!G      outdir  c:\test\test\t20\
!G      trpath  c:\test\test\t20\trt20.nna
!G      tspath  c:\test\test\t20\tst24.nna
!G      twtsee  5
!G      twtsee  5
!G      twtsee  5
!G      wthres  0
!G      nbexp   20
!G      nbpar   7
!G      trexp   OFF
!G      trgat   ON
!G      tsexp   OFF
!G      dwta2   OFF
!G      tsgat   ON
!G      awta    ON
!G      form    ON
!G      hist    5
!G      nbsp    24
!G      text    ON
!G      dwta    OFF
!G      twta    OFF
!G      rbf     ON
!G      sgle    ON
!G      svm     OFF
!G      wta     OFF
!
!Radial Basis Function Network
!
!R      kerwid  4
!R      thresh  0
!R      tgtcla  80
!R      bckcla  80
!R      strat   LVQ
!R      metr    EUCL
!R      funct   EXP
!R      lrn     0.05
!R      mom     0.9
!R      maxit   1000
!R      tol     0.2
!R      degrad  10
!
!Support Vector Machines
!
!S      penal   1000
!
```