What Intelligent Agent is Smarter? A Comparison

By

Luis C. Rabelo Mendizabal
Ph.D., Engineering Management (1990)
University of Missouri

M.S., Engineering Management (1988)
University of Missouri

M.S., Electrical Engineering (1987)
Florida Institute of Technology

B.S., Electrical and Mechanical Engineering (1983)
Technological University of Panama

Submitted to the System Design and Management Program in Partial Fulfillment of Requirements for the Degree of Masters of Science in Engineering and Management

AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY, FEBRUARY 2001.

Signature of Author: ______________________
Luis C. Rabelo Mendizabal
System Design and Management Program

Certified by: ______________________
Thesis Supervisor: Dr. Dan Ariely
Sloan School Career Development Assistant Professor of Management

Accepted by: ______________________
LFM/SDM Co-Director: Dr. Stephen Graves
Abraham Siegel Professor of Management

Accepted by: ______________________
LFM/SDM Co-Director: Dr. Paul A. Lagace
Professor of Aeronautics & Astronautics and Engineering Systems

What Intelligent Agent is Smarter? A Comparison

by
Luis C. Rabelo Mendizabal

Submitted to the System Design and Management Program January 2001 in Partial Fulfillment of Requirements for the Degree of Masters of Science in Engineering and Management

1 Abstract (less than 350 words)

Agent systems of the feature-based filtering type act as recommenders using a database about past user's behaviors and preferences to predict additional products or subjects the user might like. The effectiveness of the agent (how intelligent is it) is depending on the learning scheme utilized and the representation of this knowledge environment.

In this thesis, we describe several algorithms, which can be appropriate to be the center of such scheme, including supervised learning neural networks and support vector machines. The predictive accuracy of these methods is compared in a representative problem. Several sets of experiments were run with different sizes of training datasets and initial conditions. The results of the experiments show that support vector machines have a better performance when the training data set is very limited in size. However, supervised neural networks based on minimizing errors (i.e., Backpropagation) are able to provide good answers when the training data sets are of relative large size. In addition, supervised neural networks based on forecasting by analogy (i.e., Fuzzy ARTMAP) are also able to exhibit good performance when ensemble schemes are utilized.

Advisor: Dr. Dan Ariely
Title: Sloan School Career Development Assistant Professor of Management

2 Acknowledgements

There are no words to express my total gratitude to the BFGoodrich Corporation, Honeywell International, and my supervisors Mark Shmorhun (BFGoodrich) and John Christoffel (Honeywell Labs).

I would also like to express my most sincere gratitude to my Advisor Dan Ariely. Our conversations about agents and electronic commerce have helped me to forge my ideas about what technology is and to understand its business relationships. I would like to mention the MIT "Latin Mafia" (Luis Blando, Martin Busse, Edwin Elberg, and Rocco Paduano). We created an environment of learning and fun at MIT-SDM.

I would like to express my most sincere appreciation to my parents, my sister, and Otilia for their support. I would like to thank especially my mother who always motivated me to come to MIT. Her friendship and generosity created the necessary environment and gave me the needed inspiration to carry out my tasks. This thesis and my work at MIT are mainly dedicated to honor her.

Finally, my deep appreciation and love to my wife Gudlaug and my beloved son Luis Gisli.

3 Tables of Contents, Figures, and Tables

Table of Contents

Table of Figures

Table of Tables

4 Introduction

This section introduces the problem, describes the approach to the problem, provides the most important definitions, and presents the organization of this thesis.

4.1 The Problem and Answer

People have been doing business for thousands of years, and the only smart agent whom people have had is another human being. And this “human” agent is still the smartest agent. An analysis of current research in Artificial Intelligence would indicate that this human agent will continue to be the smartest one for quite some time. Unfortunately, a human agent is not always available; a human agent is usually very expensive, sometimes very emotional, not adequate for repetitive tasks, time restricted (e.g., not able to analyze and deliver results in nanoseconds), and limited in managing complexity (e.g., discovering useful patterns and relationships in large amounts of data, information bottleneck). According to Murch and Johnson [64], the features of a human agent are the following:

Is focused on a task.
Is a specialist with skills that I do not have.
Has access to information relevant to a task.
Has the contacts to provide the service.
Can provide the service at a fraction of the cost of doing it myself.
Provides a service I can not get any other way.

On the other hand, the growth of the Internet is currently occurring at phenomenal rates: there are more than 14 million Internet sites, 250 million online users, and the traffic in the Internet and the time spent on line are increasing in a nonlinear fashion. Web content is growing at accelerated rates too [53]. The Web consists of approximately 2.5 billion documents with a rate of growth of 7.3 million pages per day (estimates of the average page size vary from 10 Kbytes per page to 20 Kbytes per page – a total of approximately 25 to 50 terabytes of information). The specialized Web-accessible databases and dynamic websites account for another impressive 550 billion Web-connected documents (approximately 7,500 terabytes of information). In this new environment, agents are needed to support our activities, ranging from simple Web surfing to sales and shopping. For instance, effective agents will change the nature of sales on the Internet from the current basic sales scheme to a full customer management approach. Agents can change the nature of interactions on the Internet from simple access to large databases, to dynamic and personalized information and advice sources. To implement this approach information systems will have to (1) learn their users' criteria and (2) learn how to aggregate information from different mediums and help to reinforce this information using these mediums. In order to perform both these tasks, these agents must be *intelligent* and implemented in software. This thesis addresses the question: how does one provide *intelligence* to software agents?

This thesis presents empirical data regarding the relative predictive performance of Backpropagation neural networks, Fuzzy ARTMAP neural networks, and Support Vector Machines in implementing intelligent recommendation systems based on individual models for electronic commerce. Although some results are presented addressing the computational issue, the main focus is the accuracy of the predictive techniques.

4.2 What is an Agent?

The dynamic, uncertain, massive, and distributed nature of information on the Internet has made it very difficult for humans to process this information. As expressed by Ariely [2], "while this electronic marketplace gives consumers many more choices, alternatives, and freedom in their information gathering and decision making processes, the vast amounts of information and options also make their task vastly more complex." A possible solution to this problem is to develop software that not only responds to requests for information but also predicts, adapts, and aggressively develops and provides the means to help users increase their productivity and satisfaction. This next generation of software, currently being researched and developed, aims to automatically get the required information (i.e., information gathering and filtering), accept abstract tasking, solve simple problems and help the user solve difficult problems, and take action on the user's behalf [31]. This type of software has been denominated a software agent.

From the consulted literature and current research, it is possible to perceive three sides in software agents [5,7,29,50,57,58,65,101]:

1. **Researchers from the field of human-computer interaction**. These researchers are concerned about the aspects of interaction between humans and software agents.

2. **Researchers in the fields of distributed artificial intelligence and robotics.** These researchers are concerned about the development of techniques for negotiation and planning, situated action, protocols, concurrency, and component-based frameworks.

3. **Researchers in the field of electronic commerce**. These researchers emphasize agents that are Web-based but different in nature, and commercial rather than research. These researchers use technologies developed by the previous two groups.

The metaphor of a software agent emphasized in this thesis focuses on learning and intelligent assistance to users on the Internet. It belongs to the third side mentioned above. In addition, our work does not include the agent paradigm that provides coordination, interoperability, and control between loosely-coupled components of distributed systems such as air traffic control and complex military systems [31]. Our interest is in the software agent that is a computer system situated on the Internet, and that is *intelligent* and capable of *autonomous* action in order to provide intelligent assistance and make highly specific customer recommendations for products and services (e.g., books, cars, clothing, furniture, music, movies, wines, professional development courses, vacation plans).

These two concepts of *autonomous* and *intelligent* are still very controversial [67]. Autonomy means that this software agent is able to act without the direct intervention of humans or other agents. Therefore, this software agent has control over its own actions and internal state [101]. Autonomy is very different from machine intelligence. Autonomy is complementary to machine intelligence. Machine intelligence is related to the concept of "plasticity." Plasticity means the *flexibility* to respond, change, learn, and adapt in order to meet design objectives in an effective and efficient manner. An agent is intelligent when this agent is capable of being *flexible*. By *flexible,* the system must be [101]:

1. **Responsive.** Agents should understand their environment and respond in a timely fashion to changes that occur in it. The environment, in this thesis, is formed by the Internet, a user, a collection of agents, sellers, competitors, and regulators.

2. **Proactive**. An intelligent agent should be able to anticipate and predict situations with a high level of effectiveness. In addition, as stated by Wooldridge and Jennings [101], an intelligent agent should "exhibit opportunistic and goal-directed behavior." For instance, these recommendation agents should learn by observing the behavior of the consumers in order to anticipate recommendations estimated to be best for the consumers.

3. **Communications**. Communications and interactions are essential features of an intelligent agent. The agent must interact with other agents and humans in order to fulfill its own decision-making activities and to assist others with their decisions.

The dimensions of responsive and proactive can be mapped to the dimensions of learning, adaptation, and anticipation. There are several levels of accomplishment for these dimensions. For instance, to be able to say that a system is intelligent, the system has to be capable of learning. Then to measure learning, tests have to be designed to ask how well it learns what it learns. Therefore, we have to test and then label the respective level of accomplishment.

This intelligent capability has to be provided to a software agent. There are several schemes being studied to do this (and the final word is still far off!). The work in this thesis investigates three machine learning schemes that are candidates to provide machine intelligence to an agent. These three learning schemes have been the dominant paradigms of machine learning for the last 10 years. First, neural networks based on the minimization of errors (as represented by Backpropagation neural networks [73,99]) were re-introduced in the late 1980's and achieved technological maturity by the end of the 1990's. Second, neural networks based on analogies (as represented by Fuzzy ARTMAP [21,22]) have been researched extensively and further developed in the 1990's. Third, the new paradigm of Support Vector Machines, introduced "formally" by Vapnik in 1995 [90], is still in the phases of research and development and has become the intersection of three important fields: neural networks, statistics, and optimization. These three machine learning paradigms have not been previously compared for the task of providing intelligence to agents that implement intelligent recommendation systems on the Internet.

4.3 Current Approaches

We stated above that software agents can change the nature of interactions on the Internet: from simple access to large databases, to dynamic and personalized information and advice source. This approach becomes more important when product features and attributes are complex and qualitative as well as when the opportunities for differentiation, customization, and tailoring to individual preferences increase. In order to implement a software agent approach as an intelligent recommendation system, these agents have to be intelligent enough to learn their users' criteria and learn how to aggregate information from different mediums and how to help reinforce this information using these mediums. There are two approaches to the implementation of software agents that act as intelligent recommendation systems: the collaborative filtering approach (also called the community-based approach), and the feature-based filtering approach (also called the individual-based approach).

4.3.1 Collaborative Filtering

The collaborative filtering approach is based on the past behaviors of many individuals (i.e., a community) [2,37,52]. The information and relationships from this large group are collected into a knowledge base, which is then used in two steps: clustering and recommendation.

1. **Clustering.** The knowledge base is used to classify the target user into a cluster made up of other individuals who are similar to the target user based on the overlap in past behavior (i.e., individuals with similar tastes).

2. **Recommendation.** The smart agent recommends a product that has the highest possibility of being well matched for the target user based on the preferences of the cluster. The smart agent examines the products with the highest purchase rate in the cluster, and uses them as a predictor of what the target user will want.

A good example of collaborative filtering is the Firefly system [43]. The Firefly system starts by asking the target user to rank and compare a number of alternative products. The system searches through its database and tries to find other users with similar views. If matches are made, the system tries to find highly ranked choices of these matched users. These highly ranked choices become recommendations for the target user. The main advantage of this type of smart agent is that it is relatively cost effective in terms of the effort needed by the user. Collaborative filtering does not necessarily need to have enough information about a target user to be categorized into a cluster to form a prediction of what other types of products he/she will prefer. On the other hand, the assumption of fit with the cluster has additional implications, some of them potentially negative (e.g., when tastes vary widely, when matches are also domain specific, narcissism) [2,3].

4.3.2. Feature-Based Filtering

The feature-based filtering approach is very different from the collaborative filtering approach. This approach does not use information that is based on other individuals. Feature-based filtering attempts to capture the underlying utility structure of the target user. Therefore, the knowledge base consists of the past behavior of the target user. Recommendations are made based on the fit between the preference structure of the target user and the features and attributes of the different products. Smart agents using the feature-based filtering approach work in two steps: utility estimation and recommendation.

1. **Utility Estimation**. The smart agent estimates the strength of the relationship between the target user's tastes and the underlying product attributes. The knowledge base that contains the past behavior (e.g., past purchasings) of the target user is utilized to build this relationship.

2. **Recommendation**. Once the strength of the relationship between these features is established, the smart agent examines the products (from the database of products and their features), and for each product calculates the preference value. As expressed by Ariely [2] "The product with the highest expected value is then chosen as the one with the highest likelihood to be preferred by the target decision-maker."

There are several advantages to this type of smart agent. The recommendations can be optimized due to the ability to learn and adapt according to a specific target user. The system is more flexible and can change over time to reflect the changes in the target user by updating frequently the knowledge base and/or weighing past experiences. Smart agents based on feature-based filtering can also be more flexible, and react better to changes in the marketplace because they are based on relationships between the target user and the product's features. One of the main difficulties of implementing this smart agent is the size of the knowledge base. The size of the knowledge base is important in order to learn with a higher performance about each individual's preferences.

This thesis studies the performance of three different machine learning techniques in order to build smart agents of the feature-based filtering type. The next section will provide more details about the opportunities that smart agents of the feature-based filtering type provide.

4.4 Organization

The goal of this thesis is the performance analysis of three machine learning algorithms as recommender agents of the feature-based filtering type. Section 5 of this work mentions different techniques used to build intelligent recommending agents of both the collaborative filtering and feature-based filtering approaches. The machine learning methods selected in this thesis to enhance the performance of the feature-based filtering agents are discussed in Section 6. Backpropagation neural networks, Fuzzy ARTMAP neural networks, and Support Vector Machines are presented in some level of detail. Techniques and concepts to address each one of the implementations are presented. Section 7 presents the different experiments and an analysis of the results of the different algorithms. Empirical data is presented regarding the relative performance of the selected algorithms. Section 8 concludes by summarizing the different results, presenting feedback received, and lessons learned during the different experiments and implementations. In addition, Section 8 discusses directions for further research on the role of machine learning as the enabling mechanism to make intelligent agents smarter.

5 Literature Survey

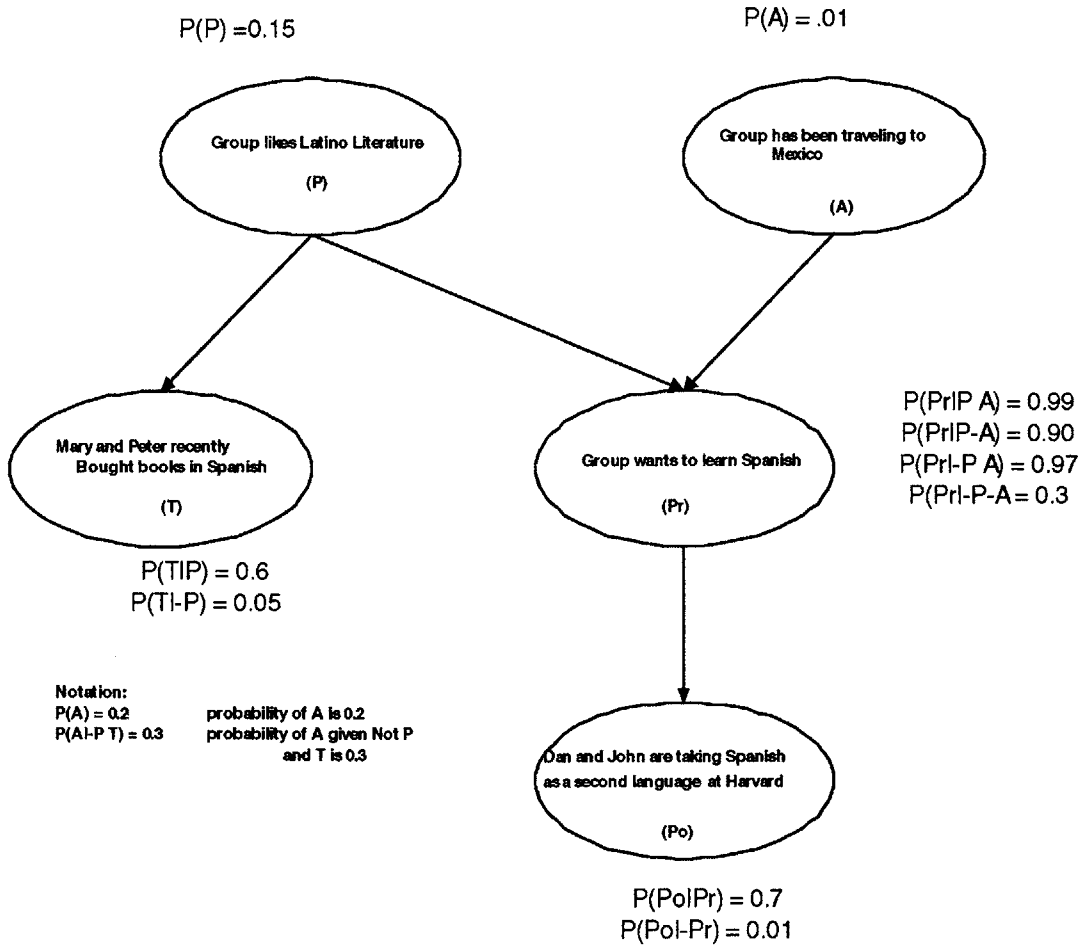
This section reviews some of the techniques used to implement the collaborative filtering and feature-based filtering approaches. In addition, this section ends by providing justifications for this research.

5.1 Collaborative Filtering

As stated before, the collaborative filtering approach is based on the past behaviors of many individuals. The information and relationships from this large group are collected into a knowledge base, which is then used in two steps: clustering and recommendation. Two of the principal algorithms used to implement collaborative filtering agents are the well known K-means clustering and the Nearest Neighbor. K-means clustering uses purchase history to cluster an individual with a group of similar individuals. The level of fitness of the target user to any group is determined by the products that the target user has bought. The recommendations are based on the products that were most frequently purchased by the other individuals of the group [26]. This approach is very simple to implement; however, it does not guarantee that the comparison customers are correctly classified into groups with other customers from their true clusters [3]. On the other hand, Nearest Neighbor is based on correlations to detect similarities between the target user and those of every other individual in the database [3]. The most correlated individuals in the database are used to recommend products to the target user. Nearest Neighbor is one of the most utilized methods for collaborative filtering [60], but there are other techniques being introduced, among them Bayesian networks, vector similarity techniques, and hybrids of Nearest Neighbor and traditional data mining (other recent techniques recently introduced are Latent Semantic Indexing [76], and clustering techniques based on measures of entropy [92] and unsupervised neural networks [76] that capture the mechanisms of Nearest Neighbor and transform high dimensional spaces to low dimensional spaces).

5.1.1. Bayesian Networks

A method of reasoning using probabilities, called Bayesian Networks [30,68], has become popular in the artificial intelligence community. Bayesian networks are directed acyclic graphs (DAGs) (see Figure 5.1), where the nodes are random variables. The random variables can have two values (True and False), or several values (discrete or continuous). The arcs specify the independent assumptions that must hold between the random variables. These independent assumptions determine what probability information is required to specify the probability distribution among the random variables in the network.



P(P) =0.15
P(A) = .01
Group likes Latino Literature
(P)
Group has been traveling to Mexico
(A)
Mary and Peter recently Bought books in Spanish
(T)
Group wants to learn Spanish
(Pr)
P(PrIP A) = 0.99
P(PrIP-A) = 0.90
P(PrI-P A) = 0.97
P(PrI-P-A= 0.3
P(TIP) = 0.6
P(TI-P) = 0.05
Notation:
P(A) = 0.2 probability of A is 0.2
P(AI-P T) = 0.3 probability of A given Not P and T is 0.3
Dan and John are taking Spanish as a second language at Harvard
(Po)
P(PoIPr) = 0.7
P(PoI-Pr) = 0.01

Figure 5.1. Example of a Bayesian network

Figure 5.1 shows an example (inspired by [68]) about a possible Bayesian network to implement a recommender system. This Bayesian network depicts several relationships of a group of students at Harvard University and its activities. This graph could be used to predict what will happen (if the group is traveling to Mexico) or to infer causes from observed effects (if Dan and John are learning Spanish, then the Group probably wants to learn Spanish). Therefore, the smart agent can tailor services to this group (e.g., trips to other Latin American destinations, books about Latin America, books in Spanish, Spanish as a second language services).

To specify the probability distribution of a Bayesian network, one must give the prior probabilities of all root nodes (nodes with no predecessors) and the conditional probabilities of all non-root nodes given all possible combinations of their direct predecessors. The nodes with no predecessors in Figure 5.1 are P and A and the non-root nodes are T, Pr, and Po. It is possible to appreciate that the concept of joint distributions plays a very important role in Bayesian networks. Bayesian networks allow one to calculate the conditional probabilities of the nodes in the network given that the values of some of the nodes have been observed. In addition, the probabilities can come from experts, Delphi cycles, or provided by other agents using different techniques (e.g., Nearest Neighbor, the output of a neural network). For example, if the agent observes that Mary and Peter recently bought books in Spanish (T = True), but Dan and John are not learning Spanish (Pr = false), it is possible to calculate the conditional probability that the group likes Latino Literature given these pieces of evidence (P(**PlT** $\neg$**Pr**)?).

The basic computation in belief networks is the computation of every node's belief (its conditional probability) given the evidence that has been accumulated; however, this computation is NP-hard! "Depending on the particular characteristics of the network, the optimization scheme used, and the care taken in the implementation, Bayesian networks as small as tens of nodes can take too long, or Bayesian networks in the thousands of nodes can be done in acceptable time." Nevertheless, single connected networks (i.e., "a polytree, one in which the underlying undirected graph has no more than one path between any two nodes") can be solved in linear times.

Bayesian networks have been studied by Microsoft Research to implement collaborative filtering agents [6]. Bayesian networks have proven superior to other methodologies such as Bayesian clustering and vector similarity methods for collaborative filtering. We have not found commercial sites that state that they use this methodology to implement the collaborative filtering approach. Nevertheless, Bayesian networks are one of the most recommended methodologies in the future in user modeling.

5.1.2 Vector Similarity

Microsoft Research has also been studying Vector Similarity Techniques to improve the performance of collaborative filtering agents. Vector Similarity (similar to Latent Semantic Indexing) is a technique adapted from the field of information retrieval. The similarity between two documents is often measured by treating each document as a vector of word frequencies and computing the cosine of the angle formed by the two frequency vectors [6]. This formalism is adopted to collaborative filtering, "where users take the role of documents, titles take the role of words, and votes take the role of word frequencies." Votes indicate a positive reference (the target user bought the recommended product). This method was found to have good prediction capabilities; however, Bayesian networks outperform vector similarity in success rate and computation characteristics.

5.1.3. Hybrids of the Nearest Neighbor and Data Mining Techniques

There are also hybrid schemes where techniques such as Nearest Neighbor deal with the clustering decisions; however, a different algorithm handles the recommendation step. Sarwar et al. [77] describe a very interesting system that follows these principles. The data mining technique is used to find associations rules between a set of co-purchased products. Essentially these techniques are concerned with discovering associations between two sets of products such that the presence of some products in a particular transaction implies that products from the other set are also present in the same transaction. These association rules are then used to develop “top-N” recommended products. These techniques do not use the entire population of users to generate the rules but only consider the l neighbors while generating the rules. Therefore, Nearest Neighbor can generate the groups and the recommendation step is performed by the data mining technique.

5.2. Feature-Based Filtering

The feature-based filtering approach does not use information that is based on other individuals. Feature-based filtering attempts to capture the underlying utility structure of the target user. Therefore, the knowledge base consists of the past behavior of the target user. Feature-based filtering approaches are less common on the Web. Companies such as Frictionlesscommerce.com, PersonaLogic.com, and activebuyersguide.com use feature-based filtering that is “constraint based,” using elimination aspects and fixed rule-based systems [27,87] to “narrow the selection of products that meet all of the individuals’ specified criteria in a noncompensatory way.” [3] These implementations are not intelligent (see Section 4 for the definitions of machine intelligence). The implementations at the research level are also non-impressive. Several schemes using influence networks [46], simple reinforcement algorithms [82], and regression [3,36] have been reported in the literature.

5.3. Collaborative Filtering and Feature-Based Filtering

There are several comparisons between collaborative filtering and feature-based filtering. Of those, the recent work of Ariely et al. [3] is the most comprehensive. Ariely et al. [3] examine the performance of collaborative filtering and feature-based filtering using a series of simulations of a marketplace. Two types of collaborative filters are used: one that relies on k-means clustering and the other one in the Nearest Neighbor algorithm. The feature-based filtering agent was implemented using logistic regression. The results of this experiment are very interesting [3]:

1. Feature-based filtering agents learn more slowly initially, but are better in the long run than collaborative filtering when the environment is stable.

2. Feature-based filtering agents are less negatively affected by permanent change in the individual's utility function.

3. Feature-based filtering agents are able to recognize new products (tailored to the target user) faster when these products appear in the marketplace (even before others have purchased these products).

Their results show that feature-based filtering agents outperform in general the collaborative filtering agents and provide much better recommendations. However, the slow learning initially of feature-based filtering agents calls for a parallel approach of both collaborative filtering and feature-based filtering.

Research performed by Good et al. [36] agrees with the results from Ariely et al. [3]. Good et al. [36] emphasize the integration of feature-based filtering agents with collaborative filtering agents. Their rationale to do this is that according to their research collaborative filtering will soon achieve its maximum level of performance. In addition, the hybrid model of feature-based filtering and collaborative filtering outperformed the feature-based filtering agents (" a small, but statistically significant improvement in accuracy"). They also found that feature-based filtering agents have better performance than collaborative filtering; however, these results are subject to the performance of the algorithm utilized to implement the feature-based filtering agent.

5.4 Summary and Justification

It can be concluded from this literature survey that there is no a unified and "standard" scheme for intelligent recommendation systems. The current schemes to implement collaborative filtering and in particular feature-based filtering are still under research and development. The conclusions from our literature survey indicate the following:

1. The implementations of collaborative filtering are efficient; however, these implementations show a low level of machine intelligence (learning, responsiveness, and adaptation). There is not yet a final answer about the best framework for collaborative filtering; researchers are still looking for new ways and more powerful algorithms to adapt to collaborative filtering and obtain major improvements.

2. Results indicate that collaborative filtering will soon reach its maximum level of performance; therefore, it is opportune to search for new ways to achieve the next levels of performance required by electronic commerce.

3. Recent research has indicated that feature-based filtering (using simple algorithms) in general outperform collaborative filtering. Therefore, the next research frontier to improve is feature-based filtering.

4. Research on collaborative filtering thinks that a possible way to enhance the current collaborative filtering agents is by introducing some elements from feature-based filtering. Therefore, this supports the idea that research in feature-based filtering also contributes to enhance collaborative filtering.

5. The commercial implementations of feature-based filtering are very primitive. These implementations do not show any signs of machine intelligence and they exhibit low performance (and shift the effort to the target user!). Therefore, it is important to conduct research in algorithms to provide machine intelligence to feature-based filtering agents.

6. Research in feature-based filtering still lags behind that of collaborative filtering.

This current climate is conducive to research directed at providing higher levels of machine intelligence to feature-based filtering agents. There are recent approaches such as Support Vector Machines, recent modifications to "standard" neural networks (e.g., Backpropagation) using more powerful algorithms, and the notion of ensembles of neural networks (e.g., Fuzzy ARTMAP ensembles) that mitigate some of the architectural and learning problems. In the following sections, these machine learning algorithms are applied to feature-based filtering and tested in a representative case study.

6 Neural Networks and Support Vector Machines

The most common learning system is the supervised learning system. Supervised learning systems can be used to classify patterns, by letting the inputs be the patterns to be classified and the desired outputs be the correct classifications. This thesis will involve the use of supervised learning systems as represented by neural networks and support vector machines.

6.1 Neural Networks

Neural networks are information processing systems motivated by the goals of reproducing the cognitive processes and organizational models of neurobiological systems. By virtue of their computational structure, neural networks feature attractive characteristics such as graceful degradation, robust recall with fragmented and noisy data, parallel distributed processing, generalization to patterns outside of the training set, nonlinear modeling capabilities, and learning.

The specific characteristics of a neural network depend on the paradigm utilized. The paradigm is determined by the architecture and the neurodynamics employed. The architecture defines the arrangement of the neurons and their interconnections (see Figures 6.1 and 6.2). The neurodynamics specifies how the inputs to the neurons are going to be combined together (i.e., short term memory), what type of function or relationship is going to be used to develop the output, and how the adaptive coefficients, also called weights (i.e., long term memory), are going to be modified.

The learning mechanism which handles modifications to the weights can be classified under supervised, unsupervised, and reinforcement learning. Supervised learning takes place when the network is trained using pairs of inputs and desired outputs. In unsupervised learning, the network is able to self-organize the categories. Reinforcement learning adds feedback to unsupervised learning to evaluate the pattern classification process.

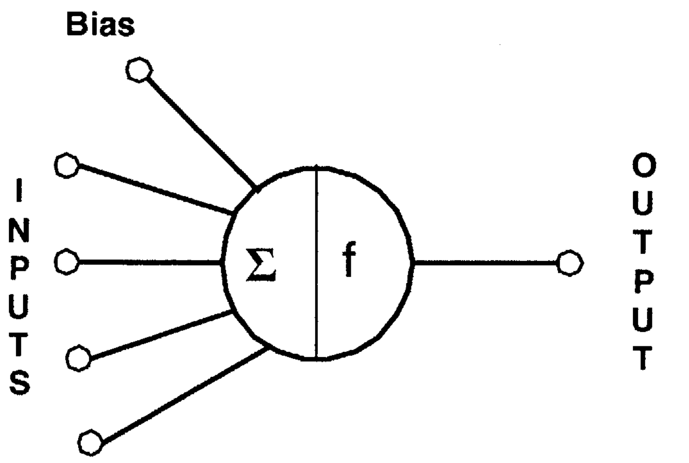


Figure 6.1. A single neuron of a typical neural network

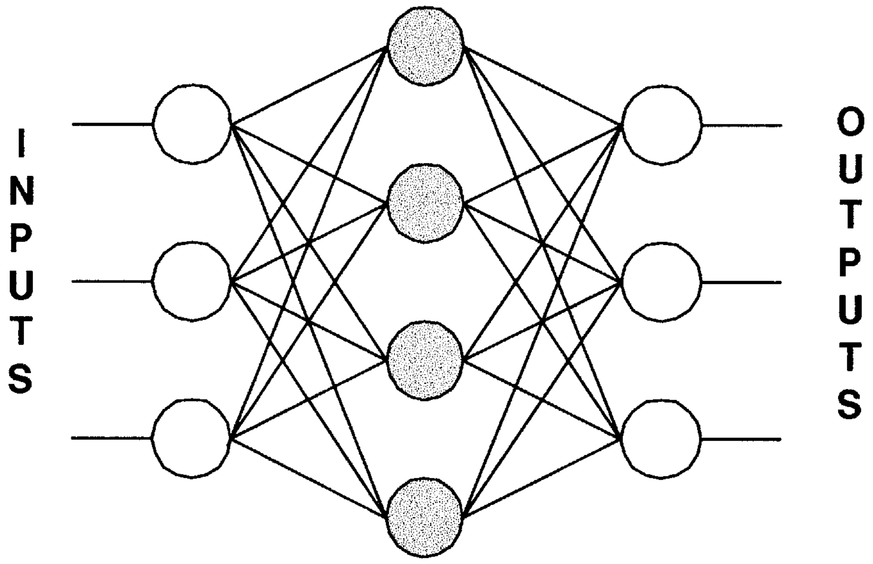


Figure 6.2. Three layer neural network with 3 inputs, four hidden units, and 3 outputs

In supervised learning, we try to adapt a neural network so that its actual outputs come close to the target outputs for some training set. The goal is to adapt the parameters of the network so that it performs well for samples from outside of the training set. Paul Werbos [99] has stated that there are two forms of the supervised learning task: real-time and non-real time. In non-real-time supervised learning, the order of the patterns is not assumed to be meaningful. "One simply cycles through the training set as often as one likes, adjusting the weights through finer and finer tuning, until the estimates of the weights converge." In real time learning, "one can cycle through the data one observation at a time, and one has to take the data as they come, adapting the weights in real time operation."

There are two types of supervised neural networks, which are useful in practice [99]:

1. Supervised neural networks based on minimizing errors (e.g., Backpropagation);

2. Supervised neural networks based on forecasting by analogy. Designs in class 2 are called "heteroassociative" memories. In addition, there are many designs available for preprocessing the input vector, most of which involve: (1) fixed preprocessing, such as Fourier transforms or breaking the data up into regions (as in "CMAC," Radial Basis Functions (RBF)); (2) unsupervised clustering or feature extraction methods, such as Adaptive Resonance Theory (ART), topological maps, and mean-field projections.

We have selected two neural network paradigms, which have had numerous successful applications, for this thesis: Backpropagation (type: minimizing errors) [73,99] and Fuzzy ARTMAP (type: forecasting by analogy) [21,22].

6.2 Backpropagation

The Backpropagation paradigm [73,99] learns adequate internal representations using deterministic neurons to provide a mapping from input to output (See Figure 6.3). This procedure involves the calculation of a set of output vectors **O** using the current weights matrix **W** (a set composed of all matrixes $\mathbf{W_m}$, $\mathbf{m = 2 \ldots l}$, where $\mathbf{W_2}$ would be the matrix of weights between the input and the first hidden layer and $\mathbf{W_l}$ the matrix of weights between the last hidden layer and the output layer) and the input vectors **I**. The error is estimated by comparing **O** with the target vector **T** and using an error function. This error function is defined for a specific $\mathbf{I_i}$ and $\mathbf{T_i}$ as follows:

$$E_k = 1/2 \, \Sigma_i (t_i - o_{il})^2$$

where **k** is an input vector-target output relationship that conforms the input vector set **I** and target output vector set **T**. **i** represents the output nodes of the output layer in the network, and **l** is the total number of layers (i.e., layer **l** is the output layer, and layer **number 1** is the input layer). $\mathbf{t_i}$ is the targeted output for the **ith** output node and $\mathbf{o_{i_1}}$ is the response obtained from the **ith** output node using the corresponding $\mathbf{I_k}$.

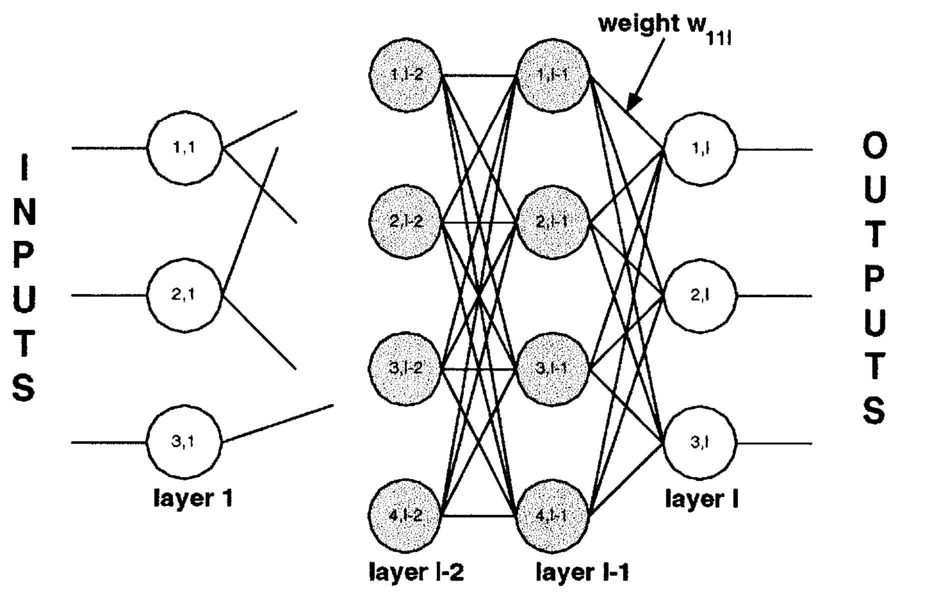


Figure 6.3. A multilayer neural network (l layers) with 3 input neurons, several hidden layers (from 2 to l-1), and 3 output neurons

Thus the total error will be determined as:

$$E = \Sigma_k E_k$$

The learning procedure minimizes $\mathbf{E_k}$ by performing steepest descent and therefore obtaining appropriate **W** and **Ø**.

The net input to a neuron is expressed as

$$net_{im} = \Sigma_j \, w_{ijm} \, O_{jm-1}$$

where $\mathbf{w_{ijm}}$ is the weight between the **jth** unit of layer **m-1** and the **ith** unit of layer **m.** In addition, the traditional activation function utilized is the logistic function given by

$$\mathbf{O_{im} = 1/(1 + e^{-net_{im}})}$$

To minimize $\mathbf{E_k}$ and achieve a convenient **W,** it is necessary to make adjustments to previous **W** obtained until the error tolerance imposed by the final desired mapping accuracy is accomplished. Therefore, it is possible to establish

$$\Delta \mathbf{w_{ijm}} \;\alpha - \partial \mathbf{E_k}/\partial \mathbf{w_{ijm}}.$$

Then $\partial \mathbf{net_{im}}/\partial \mathbf{w_{ijm}}$ is

$$\partial \mathbf{net_{im}}/\partial \mathbf{w_{ijm}} = \partial(\Sigma_j\, \mathbf{w_{ijm}\, o_{jm-1} + \varnothing_{im}})/\partial \mathbf{w_{ijm}}$$

$$\partial \mathbf{net_{im}}/\partial \mathbf{w_{ijm}} = \mathbf{o_{jm-1}}$$

Then $\partial \mathbf{O_{im}}/\partial \mathbf{net_{im}}$ is expressed by

$$\partial \mathbf{O_{im}}/\partial \mathbf{net_{im}} = \partial(\mathbf{1/(1 + e^{-net_{im}})})/\partial \mathbf{net_{im}}$$

$$\partial \mathbf{O_{im}}/\partial \mathbf{net_{im}} = \mathbf{e^{-net_{im}}/(1 + e^{-net_{im}})^2}$$

$$\partial \mathbf{O_{im}}/\partial \mathbf{net_{im}} = \mathbf{O_{im}\,(1 - O_{im})}$$

The partial derivative of $\mathbf{E_k}$ with respect to the weights and biases could be defined by:

$$\partial \mathbf{E_k}/\partial \mathbf{w_{ijm}} = (\partial \mathbf{E_k}/\partial \mathbf{net_{im}})\,(\partial \mathbf{net_{im}}/\partial \mathbf{w_{ijm}})$$

and the partial derivative of the error to the net input could be stated as:

$$\partial \mathbf{E_k}/\partial \mathbf{net_{im}} = -\delta_{\mathbf{im}}.$$

The following terms are derived by replacing the previous relationships:

$$\partial \mathbf{E_k}/\partial \mathbf{w_{ijm}} = -\delta_{\mathbf{im}}\, \mathbf{o_{jm-1}}.$$

The variable δ defined above could be calculated by back propagating the error through the network starting with the output layer where the partial derivative of the error to the output is defined as:

$$\partial E_k/\partial o_{il} = \partial(1/2 \, \Sigma_i (t_i - o_{il})^2) / \partial o_{il}$$

$$\partial E_k/\partial o_{il} = -(t_i - o_{il})$$

and δ_{il} (output layer) is

$$\delta_{il} = -(\partial E_k/\partial o_{il}) \, (\partial o_{il}/\partial net_{il})$$

$$\delta_{il} = -(t_i - o_{il}) \; o_{il} \, (1 - o_{il})$$

and the adjustments are

$$\Delta w_{ijm} = \eta \, \delta_{il} \, o_{i_1 - 1}$$

where η is the step size (also called the learning rate).

The lower layers can be concluded as follows:

$$\delta_{im} = -(\partial E_k/\partial o_{im}) \, (\partial o_{im}/\partial net_{im})$$

$$\delta_{im} = -(\partial E_k/\partial o_{im}) \; o_{im} \, (1 - o_{im})$$

$$\delta_{im} = -\Sigma_j \, (\partial E_k//\partial net_{jm+1}) \, (\partial net_{jm+1}/\partial o_{im}) \; o_{im} \, (1 - o_{im})$$

$$\delta_{im} = \Sigma_j \, (\delta_{jm+1} \, w_{jim+1}) \; o_{im} \, (1 - o_{im}).$$

Consequently the adjustments for Δw are equal to

$$\Delta w_{ijm} = \eta \, \delta_{im} \, o_{jm-1}$$

Training of a Backpropagation neural network is achieved through a sequence of iterations, or epochs. An epoch is a pass through the entire training set.

In spite of the capabilities of this proven algorithm, the rate convergence might be very slow. One of the most commonly utilized heuristics is a momentum factor (μ)

that weights the contribution of past $\Delta\mathbf{W}$. The "updating" equations will be modified as follows [73]:

$$\mathbf{W}_{ijm}(t) = \mathbf{W}_{ijm}(t-1) + \Delta\mathbf{W}_{ijm}(t) + \mu\,\Delta\mathbf{W}_{ijm}(t-1)$$

The training process using standard Backpropagation is a very difficult problem. One needs to find an appropriate architecture (e.g., the number of hidden units, the number of hidden layers), adequate size of training data, a satisfactory input scheme, satisfactory initialization (e.g., initial weights: it is recommended to try several different initial sets of weights in order to ensure that a global minimum has been obtained!), and learning parameters. In this research several alternative approaches were tested and utilized, in order to select an appropriate architecture and to achieve fast convergence.

6.2.1 Architecture Selection

It is important to select the right architecture (i.e., the number of hidden neurons) of a Backpropagation neural network. It is known from theoretical developments [61,84] and from empirical results [49,69] that the generalization ability (**GA**) of a neural network depends on a balance between the information in the training examples and its complexity (i.e., weights and hidden neurons). On the other hand, it is also clear that neural networks with too few weights will not have the capacity to represent the information accurately. Several schemes were tested in this thesis to decide the best architecture.

6.2.1.1 The Traditional Method

Traditionally in supervised neural networks GA is defined as the expected performance on future data and can be approximated by the expected performance on a finite test set (i.e., new observations that were not used in getting **W**). Therefore, several architectures (different hidden units and hidden layers) are "trained" and the one with the best GA is selected. This method is effective especially when there are enough data samples (i.e., a very large number). Unfortunately, in the problem faced by us, there are not enough observations to calculate **GA**. Therefore we decided not to use the traditional method.

6.2.1.2. Cost Functions with Two or more Terms

One methodology to get an adequate architecture is to minimize a cost function composed of two or more terms: the ordinary training error, plus some measure of network complexity. Two different schemes were tested initially in this thesis:

1. **Penalizing Small Weights**. Elimination of weights was performed as described by Weigend et al. [96,97]. This technique has the following objective function

   $$\Sigma_k E_k + \lambda \Sigma_i ((w_i/w_o)^2 / (1 + (w_i/w_o)^2)$$

   where the first term is the sum of the squared error for the training set and the second term is a cost for each weight **i**. The parameter λ represents the relative importance of the cost term with respect to the error term (usually $\lambda << 1.0$), $\mathbf{w_o}$ is the scale parameter that allows us to express a preference for fewer large weights ($\mathbf{w_o}$ is relatively small). Again, gradient-descent is utilized. This technique has been used successfully to achieve networks with a high degree of generalization when the problem is very non-linear and the training data has a high level of noise. However, slow convergence is a potential problem and makes it difficult to find the right schedule of values for λ.

2. **Penalizing Large Weights**. This methodology limits the growth of the weights. It is realized by adding a term to the cost function that penalizes large weights,

   $$\Sigma_k E_k + (1/2) \lambda \ \Sigma_i w_i^2$$

   where the first term is the sum of the squared error for the training set and the second term (the regularization parameter) is the cost for each weight **i**. The regularization parameter λ represents how strongly large weights are penalized. If gradient-descent is used, a new term is added to the weight update of the traditional Backpropagation algorithm:

$$\Delta w_{ijm} \ \alpha \ (- \partial E_k / \partial w_{ijm} - \lambda \ w_{ijm}).$$

Krogh and Hertz [48] have analyzed the effect of this term both theoretically and experimentally. They have concluded that this term of weight decay can improve generalization (with the assumption "that the neural network could be expanded around an optimal weight vector, and therefore it is strictly only valid in a little neighborhood around that vector"). Slow convergence is a potential problem. In addition, there are no references to problems with small data sets and about how to determine an optimal regularization parameter.

6.2.1.3 Dividing the Data into Three Parts

The method of dividing the data into three parts (i.e., a training set for obtaining the weights, a validation set for deciding when to stop training, and a testing set to compare performance) can also build neural networks with good performance [63]. The validation set (the error on the validation set is monitored during the training process) indicates when training should be terminated (i.e., the error on the validation set will begin to rise) before reaching a minimum training set error in order to prevent overfitting. Then, the GA is estimated by using the testing set. Again, this method is useful when the dataset is very large. We were not able to use this technique.

6.2.1.4 Prediction Risk

A very realistic assumption is that there are not enough historical data to learn the behavior and preferences of a user. The case study used by us (and explained in Section 7) does not have enough observations. Therefore, we have to estimate GA from the available data. There are three methodologies available to calculate GA for this situation (and then proceed to compare different architectures based on that result):

1. **Crossvalidation (CV).** CV is a sample re-use method that can be used to estimate GA [62]. CV makes minimal assumptions on the statistics

of the data. Each instance of the database in turn is left out, and the supervised neural network is trained on all the remaining examples (N – 1). The results of all n judgments, one for each member of the dataset, are averaged, and that average represents the final error estimate (i.e., GA). This approximation of GA is very expensive to calculate using Backpropagation neural networks (i.e., it involves constructing N neural networks, each trained with N – 1 samples). Geiser [34] has introduced a variation of the method, denominated v-fold cross validation. Instead of leaving out only one observation, larger subsets of the training data are deleted. The training data are divided into v randomly selected disjoint subsets of roughly equal size. Then CV for subset j is defined as the calculation of the Average Squared Error (ASE) for the subset j with the neural network trained with the remaining subsets. The same is performed for each of the individual subsets. GA is then approximated by the average of the CVs. Typical choices for v are 5 and 10. This has become the standard test in practical terms for machine learning techniques [47,62,100].

2. **Generalized cross-validation (GCV).** GCV combines ASE and a measure of complexity of the network trained with the entire dataset [93]. GCV is used to select architectures in linear models. Moody and Utans [62] have demonstrated its modifications used with success in some neural network problems. The equation to calculate GA for a particular neural network using a dataset of N examples is:

$$\mathbf{GA = ASE * (1/(1 - (\#\ of\ Weights/N))^2)}$$

It is very clear that we cannot use GCV when N is a number even lower than the number of weights. That could be possible in our problem. Therefore, we cannot use GCV.

3. **Final prediction error (FPE).** Akaike's FPE [1,4] combines the average training squared error (ASE) with a measure of the neural

network complexity (the number of weights) trained with the entire dataset. FPE is also utilized to select architectures in linear models. There are some successes in its extensions to select architectures for Backpropagation neural networks [61,62]. The equation to calculate GA for a particular Backpropagation neural network using a dataset of N examples is:

$$\mathbf{GA = ASE * ((1 + \#\ of\ Weights/N))/(1 - (\#\ of\ Weights/N))}$$

FPE has the same problem when N is a number smaller than the number of weights. Therefore, we cannot use FPE.

6.2.1.5 Bayesian Neural Networks

MacKay's Bayesian evidence framework [54,55,56] can be used to control overfitting in neural networks. This framework is able to determine the optimal regularization parameters in an automated fashion. This is achieved by interpreting the Backpropagation learning process as probabilistic. For instance, the error function is interpreted as defining the probability distribution of a noise model:

$$\mathbf{P(Data\ Set | W, \beta, \mathcal{H}) = (1/Z_D)\, e^{(-\beta \Sigma_k E_k)}}.$$

"Thus, the use of the sum-squared error corresponds to an assumption of Gaussian noise on the target variables, and the parameter $\boldsymbol{\beta}$ defines a noise level $\boldsymbol{\sigma^2=1/\beta}$." [56] The regularization parameter is represented in terms of a log prior probability distribution over the parameters:

$$\mathbf{P(W | \lambda, \mathcal{H}) = (1/Z_D)\, e^{(-\lambda\ (1/2) \Sigma i\ wi2)}}.$$

$\boldsymbol{\mathcal{H}}$ is a probabilistic model that defines the form of the Backpropagation neural network, and the objective function **M(W)** is the inference of **W** given the training data (following Bayes' theorem):

$$\mathbf{P(W|Data\ Set,\lambda,\beta,\mathcal{H}) = P(Data\ Set|W,\beta,\mathcal{H})\ P(W|\lambda,\mathcal{H}) / P(Data\ Set|\ \beta,\lambda,\mathcal{H})}.$$

$$\mathbf{P(W\ |Data\ Set, \lambda, \beta, \mathcal{H}) = (1/Z_m)\, e^{(-M(W))}}.$$

W is found by minimizing **M(W).** This **W** vector is then interpreted as the most probable. There are several algorithms to optimize **M(W),** such as the traditional approach to Backpropagation learning, optimization schemes with Gaussian approximation, and Monte Carlo simulation.

The Bayesian framework offers several potential advantages such as:

1. All the available data can be used to both model training and model comparison.

2. The regularization parameters are optimized online.

6.2.2 Faster Convergence

The basic Backpropagation algorithm using gradient-descent and its modification using momentum may be slow for some problems. We considered the use of other algorithms that can converge faster. These algorithms are more sophisticated from an optimization viewpoint.

6.2.2.1 Levenberg-Marquardt

This algorithm is designed to approach second-order training speed. It approximates the Hessian matrix **H** by using the Jacobian matrix **J** (which contains the first derivatives of the network errors with respect to the weights) as follows [32,42]:

$$\mathbf{H} = \mathbf{J}^T\mathbf{J},$$

And the gradient can be computed as

$$\mathbf{J}^T\mathbf{e}$$

where **e** is a vector of the neural network errors. Therefore, the update is modified to be:

$$\mathbf{W(t+1)} = \mathbf{W(t)} - [\ \mathbf{J}^T\mathbf{J} + \beta\ \mathbf{I}\ ]^{-1}\ \mathbf{J}^T\mathbf{e}$$

where β is a constant that is decreased or increased depending on the performance function. Levenberg-Marquardt is considered one of the fastest algorithms for training Backpropagation neural networks. However, problems with a large data set are difficult to avoid due to computer memory requirements.

6.2.2.2 Conjugate-Gradient

Conjugate-gradient is a family of algorithms that adjusts the search direction to produce the fastest convergence. Initially, the direction of the search is on the steepest descent path. After the first iteration, a line search is performed to determine the optimal distance (**p**) to move along the current search direction as stated as follows:

$$\mathbf{W(t+1) = W(t) + \eta\ p(t)}$$

The next search direction is determined so that it is conjugate to previous search directions. Conjugate directions produce generally faster convergence than gradient-descent directions. The procedure is given by:

$$\mathbf{p(t) = -\ \partial E/\partial w\ (t) + \beta(t)\ p(t-1)}$$

where $\beta(t)$ is calculated in different forms. For example, in Fletcher-Reeves [32,33] (used in this thesis), the procedure to calculate $\beta(t)$ is the ratio of the norm squared of the current gradient (**g**) to the norm squared of the previous gradient:

$$\mathbf{\beta(t) = g^T(t)\ g(t) / (g^T(t-1)\ g(t-1)).}$$

The family of conjugate-gradient algorithms is considered one of the best methodologies to train Backpropagation neural networks. These algorithms have a high rate of convergence in difficult problems. However, the Levenberg-Marquardt algorithm is considered faster for problems of moderate size [32].

6.3 Fuzzy ARTMAP

Adaptive resonance theory (ART) represents a family of neural networks which self-organize categories in response to arbitrary sequences of input patterns in real time for pattern recognition. A class of these networks called ART 1, which is unsupervised, can be used only for binary patterns [10,13,16]. ART 2, which is also an unsupervised class, responds to both binary and analog patterns [11,14,15]. The class ART 3 features an advanced reinforcement feedback mechanism that can alter the classification sensitivity or directly engage the search mechanism [17]. The class Fuzzy ART [19] is similar in architecture to ART 1; however, fuzzy operators [102] are added in order to handle analog patterns without losing the advantages of ART 1 architecture. The class ARTMAP ("predictive" ART) is built upon the basic ART designs, while incorporating supervision in the learning process [18]. ART 2-A ("algorithmic" ART) [20] is a special case of ART 2 which emphasizes the intermediate and fast learning rates, hence accelerating the learning process by three orders of magnitude. Fuzzy ARTMAP is built upon Fuzzy ART and is very similar in architecture to ARTMAP; however, its utilization of Fuzzy ART allows it to handle analog patterns [21,22]. Recent introductions to the ART family are Distributed ART (dART) [12] and Distributed ARTMAP (dARTMAP) [25]. dART and dARTMAP are able to learn distributed code representations.

Fuzzy ARTMAP is considered a very powerful neural network for pattern classification [23,24,51,59]. Fuzzy ARTMAP includes a pair of Fuzzy ART modules. Therefore, we will explain first the Fuzzy ART paradigm. After that, Fuzzy ARTMAP will be introduced. Finally, the Fuzzy ARTMAP *voting strategy* will be presented.

6.3.1 Fuzzy ART

Fuzzy ART [21,22] incorporates the basic architecture and neurodynamics of ART systems. The notation here will largely follow that of [21]. Each ART system includes:

1. a field $\mathbf{F_0}$ of neurons that represents an input vector;

2. a field $\mathbf{F_1}$ that receives bottom-up input from F_0; and top-down input from

3. a field $\mathbf{F_2}$ that represents the active category (see Figure 6.4). A vector $\mathbf{W_j}$ $(\mathbf{w_{j1}},\ldots,\mathbf{w_{jM}})$ of weights is associated with each F2 neuron (i.e., category). Each category is initially uncommitted (and the initial

values of the weights are set to 1). After a category is selected for learning it becomes committed.

The F_0 input vector is denoted $\mathtt{I}$ ($\mathtt{I}_1,\ldots,\mathtt{I}_M$, with each component $\mathtt{I}_i$ in the interval [0,1], i = 1, ..., M). The F_1 activity vector is denoted **X** ($\mathbf{X}_1,\ldots,\mathbf{X}_M$) and the F_2 activity vector is denoted **Y** ($\mathbf{Y}_1,\ldots,\mathbf{Y}_N$). The number of neurons in each field is arbitrary for explanation purposes.

Fuzzy ART is designed as a generalization of ART 1 (ART 1 only handles binary inputs). However, the set theory intersection operator ($\cap$) of ART 1 is replaced by the fuzzy set theory conjunction ($\wedge$). The operator (fuzzy AND) as defined by Zadeh [102] for two vectors **X** ($\mathbf{x}_1,\ldots,\mathbf{x}_n$) and **Y** ($\mathbf{y}_1,\ldots,\mathbf{y}_N$) is

$$\mathbf{X} \wedge \mathbf{Y} = \min(x_i, y_i).$$

This fuzzy operator makes Fuzzy ART capable of handling both analog and binary data. Some basic mechanisms of Fuzzy ART neurodynamics are explained as follows:

a) **Category Choice.** For each Input Vector $\mathtt{I}$ and category **j**, the choice function T_j is defined by

$$T_j(I) = |I \wedge w_j| \; / |w_j|$$

where **| |** is the **L¹** norm (i.e., $|I \wedge w_j| = \Sigma_i |\min(I_i, w_{ji})|$). The maximum T_j $(j = J)$ is defined as the winner category (also called category choice) when at most one F2 neuron can become active at a given time (in the implementation for this thesis, the first category chosen will always be the category whose weight vector $\mathbf{W}_J$ is the largest fuzzy subset of the input vector $\mathtt{I}$). If more than one T_j is maximal, the category **j** with the smallest index is chosen. In particular, neurons become committed in the order **j** = 1, 2, 3, ... When the **jth** category is chosen, $\mathbf{Y}_j = 1$; and $\mathbf{Y}_j = 0$ for $\mathbf{j} \neq \mathbf{J}$. In a choice system, the F1 activity vector **X** is equal to $\mathtt{I}$ if F2 is inactive and equal to $I \wedge w_J$ if the $\mathbf{J}^{th}$ F2 neuron is chosen.

Learning of a category occurs if the match function $|I \wedge w_J|/|I|$ of the chosen category meets the vigilance criterion (ρ):

$$|I \wedge w_J|/|I| >= \rho.$$

That means that learning occurs depending on the degree to which I is a fuzzy subset of w_J. On the other hand, a mismatch reset occurs if

$$|I \wedge w_J|/|I| < \rho.$$

Then the value of the choice function T_J is set to 0 for the duration of the input presentation to prevent the persistent selection of the same category during search. A new index **J** is then chosen. The search process continues until the chosen **J** satisfies ρ.

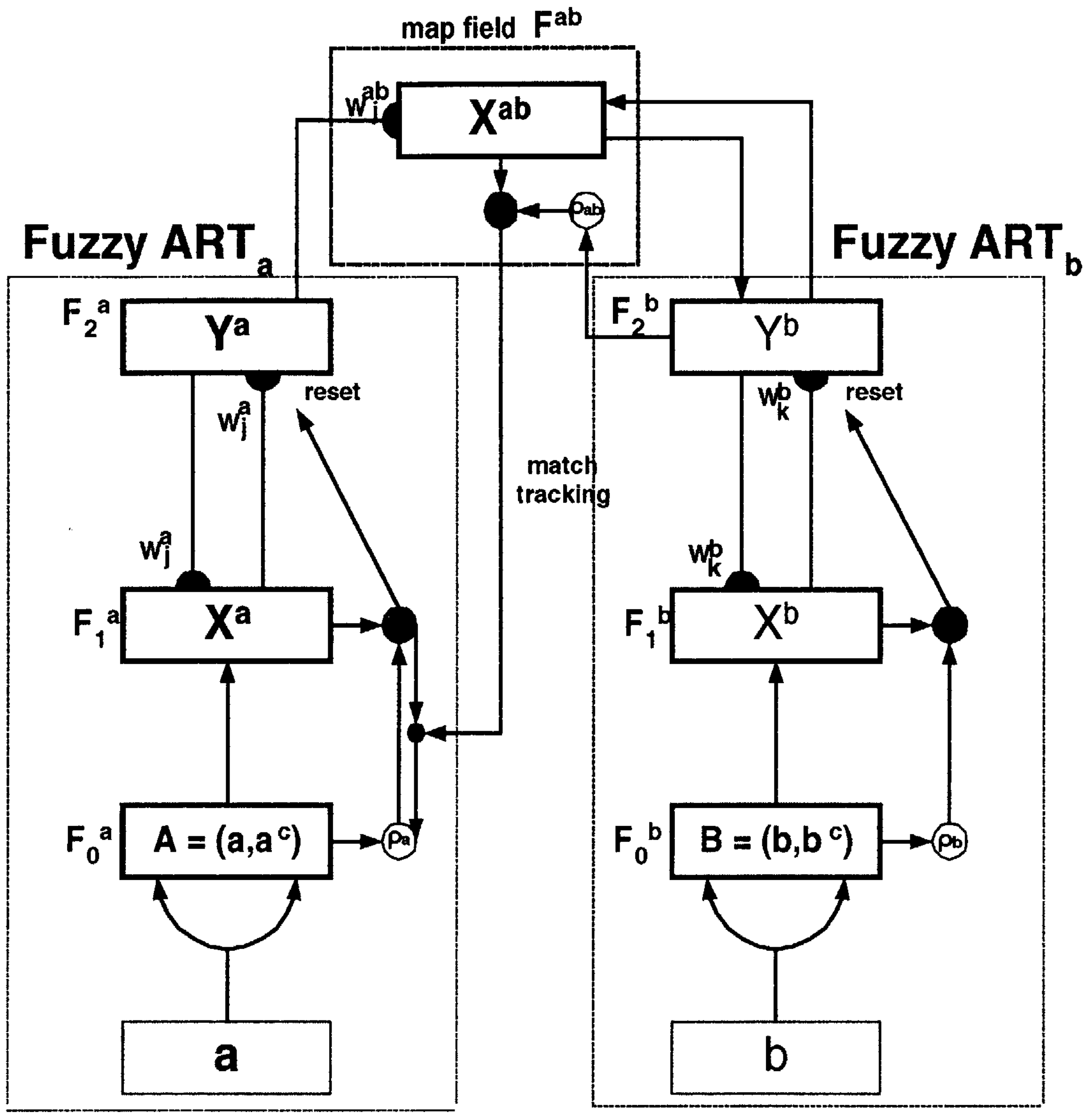


Figure 6.4. Fuzzy ARTMAP architecture using second complement coding. Fuzzy ARTMAP is formed by two Fuzzy ART modules: Fuzzy ART$_a$ and Fuzzy ART$_b$.

Once the search ends, **W$_J$** is updated according to the equation

$$w_J^{(New)} = ß\ (I \wedge w_J^{(old)}) + (1 - ß)\ w_J^{(old)}.$$

Fast learning (i.e., approximation to one-shot learning) corresponds to setting **ß** equal to 1. This is a “far” reflection of the membrane equations [44] as described in earlier papers of Grossberg [38,39,40,41] and used in the other ART neural networks such as ART1 and ART2.

b) **Complement Coding.** A very important issue is that, in Fuzzy ART, the inputs have to be normalized in order to avoid the proliferation of categories. Complement coding has been found to be the best normalization rule to preserve amplitude information. As stated by Carpenter et al. [21] "complement coding represents both the on-response and the off response to an input vector **a**. To define this operation in its simplest form, let **a** itself represent the on-response. The complement of **a**, denoted by $\mathbf{a}^c$, represents the off-response, where $\mathbf{a}_i^c = 1 - \mathbf{a}_i$." Therefore, the complemented coded input **I** to Fuzzy ART is the 2M-dimensional vector

$$\mathbf{I} = (\mathbf{a},\mathbf{a}^c) = (a_1,\ldots,a_M, a_1^c, \ldots,a_M^c).$$

c) **Geometric Interpretation of Fuzzy ART Learning.** We will adopt a simple vector **a** (following [21]) of 2 dimensions normalized using complement coding. The original vector **a** is

$$\mathbf{a} = a_1, a_2$$

and the normalized input will be:

$$\mathbf{I} = (\mathbf{a}, \mathbf{a}^c) = (a_1, a_2, 1 - a_1, 1 - a_2).$$

For example if $\mathbf{a} = 0.6, 0.3$, then $\mathbf{a}^c = 0.4, 0.7$. Each category **j** (as represented by the respective neuron in $\mathbf{F_2}$) has a geometric interpretation as a rectangle $\mathbf{Rectangle_j}$. The weight vector $\mathbf{W_j}$ (from $\mathbf{F_1^a}$ to the respective neuron in $\mathbf{F_2^a}$ and vice versa) can be written in complement coding form:

$$\mathbf{Wj} = (\mathbf{u_j}, \mathbf{v_j^c})$$

Following the explanations of [21], $\mathbf{u_j}$ and $\mathbf{v_j}$ are 2-dimensional vectors. $\mathbf{u_j}$ labels one corner of a rectangle $\mathbf{Rectangle_j}$ and $\mathbf{v_j}$ identifies another corner (see Figure 6.5). In the implementation of Fuzzy ART for this thesis, $\mathbf{W_J}^{(\mathbf{new})}$ will be equal to the input $(\mathbf{a},\mathbf{a^c})$ when **J** is an uncommitted node (i.e., it has not been selected previously). The corners of $\mathbf{Rectangle_j^{(new)}}$ are then given by vector **a** (i.e., $\mathbf{Rectangle_j^{(new)}}$ is point **a**). The learning process increases the size of each rectangle. Actually, during learning, the size of $\mathbf{Rectangle_j}$ grows as the size (magnitude) of $\mathbf{w_j}$ reduces (as a result of successive operations using the fuzzy

AND operator of the learning equation), and the maximum size of **Rectangle**$_j$ is determined by ρ. During each learning trial, **Rectangle**$_j$ increases to include other vectors (if the configuration of F_2 and ρ allow it - see Figure 6.6). The corners of the new **Rectangle**$_j$ are given by $\mathbf{a} \wedge \mathbf{u}_j$ and $\mathbf{a} \vee \mathbf{v}_j$, where $\wedge$ is the fuzzy AND operator and **v** is the fuzzy OR operator [102]. However, reset leads to another category choice if **Rectangle**$_j$ has already reached the maximum size permitted. "In summary, each **Rectangle**$_j$ equals the smallest rectangle that encloses all vectors **a** that have chosen category **j**." [21]

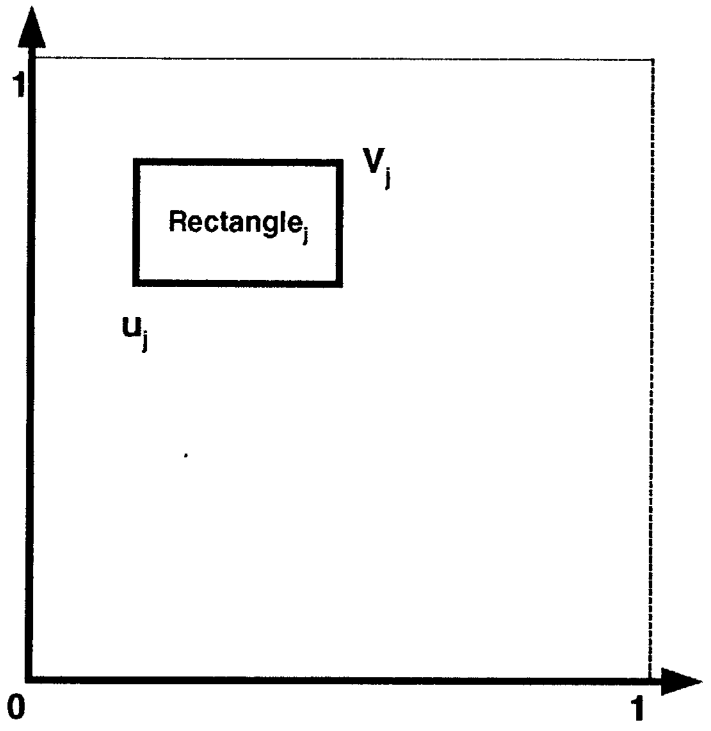


Figure 6.5. Geometric representation of a weight vector for a 2-dimensional system in Fuzzy ART using second complement normalization. The learning process increases the size of each rectangle.

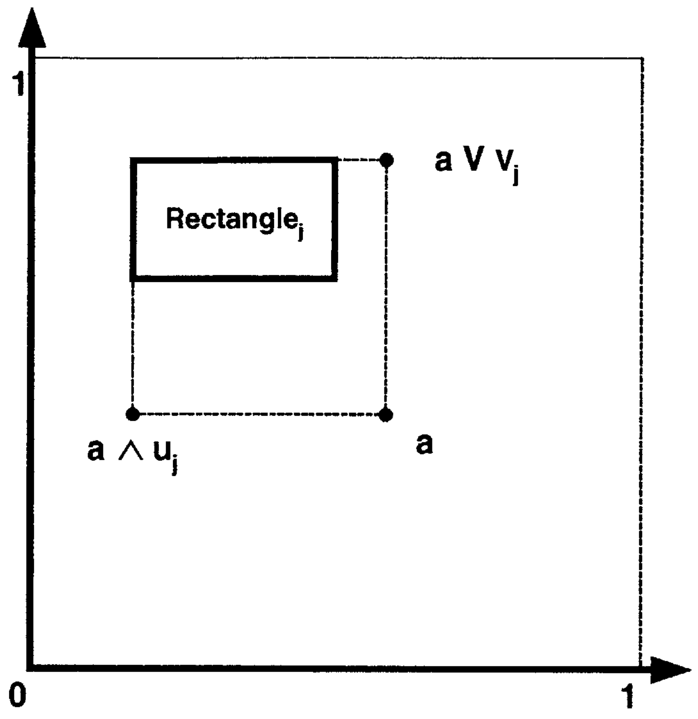


Figure 6.6. The rectangles increase during learning to include the different vectors that meet the criteria set by ρ (in this case, vector a is added to a previously "committed" neuron in F_2 and represented by Rectangle$_j$). The corners of the new Rectangle$_j$ are formed by the following fuzzy operations: Fuzzy OR (a, v_j) and Fuzzy AND (a, u_j).

6.3.2 The Fuzzy ARTMAP Paradigm

Fuzzy ARTMAP has two Fuzzy ART modules **Fuzzy ART$_a$** and **Fuzzy ART$_b$** (see Figure 6.4).These two modules are linked together via an inter-ART module **F^{ab}** called a "map field." Fuzzy ARTMAP uses the hyper-rectangles of Fuzzy ART to represent category weights in a supervised learning fashion. The neurodynamics allows the weights to be updated when an input correctly predicts the output. In addition, a mechanism called match tracking is used to reorganize category structure to eliminate predictive errors during learning. The major features of Fuzzy ARTMAP are:

a) **Second Complement Normalization.** In the implementation of Fuzzy ARTMAP for this thesis, second complement normalization is used (see Figure 6.4). The input vector for **Fuzzy ART$_a$ A** is composed of **a** and **a^c** and the input vector for **Fuzzy ART$_b$** (or supervised target/output) **B** is composed of **b** and **b^c**.

b) **Map Field Activation**. The map field F^{ab} is activated whenever one of categories of **Fuzzy ART$_a$** or **Fuzzy ART$_b$** is active. If node **J** of F_2^a is chosen, then its weights W_j^{ab} activate F^{ab}. "If neuron **K** in F_2^b is active, then the node **K** in F^{ab} is activated by 1-to-1 pathways between F_2^b and F^{ab}." If both **Fuzzy ART$_a$** and **Fuzzy ART$_b$** are active, then F^{ab} becomes active only if **Fuzzy ART$_a$** predicts the same category as **Fuzzy ART$_b$** using the weights W_j^{ab}. The F^{ab} output vector X^{ab} follows these rules:

$X^{ab} =$	$Y^b \wedge W_J^{ab}$	**If the Jth F_2^a neuron is active and F_2^b is active**
	W_J^{ab}	**If the Jth F_2^a neuron is active and F_2^b is inactive**
	Y^b	**If F_2^a is inactive and F_2^b is active**
	0	**If F_2^a is inactive and F_2^b is inactive**

Table 6.1. Rules for the output vector X^{ab}

For instance, X^{ab} will be 0 if the prediction W_j^{ab} is disconfirmed by Y^b. Such a mismatch event triggers a **Fuzzy ART$_a$** search for a better category using the match tracking mechanism.

c) **The Match Tracking Mechanism**. At the start of each input presentation the **Fuzzy ART$_a$** vigilance parameter ρ_a equals a baseline vigilance ρ**bar$_a$**. The map field vigilance parameter is ρ_{ab} (generally set to 1.0). If the activation of **Fuzzy ART$_a$** and the activation of **Fuzzy ART$_b$** do not activate F^{ab} because of a mismatch in category, then ρ_a is increased until if is slightly larger than $|A \wedge w_J^a||A|^{-1}$, where **A** is the input to F_1^a, in second complement coding form. When this occurs, the **Fuzzy ART$_a$** search leads either to activation of another F_2^a neuron **J**; or, if not such neuron exists, to the incremental addition of a neuron to F_2^a to learn this "novelty" input.

d) **Learning in the Map Field**. As soon as **J** from $\mathbf{F_2^a}$ (**Fuzzy** $\mathbf{ART_a}$) learns to predict the **Fuzzy** $\mathbf{ART_b}$ supervised target **K**, then $\mathbf{W_{JK}^{ab}}$ is set to 1 in a permanent fashion.

6.3.3 Voting Schemes for Fuzzy ARTMAP

Carpenter et al. [21,22] have described the advantages of using several Fuzzy ARTMAP neural networks trained on data sets using different orderings. The final prediction for a given test set is the one made by the largest number of neural networks. This "ensemble" strategy is based on the fact that the sequence of examples typically leads to different structures of categories. The "ensemble" strategy cancels some of the errors.

6.4 Support Vector Machines

Support Vector Machines (SVMs) are a new generation of algorithms in machine learning. SVMs are based on recent advances in statistical learning theory. There are two interesting features of SVMs:

1. **Complex Sructures.** SVMs are able to generate structures that are complex enough to deal with practical applications. These structures contain classes of radial basis functions, neural networks, polynomials and splines-based functions as particular implementations.

2. **Mathematical Analysis.** SVMs are able to generate complex structures but they are simple enough to be examined mathematically. SVMs can be conceptualized as a linear method in a high-dimensional feature space nonlinearly associated to input space.

We will introduce several important concepts required to understand SVMs.

6.4.1 The Concept of the Maximum Margin Hyperplane

Advances in statistical learning theory (i.e., VC (Vapnik-Chervonenkis) theory) [89,90], explain that it is critical to constrain the class of functions that the learning machine can generate to one with a capacity that is appropriate for the available training data. Burges [8] states “There is a remarkable family of bounds governing the relation between the capacity of a learning machine and its performance. The theory grew out of considerations of under what circumstances, and how quickly, the mean of some empirical quantity converges uniformly, as the number of data points increases, to the true mean (that which would be calculated from an infinite amount of data).” Therefore, to design efficient learning algorithms, a class of functions whose capacity can be computed is essential. SVMs are based on the class of hyperplanes

$$(\mathbf{W}.\ \mathbf{X}) + \mathbf{b} = 0$$

where **W** are the free parameters (e.g., called weights in neural networks - $\mathbf{W} \in \mathbf{R}^N$ (**R** is the set of real numbers)), **X** is an **N**-dimensional input vector, and **b** is a numeric parameter ($\mathbf{b} \in \mathbf{R}$). This class of hyperplanes corresponds to decision functions of the type(in pattern recognition)

$$\mathbf{f(x) = sign\ ((W.X) + b).}$$

Therefore **f** is a function of $\mathbf{f{:}R^N \rightarrow \{\pm 1\}}$. The maximum margin hyperplane, defined as the one with the maximal margin of separation between the two classes (i.e., +1 and –1), has the lowest capacity. To illustrate a maximum margin hyperplane, Figure 6.7 depicts a two-class dataset whose classes are linearly separable (i.e., their convex hulls cannot overlap). The maximum margin hyperplane is the one that gives the greatest separation from both convex hulls. In addition, this optimal hyperplane is orthogonal to the shortest line connecting the hulls.

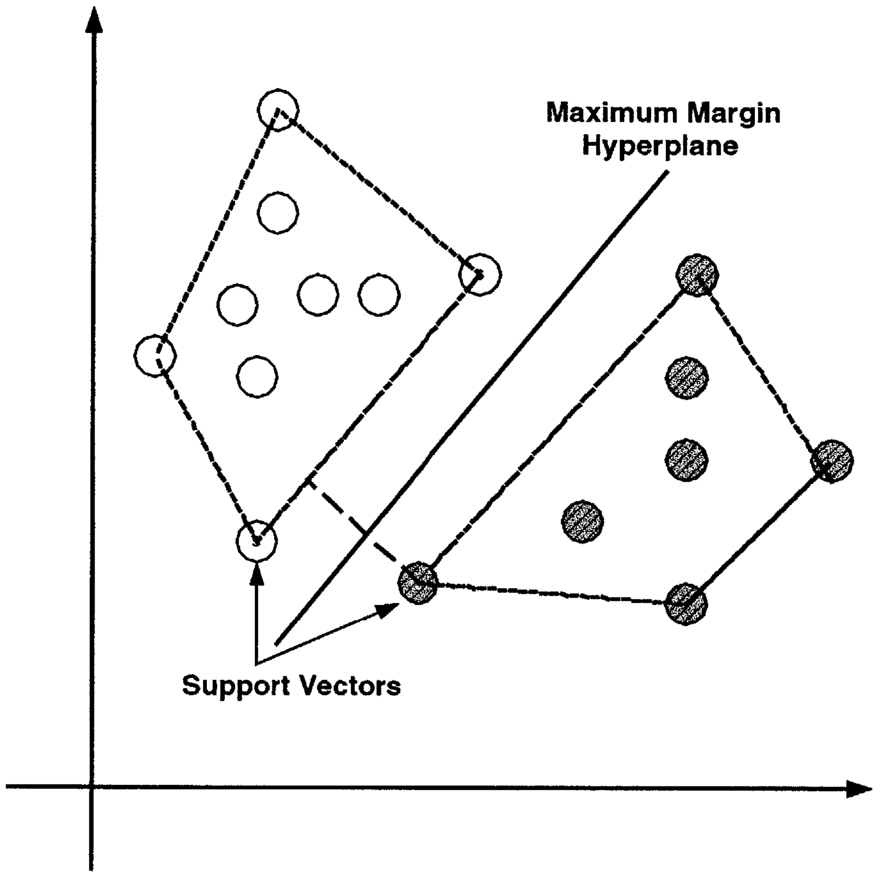


Figure 6.7. The maximum margin hyperplane is orthogonal to the shortest line connecting the convex hulls, intersecting it halfway.

The instances that are closest to the maximum margin hyperplane are called support vectors (SVs). Points that are not SVs have no influence [28,80,81,]. Therefore, this optimal hyperplane can be uniquely constructed by solving a constrained quadratic optimization problem whose solution **W** is represented by

$$\mathbf{W} = \Sigma_{i \in svs}\, y_i\, \alpha_i\, \mathbf{X}_i$$

where $\mathbf{X}_i$ is a support vector (selected from the training patterns), y_i is the respective classification $\mathbf{X}_i$ (i.e., ± 1), and α_i is a numeric parameter that has to be determined from the optimization. Then the final decision function

$$\mathbf{f(x, \alpha, b)} = \Sigma_{i \in svs}\, y_i\, \alpha_i\, (\mathbf{X}.\mathbf{X}_i) + \mathbf{b}$$

depends only on the dot products between patterns.

If the training set consists of **P** examples, then there are **P** free parameters in a SVM trained with **P** examples. These numeric parameters were called above α_i. Then, it is possible to pose the quadratic programming problem as [70,71]:

Minimize $(1/2) \Sigma_{i \in P, j=1} \alpha_i Q_{ij} \alpha_j - \Sigma_{i \in P} \alpha_i$;

Subject to $0 <= \alpha_i <= C$ **and** $\Sigma_{i \in P} \alpha_i y_i = 0$

where **Q** is a **PXP** matrix that depends on the training inputs X_i (for $i \in P$), the class y_i, and the functional form of the SVM. C is an upper bound to α_i. The objective function to be minimized depends on α_i quadratically, while α_i only appears linearly in the constraints. The quadratic problem in SVMs is to find a minimum of a convex objective function. The search for the minimum is constrained to lie within a hypercube and on a hyperplane. For most typical SVM functional forms, the matrix **Q** has special properties, so that the "objective is either positive definite or positive semidefinite." [70] Thus, there is either a unique minimum or a connected set of equivalent minima. The quadratic problem in SVMs has an optimality condition that describes these minima. These optimality conditions are described by the Karush-Kuhn-Tucker (KKT) conditions. These KKT conditions describe the set of α_i that are constrained minima.

There is one α_i for each training example. Each α_i determines how much each training example influences the SVM function. Therefore an $\alpha_i > 0$ identifies a training sample that is an SV. The constrained quadratic optimization problem can be solved using several approaches. It is possible to use the KKT conditions as a guide to obtain a solution. This fact results in several methodologies for finding the solution, such as interior point methods, combinations of gradient and conjugate-gradient ascent, and schemes based on Newtonian methods.

A very interesting fact to mention is that the matrix **Q** can be very large. It has a dimension that depends on the number of training examples. For instance, a training set of 20,000 examples will yield a **Q** matrix with 400 million elements! For these problems, researchers have been developing two different sets of methodologies [70]:

1. **Sophisticated data structures**. These methods perform samplings of the different rows and columns using heuristics based on the numeric value of α_i. These methods carry out dot products between rows or columns of **Q** and a vector, rather than performing computational expensive matrix-vector calculations. These methods use heuristics based on the value of α_i. For instance, they do not need to access the rows or columns of Q that correspond to α_i equals to 0 or the constrain C.

2. **Decomposition of the Quadratic Problem**. These methods are based on a decomposition of the quadratic problem into a series of smaller quadratic problems. Vapnik [88] recommends a method called "chunking." This "chunking" algorithm has been implemented in several SVM simulators available as freeware (e.g., RHUL release 1.0 for SVMs). The "chunking" algorithm exploits the fact that the value of the objective function is the same if the rows and columns of the matrix **Q** that correspond to zero α_i are removed. Therefore, the large problem can be broken down into a series of smaller quadratic problems, whose ultimate goal is to identify all of the nonzero α_i. At every step, this methodology solves a quadratic problem that consists of the following:

   Every nonzero α_i from the last step, and the α_i that correspond to the worst violations of the KKT conditions. [8,70]

   Nevertheless, the size of the quadratic problem tends to grow with time (with the accumulations of "unsolved" α_i). In the last step, the chunking approach has solved the overall quadratic problem by identifying the entire set of nonzero α_i.

6.4.2 Kernels

The basic idea of SVMs is to map the data into some other dot product space (called the feature space) F via a nonlinear map (see Figure 6.8):

$$\mathbf{\Phi: R^N \rightarrow F.}$$

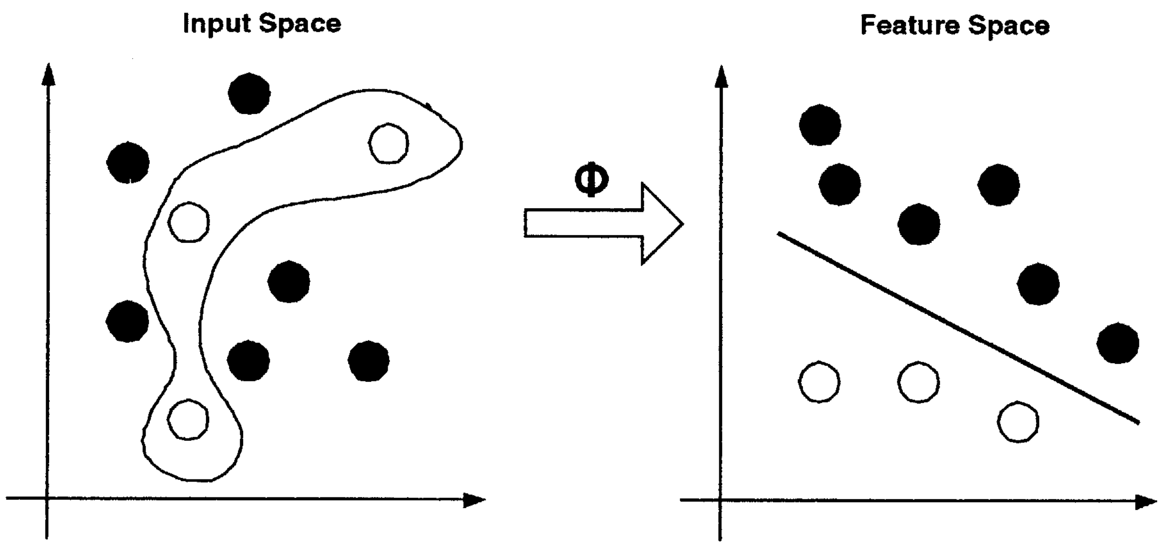


Figure 6.8. SVMs map the training data nonlinearly into a higher-dimensional feature space via Φ. Then a maximum margin hyperplane is constructed in this higher-dimensional feature space.

This only requires the evaluation of dot products:

$$\mathbf{k(X,X_i) = (\Phi(X) \,.\, \Phi(X_i)).}$$

This $\mathbf{k(X,X_i)}$ is called a kernel. For instance, the polynomial kernel

$$\mathbf{k(X,X_i) = (X.X_i)^d}$$

and defining $\mathbf{d = 2}$ and $\mathbf{\Phi(x)}$ for a 2-dimensional vector as $\mathbf{(x_1^2, 2^{0.5}\, x_1x_2, x_2^2)}$, then, the execution for vectors $\mathbf{X = [\, x_1\ x_2]^t}$ and $\mathbf{X_i = [x_{i1}\ x_{i2}]^t}$ is [81]:

$$k(X,X_i)^2 = \left(\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \cdot \begin{pmatrix} X_{i1} \\ X_{i2} \end{pmatrix}\right)^2 = \left(\begin{pmatrix} X_1^2 \\ 2^{.5}X_1X_2 \\ X_2^2 \end{pmatrix} \cdot \begin{pmatrix} X_{i1}^2 \\ 2^{.5}X_{i1}X_{i2} \\ X_{i2}^2 \end{pmatrix}\right)$$

A good way to select the degree of a polynomial kernel is to start with 1 and increment it until the estimated error ceases to improve. Other kernel functions can be used instead to implement different nonlinear mappings. Three that are often suggested are called the radial basis function kernel, the sigmoid kernel, and the splines kernel. The radial basis function kernel is represented by the following format

$$k(X,Xi) = \exp(-\|X - Xi\|^2 /(2\sigma^2)).$$

The sigmoid kernels (with gain κ and offset θ) have the following format

$$k(X,Xi) = \tanh(\kappa (X - Xi) + \theta).$$

The splines kernel, also utilized in this thesis, has the following equation [91]

$$k(X,Xi) = 1 + XX_i + XX_i \min(X,X_i) - ((X+X_i)/2)(\min(X,X_i))^2 + ((\min(X,X_i))^3)/3.$$

6.4.3 Architecture

SVMs (in pattern recognition) are nonlinear classifiers. A very important decision is the selection of the kernel. Which one produces the best results depends on the application. The user has to develop tests to decide among them. In addition, some of the kernels have parameters. This is another important decision that will affect performance. When the decision of the kernel is finalized (and $\Phi(x)$ can be substituted for each training example), the nonlinear decision function of the form

$$f(x, \alpha, b) = \text{sign} (\textstyle\sum_{i \in SVs} y_i \alpha_i (X.X_i) + b)$$

can be implemented. The parameters α are computed as the solution of the constrained quadratic programming problem. Figure 6.9 provides a good picture of the execution of an SVM for classification. The "trained" SVM has already the support vectors ($X_1, X_2, \ldots, X_n$). These vectors and the test vector (i.e., vector to be classified by the SVM) are substituted by the respective $\Phi(x)$. After that, the corresponding dot product can be performed. Then, weighting the dot product by the respective **yi αi** provides a nonlinear decision. This execution is very fast. Specialized hardware implementations using digital signal processors can be implemented for problems that require real-time performance.

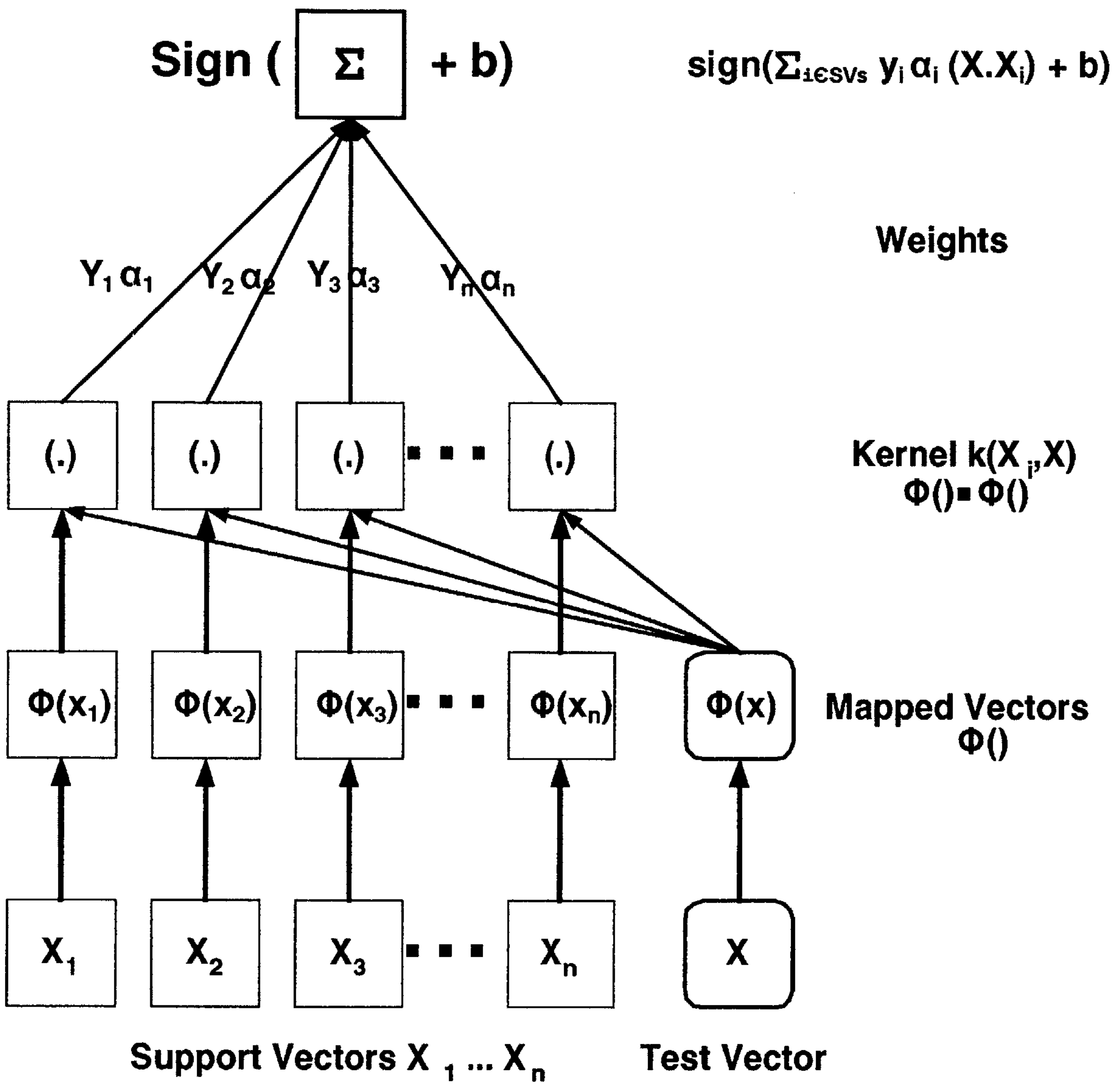


Figure 6.9. Architecture of a Support Vector Machine for classification

6.5 Software Implementations

Several software systems were developed to implement some of the algorithms and for the scaling and formatting of the training and testing datasets. These systems utilized a combination of Unix Scripts, Matlab Scripts, and C and C++ programs.

6.5.1 Backpropagation

There were several implementations for the different forms of the Backpropagation algorithm. Backpropagation with gradient-descent and other variations (momentum, penalizing weights) were implemented in a C program. Bayesian neural networks, Levenberg-Marquardt, and the conjugate-gradient algorithms were implemented using Matlab scripts. Several other C programs were built in order to support the data formatting, file handling, and crossvalidation.

6.5.1.1 Backpropagation With Gradient-Descent

The computer program developed in the C programming language has several features:

1. **Momentum Factor**. A momentum factor to accelerate training is implemented.
2. **Annealing schedules**. The annealing schedules of learning rates (η) and momentum factors (μ) were implemented. The user can define an annealing schedule to change learning rates and momentum factors during training as a function of the epochs or heuristics based on the changes in the mean squared error (MSE). Figure 6.10 shows an annealing schedule specified for a momentum factor as a function of the number of epochs (training iterations).

3. **Implementation Using Sockets**. The program uses sockets to implement a client. Then, the server program is able to handle several training sessions with different datasets at the same time. The server is able to stop the training session and make corrections to training parameters such as learning rates and momentum factors.

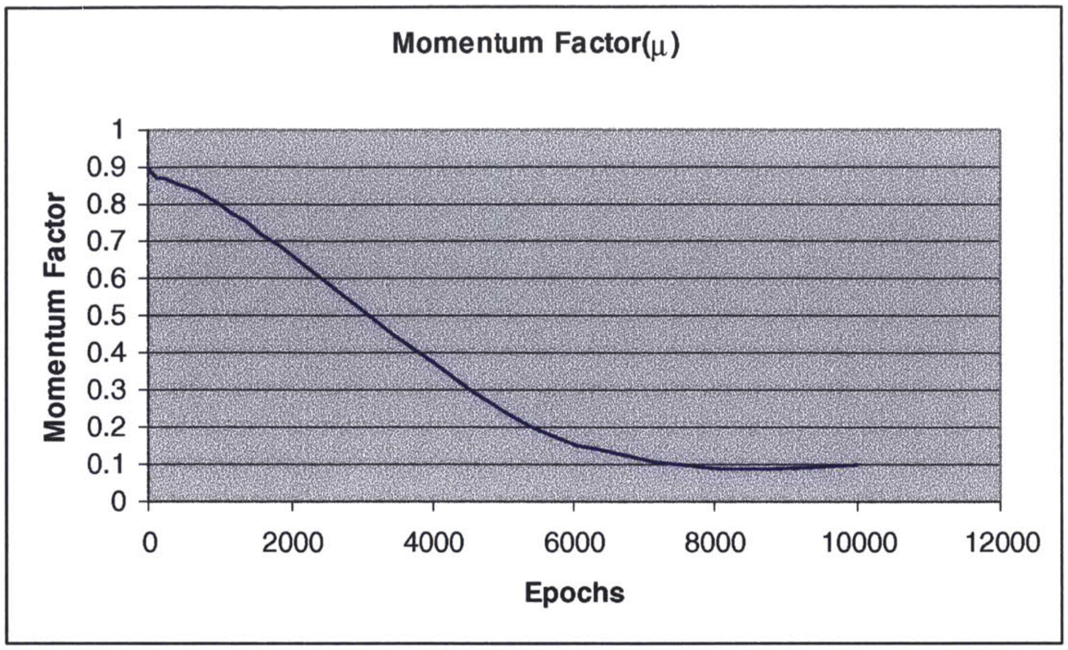


Figure 6.10. Example of an annealing schedule of the momentum factor for a learning session

4. **Testing of Files.** The user is able to test datasets at any point of the learning process.

5. **Network Architecture.** A file called "network.net" has the network architecture and the value of the current weights.

6. **Historical File.** A file contains the history of training (the learning rates, momentum factors, learning error, epoch number) and the major learning modifications. This file is important for gathering information in order to improve future sessions.

7. **Random Generators**. Random generators were adapted to the program. These random generators are very important to initialize the network and repeat experiments.

8. **Penalizing Weights (Regularization Factors).** The user can select penalizing equations for the learning session. Two equations are currently implemented:

   a) $\Sigma_k \mathbf{E}_k + \lambda \Sigma_i \left((w_i/w_o)^2 / (1 + (w_i/w_o)^2) \right)$

   b) $\Sigma_k \mathbf{E}_k + (1/2) \lambda \; \Sigma_i w_i^2$

   The regularization parameter λ is decided by the user. An annealing schedule can be designed for λ.

9. **Importance Factor**. An importance factor is used to denote the importance of a pattern.

10. **Simple User Interface**. A simple user interface is used. Text and files are used to communicate with the program. The program displays the Root Mean Squared (RMS) error and the current Epoch. For example, the training data is entered in ASCII format and uses the following format:

    # of input neurons
    # of hidden neurons
    # of output neurons
    # of training samples
    Momentum Characteristics
    Regularization Characteristics
    Stopping Decision: Minimum Error Desired Or Total Number of Epochs
    Training patterns each separated by one or more spaces, first for the input and then for the output, and then the importance factor

Other programs were constructed in C to scale and format the datasets and automate the process for the user as depicted in Figure 6.11.

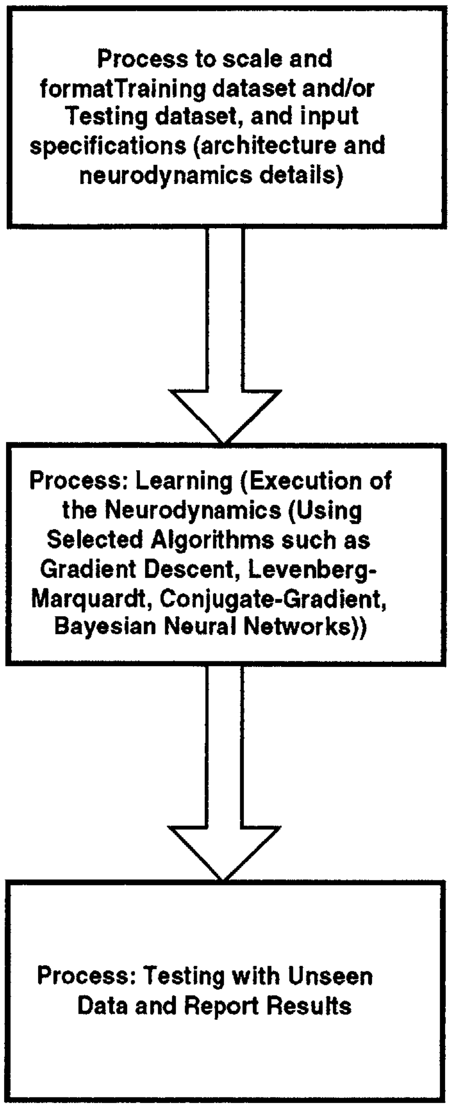

Process to scale and formatTraining dataset and/or Testing dataset, and input specifications (architecture and neurodynamics details)
Process: Learning (Execution of the Neurodynamics (Using Selected Algorithms such as Gradient Descent, Levenberg-Marquardt, Conjugate-Gradient, Bayesian Neural Networks))
Process: Testing with Unseen Data and Report Results

Figure 6.11. The implementation of Backpropagation

6.5.1.2 Bayesian Neural Networks

This thesis implements the Bayesian Neural Networks paradigm using Matlab scripts. Matlab and its Neural Network toolbox have the function “trainbr” [32] to build neural networks with automated regularization capabilities. A complete Matlab script was developed that executes the processes in Figure 6.11; however, the process of learning uses “trainbr.”

6.5.1.3 Levenberg-Marquardt

Matlab and its Neural Network toolbox has the function “trainlm” [32] to build Backpropagation neural networks using the Levenberg-Marquardt algorithm. The process of learning in Figure 6.11 is constructed using "trainlm." The Neural Network Toolbox has a very efficient implementation of this algorithm.

6.5.1.4 Conjugate-Gradient

Conjugate-Gradient is implemented in the function "traincgf" [32] in Matlab and its Neural Network toolbox to build Backpropagation neural networks. Similar to the Bayesian Neural Networks and Levenberg-Marquardt implementations, a Matlab script is built and the process of learning is composed using "traincgf."

6.5.1.5 Crossvalidation

A computer program written in C was developed to read the training file and create "randomly" 10 disjoint sets (i.e., using tenfold crossvalidation). Then, several combinations are created as training-testing pairs:

No.	Training File (Disjoint Sets)	Testing File (Disjoint Set)
1	1,2,3,4,5,6,7,8,9	10
2	10,1,2,3,4,5,6,7,8	9
3	9,10,1,2,3,4,5,6,7	8
4	8,9,10,1,2,3,4,5,6	7
5	7,8,9,10,1,2,3,4,5	6
6	6,7,8,9,10,1,2,3,4	5
7	5,6,7,8,9,10,1,2,3	4
8	4,5,6,7,8,9,10,1,2	3
9	3,4,5,6,7,8,9,10,1	2
10	2,3,4,5,6,7,8,9,10	1

Table 6.2. Development of training and testing files for crossvalidation

In addition, a Matlab script was utilized to implement a simple search routine to find an appropriate architecture using the estimation of GA from crossvalidation (see Figures 6.12 and 6.13). The search routine starts using several base architectures with specific number of hidden units. There are 8 base architectures with the following hidden units:

Base Architecture No.	Number of Hidden Units (Hidden Units of Respective Local Architectures)
1	0
2	4 (2, 3, 4)
3	8 (5, 6, 7, 8)
4	12 (9, 10, 11, 12)
5	16 (13, 14, 15, 16)
6	20 (17, 18, 19, 20)
7	24 (21, 22, 23, 24)
8	32 (25, 26, 27, 28, 29, 30, 31, 32)

Table 6.3. Base and local architectures

A base architecture is selected based on its respective estimated GA by using crossvalidation. The base architecture is tied to several local architectures. For instance, a base architecture of 4 hidden units has three local architectures: 2, 3, and itself. The base architecture with 8 hidden units has four local architectures: 5, 6, 7, and itself. The final architecture is selected based on its respective estimated GA by using crossvalidation. Then, this architecture is trained using the entire training dataset. The final test is performed using the testing dataset. The respective algorithm has to be selected previously (e.g., gradient-descent, conjugate-gradient, Levenberg-Marquardt).

.

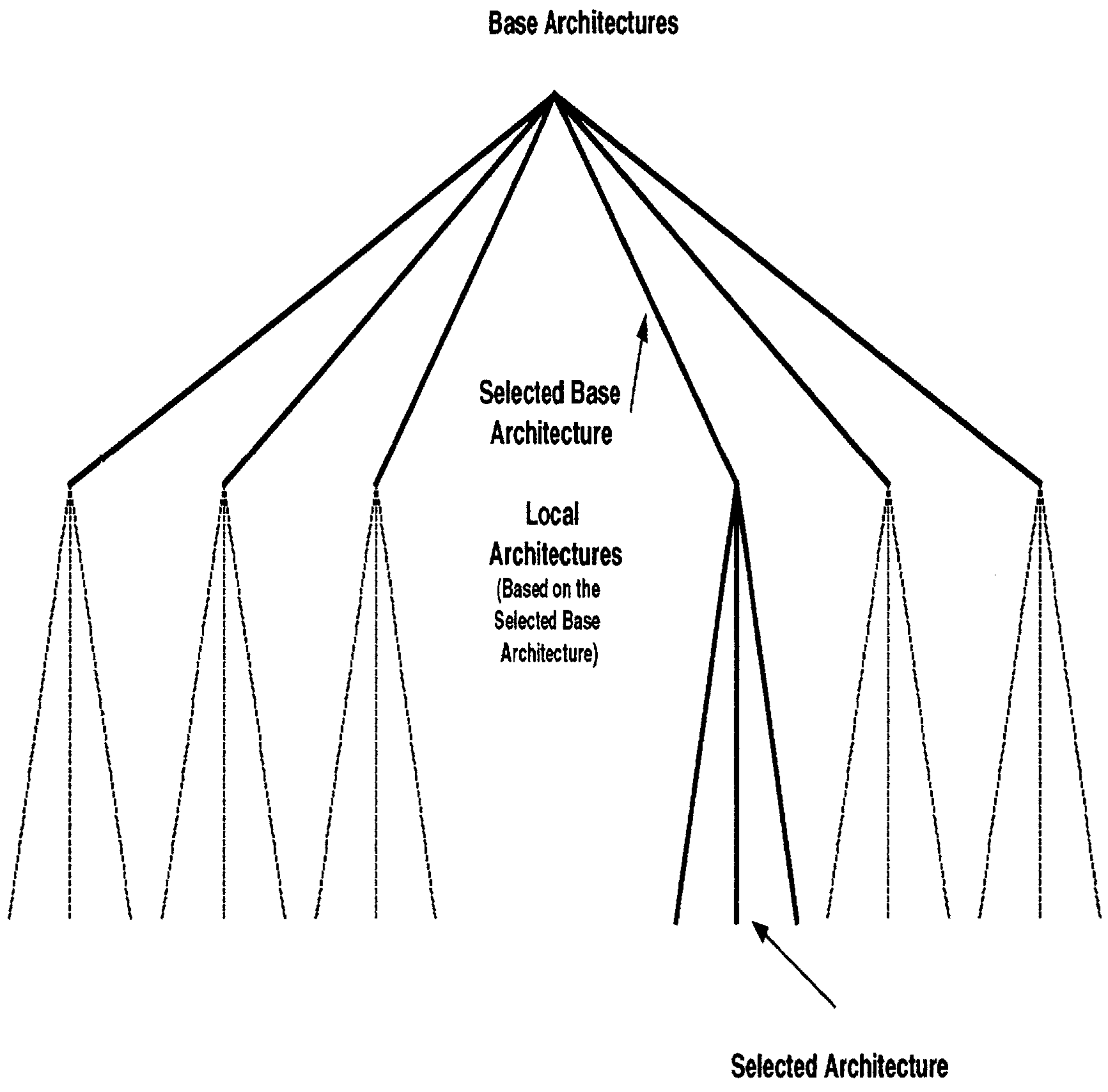

Base Architectures
Selected Base Architecture
Local Architectures (Based on the Selected Base Architecture)
Selected Architecture

Figure 6.12. Simple search routine to find an appropriate architecture.

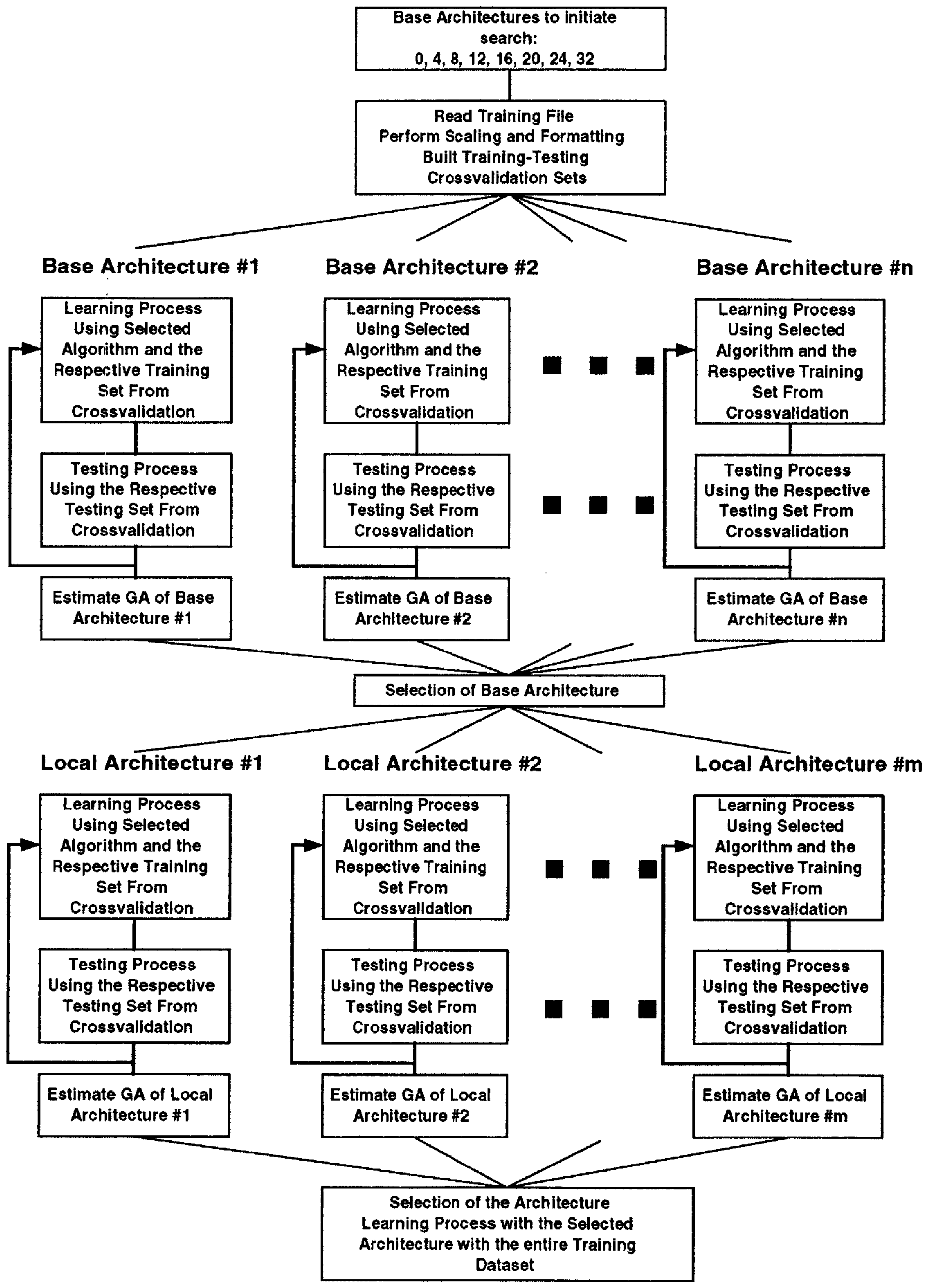

Base Architectures to initiate search:
0, 4, 8, 12, 16, 20, 24, 32
Read Training File
Perform Scaling and Formatting
Built Training-Testing
Crossvalidation Sets
Base Architecture #1
Base Architecture #2
Base Architecture #n
Learning Process Using Selected Algorithm and the Respective Training Set From Crossvalidation
Testing Process Using the Respective Testing Set From Crossvalidation
Estimate GA of Base Architecture #1
Estimate GA of Base Architecture #2
Estimate GA of Base Architecture #n
Selection of Base Architecture
Local Architecture #1
Local Architecture #2
Local Architecture #m
Estimate GA of Local Architecture #1
Estimate GA of Local Architecture #2
Estimate GA of Local Architecture #m
Selection of the Architecture
Learning Process with the Selected Architecture with the entire Training Dataset

Figure 6.13. Selection of an appropriate architecture using crossvalidation

6.5.2 Fuzzy ARTMAP

A software system was developed in the C Programming language to implement Fuzzy ARTMAP. Several other C programs were developed to format and scale the training and testing datasets. In addition, an innovative mechanism was developed in this thesis to decide ρ_a. The initial value of ρ_a decides the maximum size of the hyper-rectangles. This affects the final predictive performance of Fuzzy ARTMAP (See Figure 6.14).

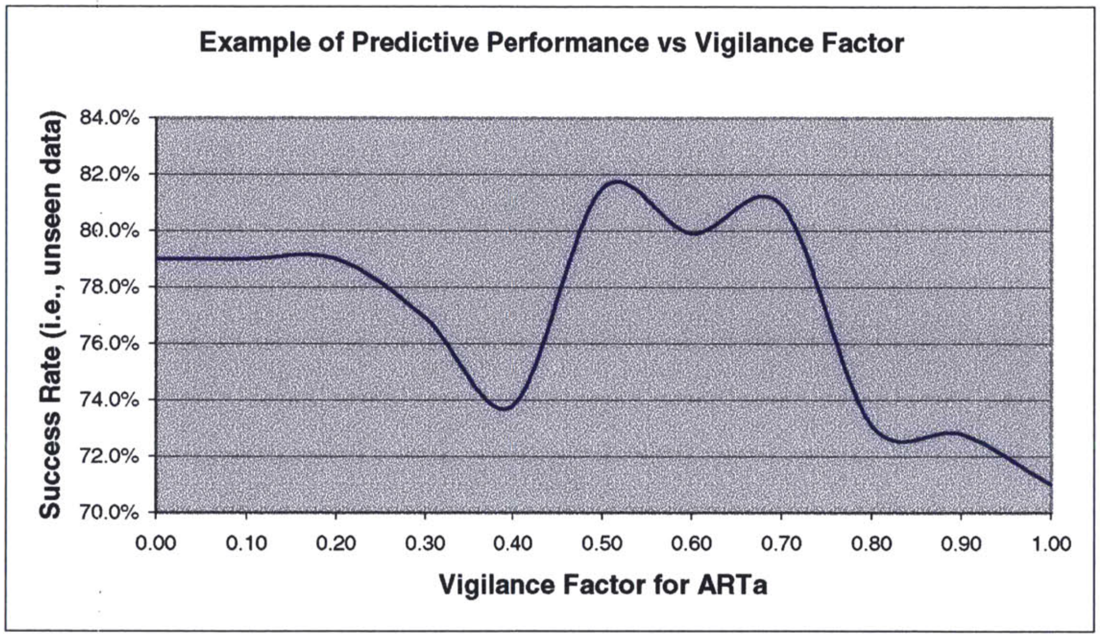
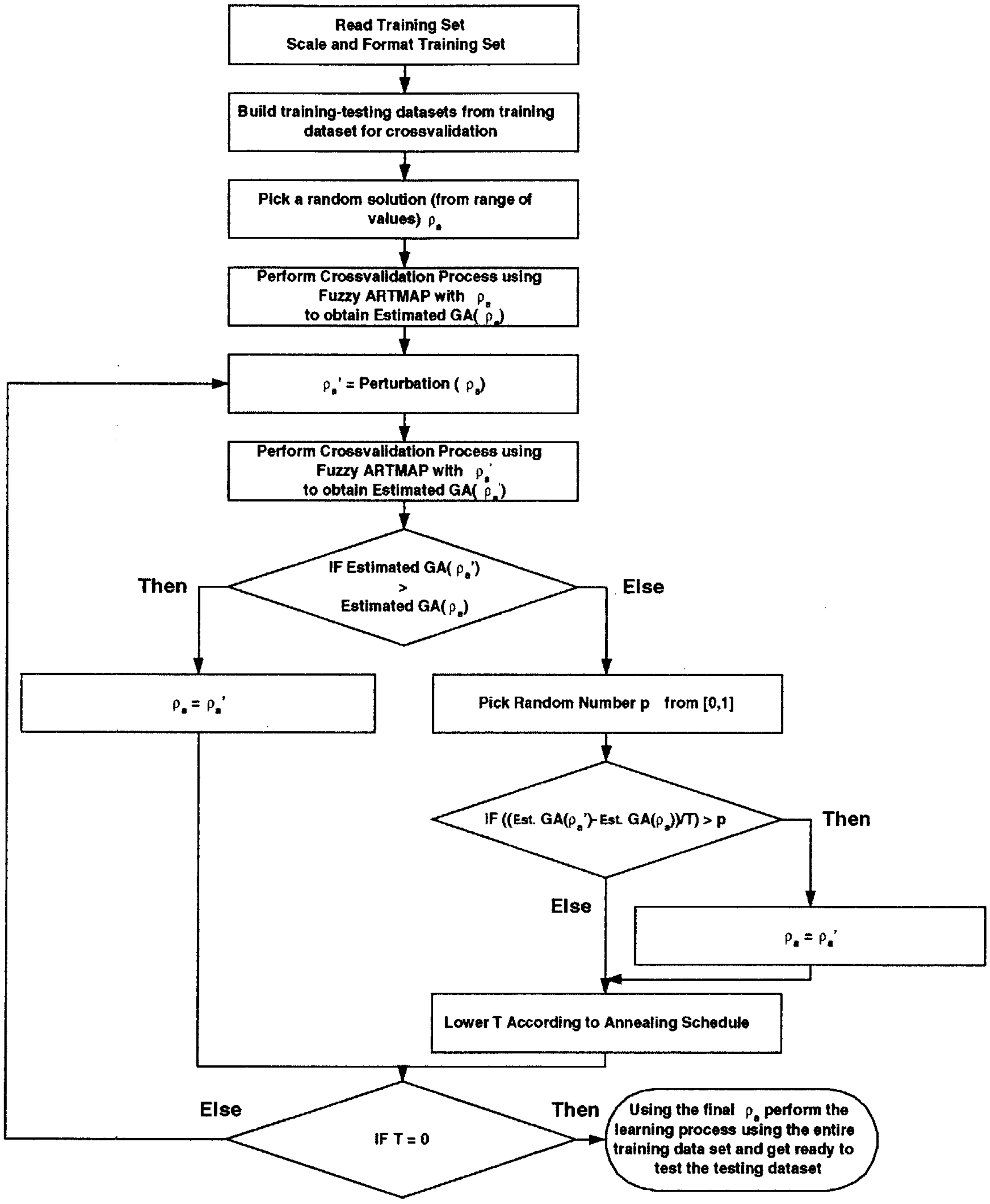


Figure 6.14. The predictive performance of a Fuzzy ARTMAP network is dependent on the initial ρ_a. This picture shows the success rate of Fuzzy ARTMAP trained with different ρ_a for a particular problem.

The innovative mechanism developed uses crossvalidation (using the training dataset). The estimated GA using crossvalidation allows the space to be searched for possible values of ρ_a ($\rho_a \in [0,1]$). A simple simulated-annealing engine [29,72] was developed as the search mechanism. Simulated-annealing was selected due to its stochastic nature, efficiency, and ability to work with multi-modal problems. In addition, the range of values for ρ_a facilitated its implementation.



Read Training Set
Scale and Format Training Set
Build training-testing datasets from training dataset for crossvalidation
Pick a random solution (from range of values) ρ_a
Perform Crossvalidation Process using Fuzzy ARTMAP with ρ_a to obtain Estimated GA(ρ_a)
ρ_a' = Perturbation (ρ_a)
Perform Crossvalidation Process using Fuzzy ARTMAP with ρ_a' to obtain Estimated GA(ρ_a')
IF Estimated GA(ρ_a') > Estimated GA(ρ_a)
Then
Else
ρ_a = ρ_a'
Pick Random Number p from [0,1]
IF ((Est. GA(ρ_a')-Est. GA(ρ_a))/T) > p
Then
Else
ρ_a = ρ_a'
Lower T According to Annealing Schedule
Else
IF T = 0
Then
Using the final ρ_a perform the learning process using the entire training data set and get ready to test the testing dataset

Figure 6.15. Process to find an appropriate ρ_a for Fuzzy ARTMAP using crossvalidation and simulated annealing.

Figure 6.15 shows the process to find an ρ_a. This process was implemented using several C programs. The perturbation of ρ_a in order to obtain ρ_a' is performed by a normal distribution and a routine that checks the value of ρ_a' (if the value of ρ_a' is not between the specified bounds, then, the perturbation will be repeated). A UNIX script was utilized to assemble the different processes.

6.5.3 Support Vector Machines

The RHUL release 1.0 for SVMs was utilized [78,79]. This software was developed by well recognized researchers in SVMs from the University of London, AT&T, and GMDH First [79]. RHUL release 1.0 uses for pattern recognition an interior-programming algorithm. The permission given by the authors of this software allows the usage of the programs but no modifications. Therefore, several C programs were developed to parser the outputs and write the inputs in the right formats. A UNIX script was able to assemble and automate the learning and testing process.

The kernels utilized in this thesis were:

1. Radial Basis Function: $\mathbf{exp(-\|X - Xi\|^2 \gamma)}$.

2. Linear Splines: $\mathbf{1 + XX_i + XX_i \min(X,X_i) - ((X+X_i)/2)(\min(X,X_i))^2 + ((\min(X,X_i))^3)/3}$.

In addition, an innovative mechanism was developed in this thesis to decide:

1. **Kernel.** The kernel to be used from the two choices selected: radial basis function (RBF) or the Linear Splines.

2. **γ for RBFs**. γ is related to the radius of the RBF and ranges from 1000.0 to 0.001. The predictive performance of a SVM with a RBF kernel depends on the value of γ (see Figure 6.16). The innovative mechanism developed uses crossvalidation. The estimated GA (i.e., fitness) using crossvalidation allows the space to be searched to decide a kernel and if the kernel selected is RBF, then, the value for γ is provided.

A simple genetic algorithm engine was developed as the search mechanism. Genetic algorithms were selected due to their stochastic nature, efficiency, ability to work with multi-modal problems, and the fact that problem could be encoded in a binary form [35,45].

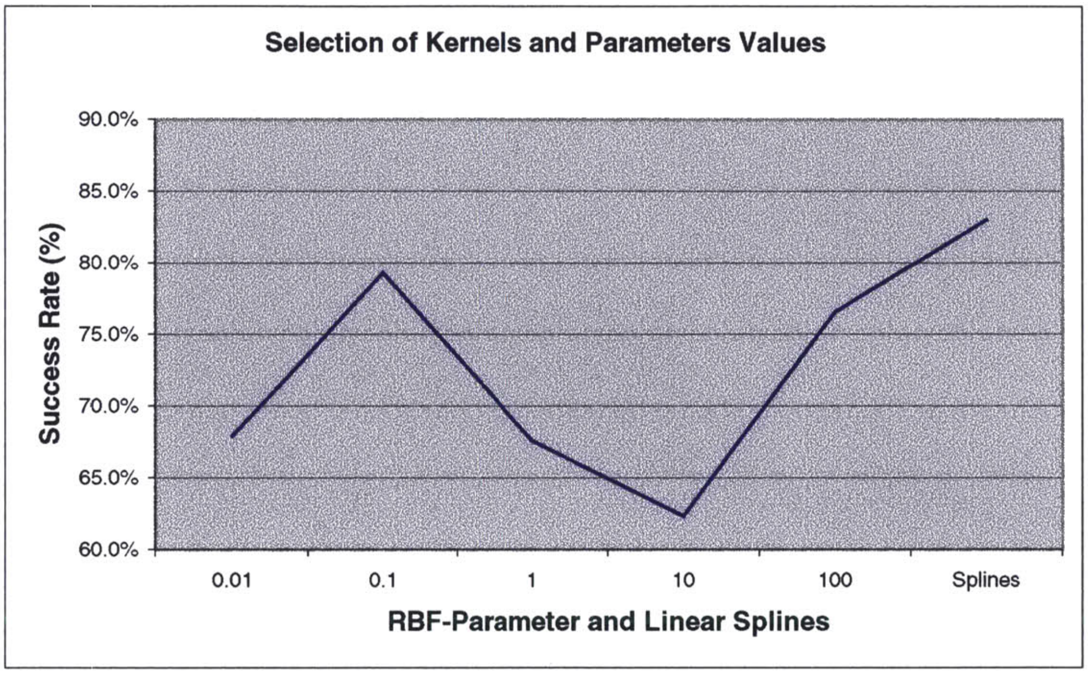


Figure 6.16. The predictive performance of an SVM is dependent on the kernel selected. This picture shows the success rate of an SVM trained with different kernels (Linear Splines and RBFs) and parameters γ (RBFs) for a particular problem.

γ is not uniform (i.e., a value of 1000.05 is very different from a value of 0.05). The chromosome to represent the problem is shown below in Figure 16.17.

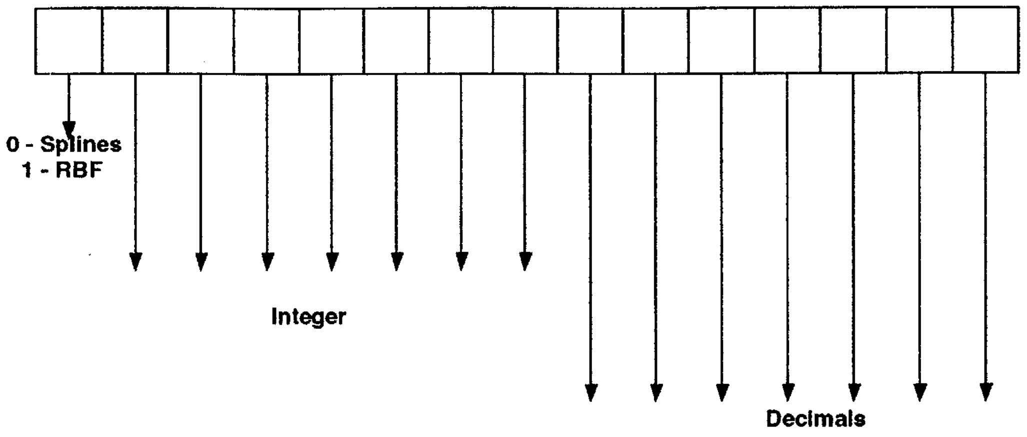


Figure 6.17. The chromosome has 15 genes. The first gene represents the kernel. The next 7 binary genes represent the integer part of γ . The final 7 binary genes represent the decimals of γ.

For instance, if the selected kernel is RBF and a γ of 100.04 is represented by the following individual 111000100011001 (See Figure 16.18).

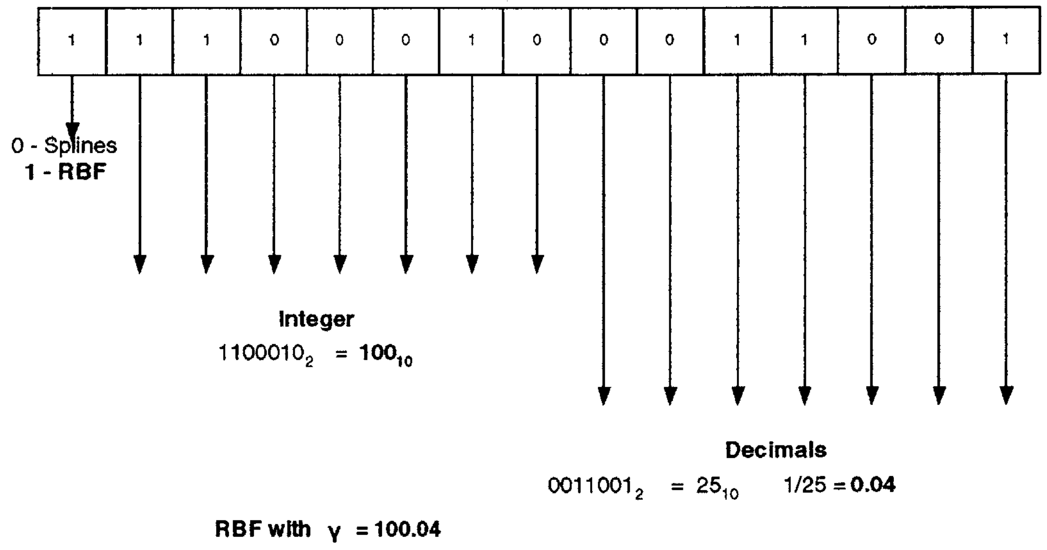


Figure 16.18. Representation of an SVM with kernel = RBF and γ = 100.04

The implementation of the genetic algorithm is very simple (see Figure 16.19). The genetic algorithm uses a population of SVM solutions and lets them reproduce according to a process of selection and recombination. Crossvalidation provides the fitness value for each solution. The solutions (i.e., SVMs) with the highest fitness are selected (i.e., elite). This sub-population p from the entire population n is the base from which to generate the new solutions of the population (all the old members that are not part of the elite die) via sexual reproduction between pairs of individuals from the elite (sometimes the other members can contribute with a very low probability to the new population), using the Crossover function. Mutation (i.e., perturbation) of these new SVMs is also applied at a low background rate. The new population is formed by the SVMs that are members of the elite and the new SVMs formed by the evolutionary process. The process continues until the stopping objective is achieved. This thesis implemented three stopping objectives: (1) the number of iterations (i.e., generations), (2) the fitness achieved, and (3) the calls to the SVM software to obtain with crossvalidation the fitness for a solution.

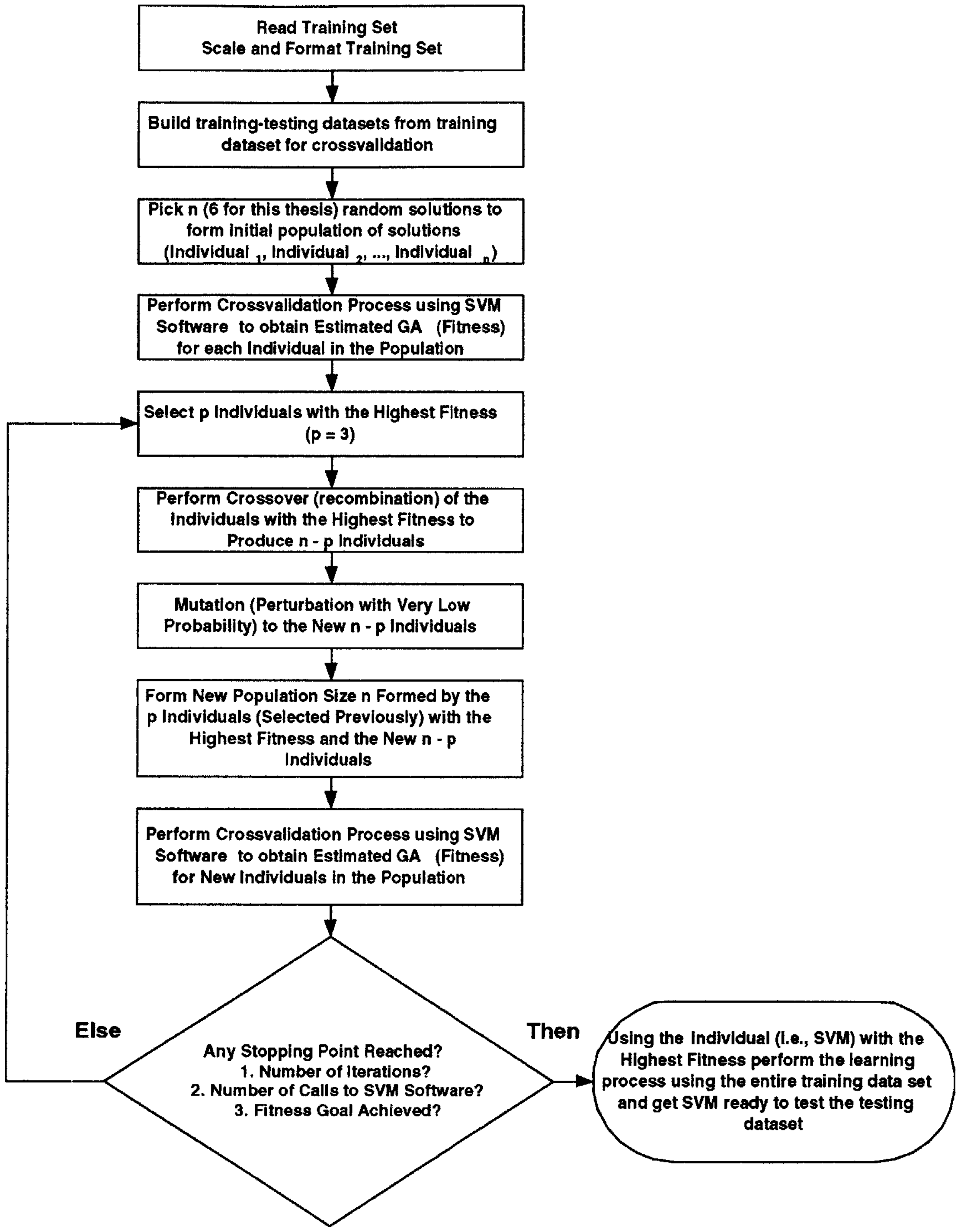

Read Training Set
Scale and Format Training Set
Build training-testing datasets from training dataset for crossvalidation
Pick n (6 for this thesis) random solutions to form initial population of solutions (Individual $_1$, Individual $_2$, ..., Individual $_n$)
Perform Crossvalidation Process using SVM Software to obtain Estimated GA (Fitness) for each Individual in the Population
Select p Individuals with the Highest Fitness (p = 3)
Perform Crossover (recombination) of the Individuals with the Highest Fitness to Produce n - p Individuals
Mutation (Perturbation with Very Low Probability) to the New n - p Individuals
Form New Population Size n Formed by the p Individuals (Selected Previously) with the Highest Fitness and the New n - p Individuals
Perform Crossvalidation Process using SVM Software to obtain Estimated GA (Fitness) for New Individuals in the Population
Any Stopping Point Reached?
1. Number of Iterations?
2. Number of Calls to SVM Software?
3. Fitness Goal Achieved?
Else
Then
Using the Individual (i.e., SVM) with the Highest Fitness perform the learning process using the entire training data set and get SVM ready to test the testing dataset

Figure 16.19 Selection of a SVM using a genetic algorithm with crossvalidation

6.5.4 Random Machines

Two random machines were developed. The first random machine ("PRAN" for pure random generator) decides the answer to a specific input pattern randomly.

A simple C program was built for this. On the other hand, the second random machine reads the training dataset to determine the frequencies of each category and based on this frequency creates a random generator. This random machine ("FRAN" for frequency random generator) was written in C.

7 Case Study and Results

This section deals with the development and testing of the different machine learning paradigms selected. A case study provides the testing environment. The rationale for choosing certain algorithms and refinements to the software implementations is presented. This discussion also includes an analysis of the results.

7.1 Description

The case study involved a data set that consisted of information on 125 subjects from a study conducted by Ryan [74]. A web site was used for this experiment. 648 images were shown sequentially to each subject (all of the images were saved using a JPG quality of 5). The response required from the individuals was their preference for each image (1: Yes, 0: No).

The images varied on seven attributes (features) with some specific levels (Please see Table 7.1):

- Density – Describes the number of circles in an image (three levels).
- Color Family – Describes the hue of the circles (three levels).
- Pointalization – Describes the size of the points that make the individual circles (three levels).
- Saturation – Describes the strength of the color within the circles (three levels).
- Brightness – Describes the amount of light in the circles themselves (four levels).
- Blur – Describes the crispness of the circles (two levels).
- Background – Describes the background color of the image (three levels).

However, the 624 images were generated using the levels depicted in Table 7.2. An illustration of these attributes can be seen in the images of Figures 7.1, 7.2, and 7.3.

There were 125 data sets (one from each subject). Each data set consisted of 648 image-response pairs. These 648 image-response pairs were divided randomly into two sets: each one with 324 pairs. One of these sets was going to be used to generate 5 training sets. The other set was going to be the testing set.

	Attribute	Level 1	Level 2	Level 3	Level 4
1	**Density**	X3	X2	X1	--
2	**Cold vs. Warm**	Cold: blue, green	purples	Warm: red, orange	--
3	**Pointalized**	5	15	50	--
4	**Saturation**	50	0	-50	--
5	**Light/Dark**	50	25	0	-25
6	**Motion blur**	0	10	--	--
7	**BKG**	Black	Gray	White	--

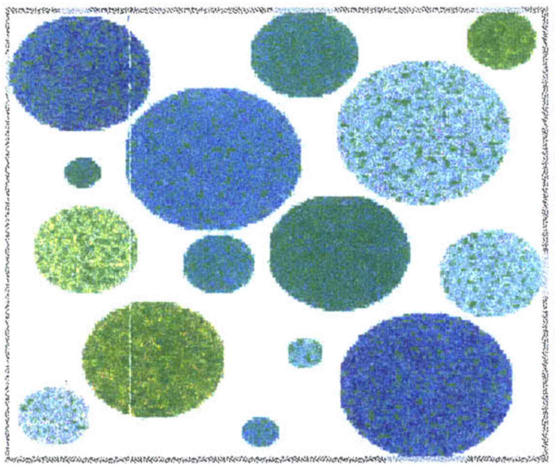
Total Images: 3 x 3 x 3 x 3 x 4 x 2 x 3 = 1944

Table 7.1. Features and their respective levels

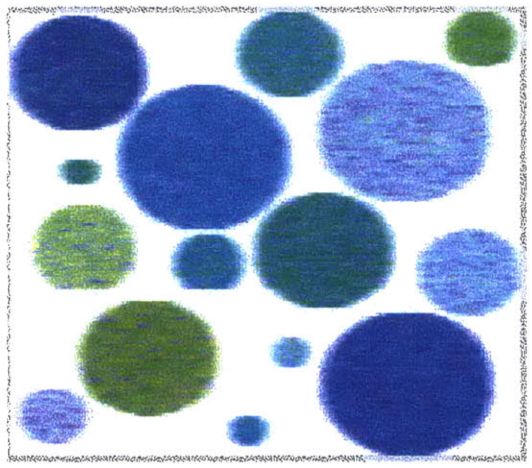
	Attribute	Level 1	Level 2	Level 3	Level 4
1	**Density**	X3	X2	X1	--
2	**Cold vs. Warm**	Cold: blue, green	purples	Warm: red, orange	--
3	**Pointalized**	5	15	50	--
4	**Saturation**	50	0	--	--
5	**Light/Dark**	50	--	--	-25
6	**Motion blur**	0	10	--	--
7	**BKG**	Black	Gray	White	--

Total Images: 3 x 3 x 3 x 2 x 2 x 2 x 3 = 648

Table 7.2. Features used to generate the 624 images

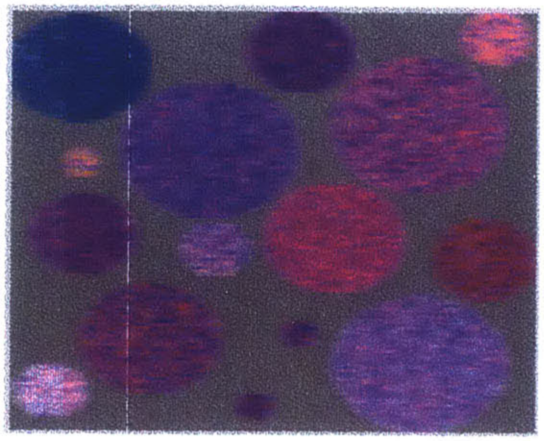


Density: Level 1 **Cold vs Warm**: Level 1
Pointalized: Level 1 **Saturation**: Level 1
Light/Dark: Level 1 **Motion blur**: Level 1
BKG: Level 3

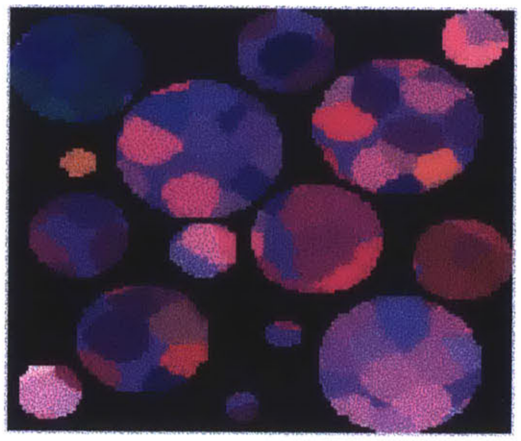


Density: Level 1 **Cold vs Warm**: Level 1
Pointalized: Level 1 **Saturation**: Level 1
Light/Dark: Level 2 **Motion blur**: Level 2
BKG: Level 3

Figure 7.1. Images with features 1111113 and 1111223 respectively.

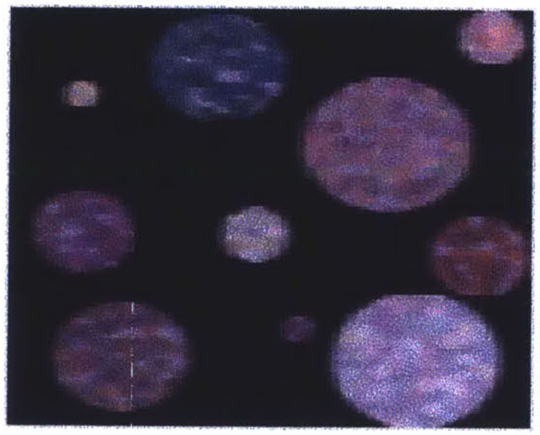


Density: Level 1 **Cold vs Warm**: Level 2
Pointalized: Level 1 **Saturation**: Level 1
Light/Dark: Level 3 **Motion blur**: Level 2
BKG: Level 2

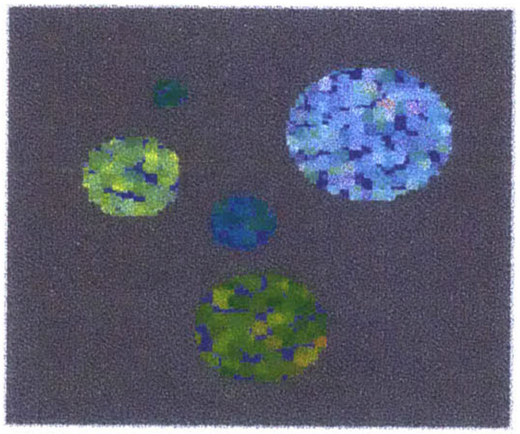


Density: Level 1 **Cold vs Warm**: Level 2
Pointalized: Level 3 **Saturation**: Level 1
Light/Dark: Level 3 **Motion blur**: Level 1
BKG: Level 1

Figure 7.2. Images with features 1211323 and 1231311 respectively.



Density: Level 2 **Cold vs Warm**: Level 2
Pointalized: Level 2 **Saturation**: Level 3
Light/Dark: Level 3 **Motion blur**: Level 2
BKG: Level 1



Density: Level 3 **Cold vs Warm**: Level 1
Pointalized: Level 2 **Saturation**: Level 1
Light/Dark: Level 2 **Motion blur**: Level 1
BKG: Level 2

Figure 7.3. Images with features 2223321 and 3121212 respectively.

The five training sets are selected randomly from the 324 image-response pairs. The first training set has 10 pairs. The second training set has 25 pairs. The third training set has 50 pairs. 100 pairs formed the fourth training set. And the last training set has 324 pairs. These different partitions provided a way to test the effects of the size of the training set on the GA of the learning machines developed.

7.2 Selection of Algorithms and Crossvalidation

This subsection reports on the initial performance and the adjustments required to the different implementations. It was decided to use the first 10 subjects to make these decisions.

7.2.1 Backpropagation Algorithms

The different Backpropagation algorithms were trained and compared to search for a suitable algorithm for the problem. The input scheme applied to the experiment data was to scale the inputs between –1 and 1. The output was represented as 1 for Yes, and 0 for No. The different algorithms had differences in the learning times. However, the execution of a trained network was on the order of microseconds.

7.2.1.1 Backpropagation with Gradient-Descent

Backpropagation with gradient-descent was tested with the first 10 subjects. It has a good computational efficiency with datasets formed by 10, 25, and 50 examples. However, it was obvious that the performance was dependent on the annealing schedules for the momentum factors and the learning rates. The algorithm has to be run several times (with different annealing schedules and initial random weights) in order to guarantee that at least one learning session was able to converge to a minimum error. There are architectures that cannot converge with specific datasets due to the lack of an appropriate number of free parameters (i.e., weights); however, Backpropagation with gradient-descent repeatedly failed to converge with architectures and datasets that algorithms such as Levenberg-Marquardt were able to do.

The training datasets of 324 examples created problems for gradient-descent. The learning time was excessive and each session approximately took, for each training dataset of crossvalidation (288 examples), more than 25 minutes. Therefore, the entire crossvalidation session (including all training datasets) for a single architecture took on average 250 minutes of computer time (Compaq Presario, Pentium III @ 600 MHz). These 250 minutes do not take into consideration the convergence problem. The percentage of convergence was very low for these datasets of 324 examples (even though other algorithms converged with the same architecture!). This percentage was approximately 20% (the percentage of convergence was higher with the datasets of 10 examples (approximately 90%), datasets of 25 examples (approximately 90%), datasets of 50 examples (approximately 80%), and the dataset of 100 examples (only a 50% rate of convergence)). Therefore, it was decided to abandon the efforts for Backpropagation with gradient-descent.

7.2.1.2 Backpropagation with Penalizing Weights

The first cost function described by

$$\Sigma_k \mathbf{E}_k + \lambda \Sigma_i ((w_i/w_o)^2 / (1 + (w_i/w_o)^2),$$

was investigated. We selected $\mathbf{w_o}$ to be 1.0 because the activations are of order unity (as provided by Weigend et al. [96,97]). It was very difficult to select the value of λ for each subject. Several annealing schedules and values were studied. For instance, one of the investigated alternatives was to assign λ values according to the following schedule:

1. Initially, $\lambda = 0$.
2. λ is incremented by a small quantity ($\delta\lambda << 1.0$, e.g., $\delta\lambda = 0.000005$) if $\mathbf{E_n}$ (where $\mathbf{E_n}$ is equal to $(1/N)\, \Sigma_k \mathrm{E}_k$ at epoch **n**) is less than the final desired error (**D**) and/or $\mathbf{E_n}$ is less than $\mathbf{E_{n-1}}$.
3. λ is decreased by $\delta\lambda$ if $\mathbf{E_n}$ is greater than or equal to **D** and $\mathbf{E_n}$ is greater than or equal to $\mathbf{E_{n-1}}$ and $\mathbf{E_n}$ is less than $\mathbf{A_n}$ (the exponentially weighted error at epoch **n**). $\mathbf{A_n}$ is calculated by using the following expression:

$$\mathbf{A}_n = \gamma\, \mathbf{A}_{n-1} + (1 - \gamma)\, \mathbf{E}_n$$

4. where γ is relatively close to 1.
5. λ is reduced by a small factor (e.g., $\lambda = 0.9\,\lambda$) if $\mathbf{E}_n$ is greater than or equal to **D** and $\mathbf{E}_n$ is greater than or equal to $\mathbf{E}_{n-1}$ and $\mathbf{E}_n$ is greater than or equal to $\mathbf{A}_n$.

Besides the problem of finding good values for λ and an appropriate schedule, convergence was even more difficult. The problem was that convergence to an appropriate error was usually not achieved (not only with the relatively large training databases of more than 100 examples but also with the training databases of 25 and 50 examples). Therefore, it was decided not to use this algorithm.

The second cost function investigated was

$$\Sigma_k \mathbf{E}_k + (1/2)\, \lambda\, \Sigma_i \mathbf{w}_i^2.$$

Different λs and annealing schedules were studied. $\lambda\mathbf{w}^2$ increased the performance of the Backpropagation neural networks. However, it was difficult to select appropriate λs for each subject. It usually took several trials. In addition, there were problems with convergence and learning times (especially with the relatively large training datasets). Another basic problem was to obtain an appropriate λ in an automated way tailored to a specific problem.

7.2.1.3 Bayesian Neural Networks

Bayesian neural networks were tested. These networks had good performance. However, their convergence rate for the relative large training datasets was less than desirable (less than 10%). Therefore, it was decided to investigate other alternatives.

7.2.1.4 Levenberg-Marquardt and Conjugate-Gradient

Backpropagation using Levenberg-Marquardt was investigated. Levenberg Marquardt provided a reliable and fast option. The algorithm was able to obtain a minimum error for the relatively large training datasets in approximately 25 seconds (in comparison with 25 minutes for gradient-descent). In addition, Levenberg-Marquardt was very reliable with an 80% convergence rate for the relatively large training datasets. The convergence rate was approximately greater than 90% for the other datasets.

Conjugate-gradient was also investigated. Conjugate-gradient obtained minimum errors for the relatively large training datasets in approximately 2 minutes. It was reliable with a 70% convergence rate for these datasets and 90% for the other training datasets that contained a smaller number of examples.

7.2.1.5 Selection and Modifications

It was clear that crossvalidation was the method of choice. The selected algorithm was Levenberg-Marquardt. However, the studies had to analyze 125 subjects and 5 different training datasets. The emphasis was on the automation of the process for training and testing. The automation of the learning process had to guarantee with a high reliability the delivery of a neural network that achieves a minimum error (convergence). There are architectures that cannot provide a minimum error because of the lack of enough free parameters (i.e., weights). However, the learning process has to re-initiate automatically the learning parameters and initialization of weights, and utilize other algorithms in order to assure that convergence, if possible, will be achieved. Therefore, the procedure depicted in Figure 7.4 was developed for the learning process. The algorithm used initially is Levenberg-Marquardt. If convergence is not achieved the initial weights are re-initialized up to a maximum of N1 times (N1 = 5 for this thesis). If these changes do not produce convergence, then changes to the learning parameters will be provided up to a maximum of N2 times (N2 = 3 for this thesis). The learning parameters such as number of epochs, minimum MSE error required, and gradient are modified. These changes to the learning parameters are combined with weight re-initializations. Finally, if convergence is not achieved, the current algorithm is changed to Conjugate-gradient, and the process described before is initiated again. If the network architecture cannot converge, and the neural network was for the crossvalidation process, the neural network with the minimum error is delivered to the process and the user is notified about this. However, if the neural network is for the final training process, the network architecture is labeled as “NC” (Non-convergence) and the user is notified of this.

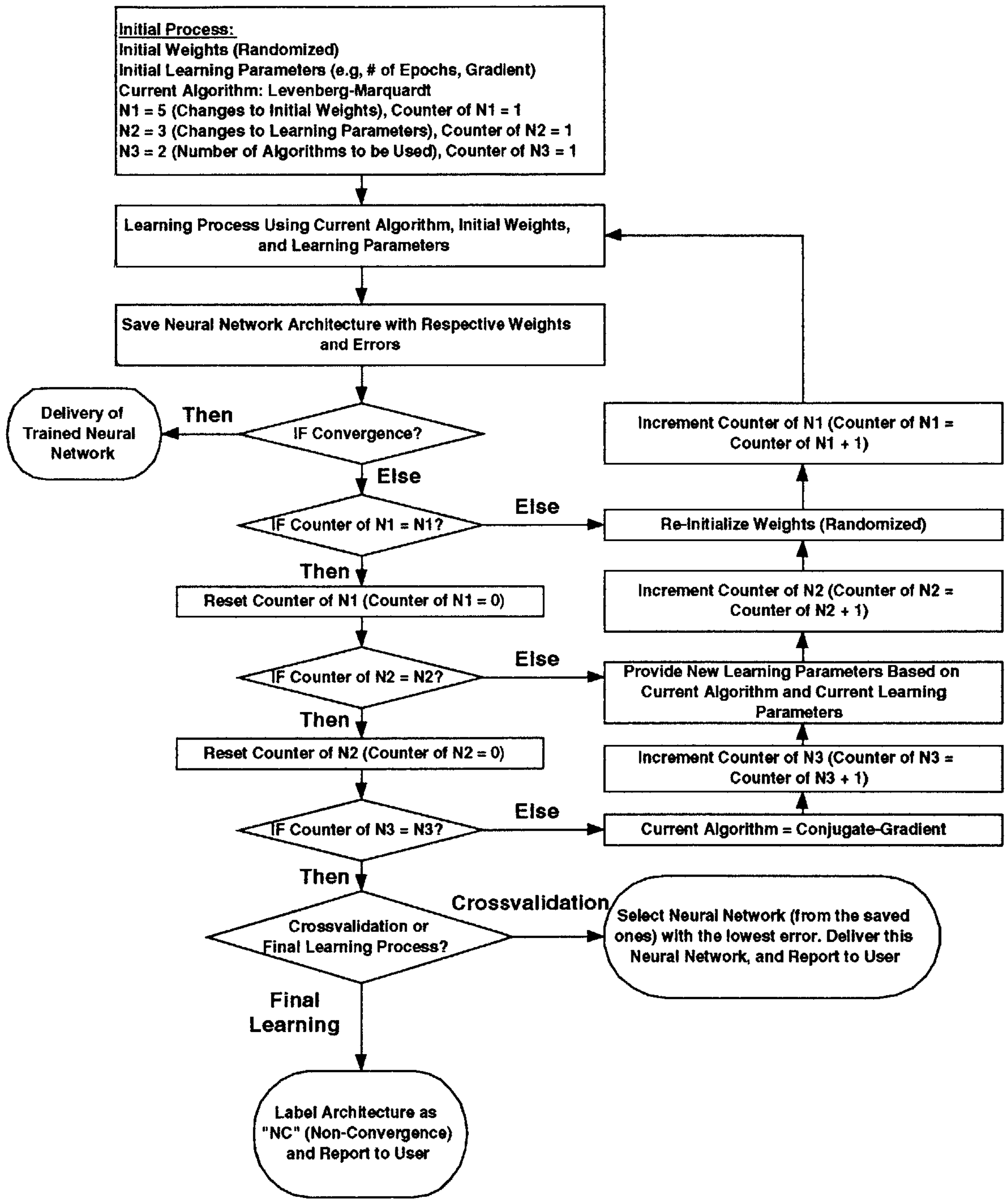

Initial Process:
Initial Weights (Randomized)
Initial Learning Parameters (e.g, # of Epochs, Gradient)
Current Algorithm: Levenberg-Marquardt
N1 = 5 (Changes to Initial Weights), Counter of N1 = 1
N2 = 3 (Changes to Learning Parameters), Counter of N2 = 1
N3 = 2 (Number of Algorithms to be Used), Counter of N3 = 1
Learning Process Using Current Algorithm, Initial Weights, and Learning Parameters
Save Neural Network Architecture with Respective Weights and Errors
IF Convergence?
Then
Delivery of Trained Neural Network
Else
IF Counter of N1 = N1?
Else
Re-Initialize Weights (Randomized)
Increment Counter of N1 (Counter of N1 = Counter of N1 + 1)
Then
Reset Counter of N1 (Counter of N1 = 0)
IF Counter of N2 = N2?
Else
Provide New Learning Parameters Based on Current Algorithm and Current Learning Parameters
Increment Counter of N2 (Counter of N2 = Counter of N2 + 1)
Then
Reset Counter of N2 (Counter of N2 = 0)
IF Counter of N3 = N3?
Else
Current Algorithm = Conjugate-Gradient
Increment Counter of N3 (Counter of N3 = Counter of N3 + 1)
Then
Crossvalidation or Final Learning Process?
Crossvalidation
Select Neural Network (from the saved ones) with the lowest error. Deliver this Neural Network, and Report to User
Final Learning
Label Architecture as "NC" (Non-Convergence) and Report to User

Figure 7.4. Learning process using Backpropagation and selected algorithms

7.2.2. Backpropagation and Crossvalidation

V-fold (V = 5, 10) Crossvalidation has been recognized recently as the standard way of measuring the error rate of a learning scheme on a particular dataset [47,100]. There are several studies related to the utilization of V-fold crossvalidation to select appropriate neural network architectures for Backpropagation [62]. However, it was decided to run and compare the crossvalidation scheme against the strategy of using a fixed architecture. Figure 7.5 and Tables 7.3 and 7.4 show the predictive performance (testing datasets) of several Backpropagation neural networks using different numbers of hidden neurons (0, 4, 8, 12, 20, 24, 28, and 49) and the selection of the crossvalidation process and its number of hidden neurons. The results indicate that crossvalidation is a valid process with which to select architectures. Nevertheless, it is not a silver bullet!

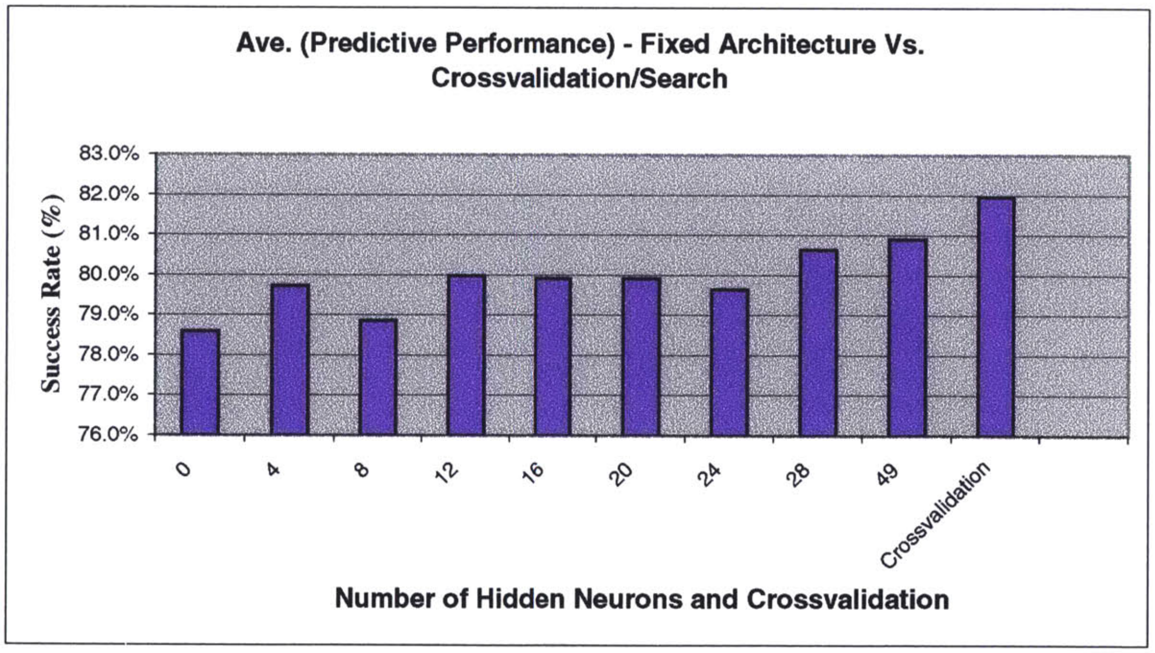


Figure 7.5. Performance of crossvalidation vs. a fixed architecture

		Performance (Succes Rate%)										
Subject	Size of	Hidden Layer Size (neurons)									Crossvalidation	
	Training File	0	4	8	12	16	20	24	28	49		Size
1	10	98.5%	96.0%	81.8%	94.1%	90.4%	87.3%	95.7%	98.8%	97.2%	98.8%	2
	25	98.8%	98.8%	98.8%	97.5%	95.1%	95.4%	98.8%	98.8%	98.8%	98.8%	2
	50	97.2%	98.1%	97.5%	94.8%	97.2%	97.8%	92.0%	96.6%	99.1%	98.8%	25
	100	NC	93.2%	94.1%	95.1%	94.4%	91.0%	96.3%	96.3%	96.9%	96.9%	23
	324	NC	90.4%	95.1%	94.4%	96.3%	95.1%	96.3%	97.2%	96.9%	97.8%	17
2	10	54.6%	76.2%	78.1%	77.2%	76.2%	74.4%	72.5%	78.1%	71.5%	78.1%	8
	25	73.8%	77.8%	76.2%	76.5%	76.2%	76.2%	74.1%	77.2%	76.5%	76.2%	19
	50	NC	76.5%	74.4%	72.5%	76.2%	73.1%	75.6%	72.8%	72.5%	76.5%	19
	100	NC	78.1%	78.1%	74.4%	74.1%	77.5%	79.6%	79.9%	77.2%	75.3%	23
	324	NC	NC	84.0%	78.7%	82.7%	80.2%	81.8%	77.8%	81.8%	84.0%	8
3	10	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	0
	25	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	0
	50	85.5%	84.0%	87.0%	91.4%	84.0%	88.9%	89.5%	89.5%	87.8%	86.4%	2
	100	NC	87.0%	86.1%	84.6%	87.7%	88.6%	86.4%	90.1%	88.0%	89.2%	2
	324	NC	NC	88.0%	85.2%	87.3%	89.2%	89.2%	89.2%	86.1%	92.0%	6
4	10	59.3%	69.1%	67.9%	59.3%	72.5%	63.0%	69.1%	62.0%	76.5%	67.6%	2
	25	NC	68.8%	69.4%	74.7%	70.4%	69.4%	70.4%	72.8%	70.4%	71.6%	2
	50	NC	70.4%	69.4%	73.8%	70.4%	74.7%	72.2%	73.8%	70.1%	74.7%	20
	100	NC	NC	66.0%	69.8%	65.1%	69.1%	71.0%	72.5%	70.7%	73.1%	25
	324	NC	NC	NC	NC	68.2%	69.8%	68.5%	68.5%	72.5%	70.7%	23
5	10	64.5%	70.7%	70.7%	70.7%	70.7%	70.7%	70.7%	70.7%	70.7%	70.7%	2
	25	70.7%	76.2%	66.0%	67.6%	72.5%	75.9%	72.5%	80.9%	72.5%	75.3%	14
	50	NC	79.0%	74.7%	81.5%	78.4%	80.6%	75.9%	81.2%	75.9%	80.6%	18
	100	NC	73.8%	77.8%	73.5%	80.2%	80.6%	80.9%	79.3%	80.2%	80.9%	19
	324	NC	NC	NC	74.7%	77.2%	82.1%	81.5%	79.6%	79.3%	82.1%	23
6	10	47.8%	76.5%	74.4%	68.5%	74.4%	67.3%	78.1%	70.4%	79.4%	64.8%	7
	25	76.5%	76.9%	73.8%	77.5%	79.0%	77.5%	76.2%	83.3%	70.6%	83.3%	25
	50	NC	67.6%	72.8%	76.5%	70.4%	75.9%	67.9%	76.5%	79.9%	78.7%	25
	100	NC	73.5%	76.9%	82.1%	86.1%	76.5%	79.0%	83.6%	79.0%	88.0%	25
	324	NC	NC	NC	NC	79.9%	79.3%	80.2%	81.5%	85.5%	88.9%	25
7	10	66.0%	59.6%	69.8%	65.4%	67.0%	72.2%	59.6%	62.7%	59.6%	67.0%	2
	25	NC	52.8%	58.3%	61.7%	60.8%	59.6%	60.8%	62.7%	73.5%	59.3%	14
	50	NC	69.8%	71.6%	72.2%	73.1%	65.7%	71.0%	72.5%	71.9%	73.1%	15
	100	NC	NC	61.1%	71.0%	70.1%	70.7%	67.6%	69.1%	69.4%	73.1%	6
	324	NC	NC	NC	NC	67.3%	65.7%	61.7%	69.1%	67.9%	75.9%	14
8	10	90.1%	91.4%	77.2%	88.6%	88.0%	88.6%	75.9%	83.6%	91.4%	91.4%	2
	25	84.3%	89.5%	81.2%	81.5%	82.4%	87.7%	84.9%	83.0%	84.0%	89.5%	4
	50	80.2%	87.7%	87.7%	90.1%	89.2%	86.7%	87.7%	87.0%	87.0%	89.2%	18
	100	NC	89.8%	88.6%	89.2%	89.5%	87.7%	89.8%	89.5%	91.4%	88.3%	19
	324	NC	NC	84.3%	84.0%	81.5%	88.3%	85.8%	85.8%	86.7%	84.9%	17
9	10	76.2%	80.2%	81.8%	89.5%	85.2%	93.2%	86.7%	82.7%	89.8%	82.4%	10
	25	88.6%	90.4%	89.5%	94.1%	92.6%	90.4%	90.4%	94.4%	94.4%	94.1%	12
	50	90.7%	91.4%	89.8%	92.3%	95.7%	94.1%	94.4%	94.8%	94.1%	95.7%	16
	100	NC	86.7%	88.6%	90.1%	89.2%	90.7%	88.9%	90.4%	85.8%	90.1%	24
	324	NC	NC	90.1%	91.4%	92.3%	93.2%	88.9%	90.1%	88.9%	92.3%	17
10	10	65.7%	67.6%	61.1%	66.7%	67.0%	66.4%	67.9%	67.9%	71.3%	67.6%	4
	25	NC	72.5%	71.3%	67.3%	74.4%	70.4%	70.7%	70.4%	70.4%	74.5%	15
	50	NC	67.3%	68.2%	64.8%	68.5%	68.2%	73.5%	67.0%	70.1%	73.5%	24
	100	NC	72.8%	67.9%	71.0%	67.6%	68.2%	72.8%	67.9%	71.6%	72.8%	24
	324	NC	NC	NC	67.9%	69.1%	67.9%	72.2%	70.7%	70.4%	72.2%	24

Table 7.3. Success rate of different architectures and the crossvalidation process for the first 10 subjects. NC indicates "Non-Convergence." Size indicates the number of hidden neurons for those architectures selected by crossvalidation.

	Performance (Succes Rate%)									
	Hidden Layer Size (neurons)									Crossvalidation
	0	4	8	12	16	20	24	28	49	
Ave. (Predictive Performance)	78.6%	79.7%	78.9%	80.0%	79.9%	79.9%	79.6%	80.6%	80.9%	82.0%
Std. Dev.	14.9%	11.0%	10.4%	10.6%	9.9%	10.3%	10.2%	10.4%	10.0%	10.2%
95% Confidence Upper	80.0%	81.1%	80.2%	81.3%	81.3%	81.3%	81.0%	82.0%	82.2%	83.2%
95% Confidence Lower	77.1%	78.3%	77.4%	78.6%	78.5%	78.5%	78.2%	79.2%	79.5%	80.6%
Frequency of best answer#	3	9	6	7	5	10	8	11	15	31
Frequency of best answer%	6.0%	18.0%	12.0%	14.0%	10.0%	20.0%	16.0%	22.0%	30.0%	62.0%
95% Confidence Upper	16.2%	30.8%	23.8%	26.2%	21.4%	33.0%	28.5%	35.2%	43.8%	74.1%
95% Confidence Lower	2.1%	9.8%	5.6%	7.0%	4.3%	11.2%	8.3%	12.8%	19.1%	48.2%

Table 7.4. 95% confidence intervals for the success rate of the different architectures and the crossvalidation process for the first 10 subjects

7.2.3 Fuzzy ARTMAP Implementation

The implementation of Fuzzy ARTMAP was tested and it worked very well for all problems. There are several advantages to this neural network paradigm such as self-organization and convergence. This paradigm decides its own architecture (hidden units) and always converges. The computational speed of Fuzzy ARTMAP for the learning process for the relatively large training database is approximately 3 seconds. The execution of a trained network is on the order of microseconds. The second complement normalization and the scaling of the inputs from 0 to 1 were accomplished. For instance, the input for the first diagram of Figure 7.1 and features 1111113 is 0000001 and including the second complement normalization the entire input is 00000011111110 (therefore, 14 input neurons were needed to represent it). The output for yes was represented by 10 and the output for no was represented by 01. Therefore, including the corresponding second complement normalization, the output for yes is 1001 and for no is 0110 (therefore, 4 output neurons are required to represent it).

7.2.4 Fuzzy ARTMAP and Crossvalidation

This thesis developed a mechanism to provide ρ_a. This mechanism uses simulated annealing and crossvalidation. It was decided to test this mechanism against the utilization of a fixed ρ_a (a common method used by researchers of Fuzzy ARTMAP). Figure 7.6 andTables 7.5 and 7.6 show the predictive performance (testing datasets) of several Fuzzy ARTMAP neural networks using different ρ_as and the selection of the crossvalidation process and its respective ρ_as. The results indicate that the developed mechanism using crossvalidation and simulated annealing is a valid process with which to select ρ_a for Fuzzy ARTMAP.

Subject	Size of Training File	Performance (Success Rate %) Vigilance Factor (ρa) 0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	Crossvalidation	ρa
1	10	98.8%	98.8%	98.8%	97.5%	89.5%	90.4%	89.2%	85.2%	87.3%	82.4%	98.8%	0.00
	25	98.8%	98.8%	98.8%	94.4%	93.8%	94.1%	96.3%	94.1%	91.7%	89.2%	98.8%	0.00
	50	98.8%	98.8%	98.8%	97.5%	97.8%	96.0%	96.0%	95.1%	92.6%	93.2%	98.8%	0.00
	100	98.5%	98.5%	98.5%	96.9%	97.5%	96.9%	95.1%	96.9%	94.8%	94.1%	98.5%	0.00
	324	93.2%	93.2%	93.2%	94.4%	96.0%	95.7%	94.4%	95.7%	92.3%	90.4%	94.4%	0.30
2	10	68.5%	68.5%	68.5%	68.5%	68.5%	68.5%	68.5%	66.0%	65.1%	65.1%	77.8%	0.65
	25	73.1%	73.1%	73.1%	73.1%	71.6%	69.8%	73.5%	72.5%	71.9%	71.0%	75.9%	0.75
	50	71.3%	71.3%	71.3%	67.0%	72.5%	72.2%	69.8%	71.0%	72.5%	72.2%	76.2%	0.85
	100	67.0%	67.0%	67.0%	67.0%	71.3%	72.2%	69.1%	69.4%	69.1%	69.8%	72.2%	0.60
	324	71.3%	71.3%	71.3%	68.2%	71.9%	71.9%	71.6%	66.7%	67.3%	66.0%	81.2%	0.25
3	10	90.7%	90.7%	90.7%	90.7%	90.4%	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	0.00
	25	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	0.00
	50	90.1%	90.1%	90.1%	85.5%	91.0%	87.7%	87.7%	88.6%	86.4%	87.7%	90.1%	0.00
	100	80.6%	80.6%	80.6%	76.9%	85.5%	86.1%	85.5%	86.7%	85.8%	85.8%	86.7%	0.70
	324	84.9%	84.9%	84.9%	80.2%	84.6%	86.7%	89.2%	86.1%	85.5%	82.1%	87.0%	0.55
4	10	68.5%	68.5%	68.5%	68.5%	68.5%	68.5%	68.5%	66.0%	65.1%	65.1%	66.0%	0.65
	25	73.1%	73.1%	73.1%	73.1%	71.6%	69.8%	73.5%	72.5%	71.9%	71.0%	72.2%	0.75
	50	71.3%	71.3%	71.3%	67.0%	72.5%	72.2%	69.8%	71.0%	72.5%	72.2%	72.5%	0.80
	100	67.0%	67.0%	67.0%	67.0%	71.3%	72.2%	69.1%	69.4%	69.1%	69.8%	69.4%	0.65
	324	71.3%	71.3%	71.3%	68.2%	71.9%	71.9%	71.6%	66.7%	67.3%	66.0%	71.9%	0.50
5	10	70.7%	70.7%	70.7%	70.7%	71.0%	70.7%	70.7%	70.7%	70.7%	70.7%	70.7%	0.00
	25	71.6%	71.6%	71.6%	74.4%	74.1%	75.3%	74.7%	72.8%	74.4%	75.3%	74.7%	0.55
	50	67.6%	67.6%	67.6%	67.9%	80.9%	78.1%	76.2%	78.1%	76.2%	76.9%	78.1%	0.50
	100	73.1%	73.1%	73.1%	66.0%	78.7%	81.5%	77.5%	71.0%	74.4%	75.3%	74.4%	0.80
	324	76.9%	76.9%	76.9%	75.0%	75.3%	75.3%	75.3%	78.7%	77.2%	75.9%	73.8%	0.45
6	10	81.5%	81.5%	81.5%	81.5%	78.7%	78.7%	77.5%	75.6%	77.5%	73.1%	73.1%	0.90
	25	76.5%	76.5%	76.5%	72.8%	75.0%	75.3%	76.5%	77.5%	75.9%	75.6%	76.5%	0.00
	50	79.9%	79.9%	79.9%	72.8%	78.4%	78.1%	77.5%	76.5%	72.8%	74.4%	75.9%	0.55
	100	81.8%	81.8%	81.8%	73.8%	77.8%	78.7%	75.3%	80.6%	77.8%	77.2%	81.8%	0.00
	324	79.0%	79.0%	79.0%	76.9%	73.8%	81.5%	79.9%	80.9%	73.1%	72.8%	79.0%	0.00
7	10	69.4%	69.4%	69.4%	65.4%	63.0%	60.5%	59.6%	67.0%	63.0%	63.0%	69.4%	0.00
	25	66.4%	66.4%	66.4%	67.6%	70.7%	68.8%	67.9%	68.8%	67.3%	68.2%	70.7%	0.40
	50	65.7%	65.7%	65.7%	67.9%	68.8%	70.4%	70.4%	73.1%	73.5%	74.7%	74.7%	0.90
	100	66.0%	66.0%	66.0%	66.7%	66.7%	66.0%	68.5%	70.7%	70.4%	72.5%	72.5%	0.90
	324	59.6%	59.6%	59.6%	64.2%	65.7%	69.8%	67.3%	64.2%	67.9%	65.4%	70.4%	0.45
8	10	91.4%	91.4%	91.4%	90.1%	83.3%	83.6%	81.2%	80.2%	79.9%	75.6%	91.4%	0.00
	25	91.4%	91.4%	91.4%	87.3%	84.3%	89.2%	84.3%	87.0%	83.6%	81.2%	83.6%	0.00
	50	91.7%	91.7%	91.7%	89.5%	88.9%	88.6%	87.7%	87.0%	86.7%	86.1%	91.7%	0.00
	100	90.4%	90.4%	90.4%	91.0%	91.0%	91.0%	91.4%	90.7%	91.0%	89.5%	91.0%	0.30
	324	83.0%	83.0%	83.0%	84.0%	85.2%	86.4%	87.0%	85.8%	86.1%	84.0%	83.0%	0.00
9	10	81.2%	81.2%	81.2%	81.2%	83.6%	84.0%	84.3%	83.3%	85.2%	85.2%	83.3%	0.65
	25	91.4%	91.4%	91.4%	91.4%	91.7%	92.0%	91.7%	92.0%	94.1%	95.1%	91.4%	0.00
	50	95.7%	95.7%	95.7%	95.7%	92.9%	96.0%	95.1%	95.4%	94.4%	95.4%	96.0%	0.50
	100	89.5%	89.5%	89.5%	89.5%	84.0%	88.0%	87.7%	88.3%	85.2%	87.3%	85.2%	0.80
	324	89.8%	89.8%	89.8%	87.3%	92.6%	91.4%	89.2%	88.9%	86.1%	85.5%	89.8%	0.25
10	10	58.0%	58.0%	58.0%	58.0%	58.0%	58.3%	58.3%	59.3%	60.5%	60.5%	60.5%	0.85
	25	66.4%	66.4%	66.4%	66.4%	66.4%	71.0%	64.2%	64.8%	65.1%	65.4%	65.4%	0.75
	50	66.4%	66.4%	66.4%	66.4%	65.4%	71.6%	68.8%	67.6%	63.6%	65.4%	65.4%	0.90
	100	69.8%	69.8%	69.8%	69.1%	69.8%	67.6%	70.4%	70.7%	65.4%	65.1%	70.7%	0.70
	324	66.4%	66.4%	66.4%	70.1%	72.5%	70.4%	67.0%	68.8%	68.2%	65.7%	72.5%	0.40

Table 7.5. Success rate of different neural networks trained with different ρ_as and the crossvalidation process for the first 10 subjects. ρ_a under crossvalidation indicates the ρ_a for those architectures selected by crossvalidation and simulated annealing.

	Performance (Success Rate %) Vigilance Factor (ρa) 0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	Crossvalidation
Ave. (Predictive Performance)	79.0%	79.0%	79.0%	77.6%	78.9%	79.4%	78.7%	78.5%	77.7%	77.3%	80.3%
Std. Dev.	11.6%	11.6%	11.6%	11.2%	10.3%	10.3%	10.4%	10.3%	10.1%	9.9%	10.2%
95% Confidence Upper	80.3%	80.3%	80.3%	79.0%	80.2%	80.8%	80.1%	79.9%	79.1%	78.7%	81.6%
95% Confidence Lower	77.5%	77.5%	77.5%	76.1%	77.4%	78.0%	77.2%	77.1%	76.3%	75.8%	78.9%
Frequency of best answer#	15	15	15	5	11	12	7	6	5	8	26
Frequency of best answer%	30.0%	30.0%	30.0%	10.0%	22.0%	24.0%	14.0%	12.0%	10.0%	16.0%	52.0%
95% Confidence Upper	43.8%	43.8%	43.8%	21.4%	35.2%	37.4%	26.2%	23.8%	21.4%	28.5%	65.2%
95% Confidence Lower	19.1%	19.1%	19.1%	4.3%	12.8%	14.3%	7.0%	5.6%	4.3%	8.3%	38.5%

Table 7.6. 95% confidence intervals for the success rate of the neural networks trained with different ρ_as and the crossvalidation process for the first 10 subjects

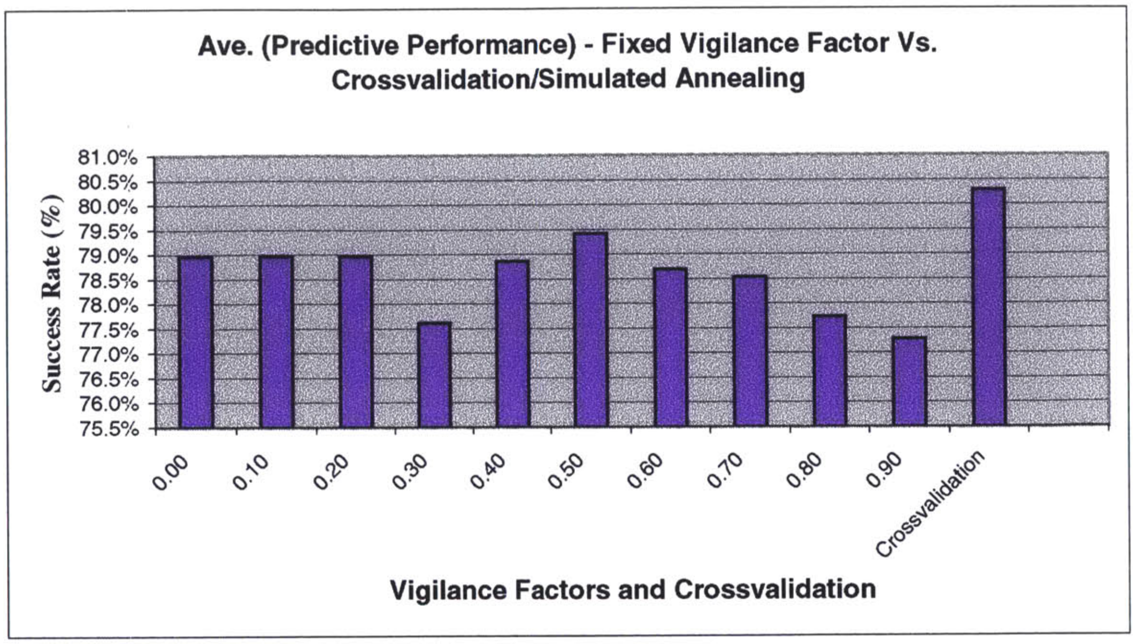


Figure 7.6. Crossvalidation/simulated annealing vs. a fixed vigilance factor

7.2.5 Fuzzy ARTMAP using a Voting Strategy

The implementation, which builds an ensemble of Fuzzy ARTMAP modules, was tested. The problem was to select the required number of Fuzzy ARTMAP modules to be utilized. Therefore, it was decided to test different numbers of modules: 5, 11, 25, 49, and 99. Tables 7.7 and 7.8 show the results for the different ensembles implemented. The ensembles of 25, 49, and 99 modules produce very similar performances. The ensemble of 49 modules was selected due to its performance and computational time.

	Size of	Performance (Success Rate %)				
Subject	Training File	Number of Fuzzy ARTMAP Modules				
		5	11	25	49	99
1	10	97.5%	97.5%	98.8%	98.8%	98.8%
	25	98.8%	98.8%	98.8%	98.8%	98.8%
	50	98.8%	98.8%	98.8%	98.8%	98.8%
	100	98.5%	98.5%	98.5%	98.5%	98.5%
	324	95.7%	97.2%	97.5%	97.5%	98.1%
2	10	75.6%	76.9%	78.4%	78.4%	76.9%
	25	75.0%	74.4%	75.0%	75.0%	74.7%
	50	75.3%	74.4%	75.9%	74.1%	75.3%
	100	78.4%	78.7%	75.9%	79.6%	78.7%
	324	83.6%	84.3%	85.2%	83.3%	85.8%
3	10	90.7%	90.7%	90.7%	90.7%	90.7%
	25	90.7%	90.7%	90.7%	90.7%	90.7%
	50	90.1%	90.1%	90.1%	90.1%	90.1%
	100	88.9%	90.1%	89.2%	90.4%	89.5%
	324	91.7%	92.0%	91.7%	90.7%	90.7%
4	10	65.7%	67.3%	68.5%	67.9%	68.5%
	25	70.1%	71.0%	71.3%	71.9%	72.2%
	50	73.5%	75.0%	76.5%	75.6%	75.6%
	100	69.1%	71.3%	71.6%	71.3%	71.3%
	324	66.4%	68.5%	70.7%	72.5%	71.9%
5	10	70.7%	70.7%	70.7%	70.7%	70.7%
	25	71.9%	71.9%	73.5%	75.0%	75.0%
	50	81.2%	81.5%	81.2%	82.1%	82.1%
	100	78.4%	81.5%	82.1%	83.6%	83.3%
	324	80.9%	83.0%	81.8%	83.3%	83.0%
6	10	70.4%	75.6%	77.5%	70.1%	71.9%
	25	73.8%	76.2%	76.2%	76.9%	76.9%
	50	74.1%	77.5%	79.3%	79.6%	79.9%
	100	82.1%	84.6%	85.5%	85.8%	84.9%
	324	81.8%	84.6%	86.4%	88.0%	88.9%
7	10	67.3%	63.3%	66.7%	66.4%	67.3%
	25	71.6%	71.6%	72.5%	72.5%	71.6%
	50	71.0%	74.7%	72.2%	72.2%	71.6%
	100	71.9%	73.5%	72.5%	73.1%	72.8%
	324	72.2%	72.8%	74.4%	73.5%	70.4%
8	10	90.7%	91.4%	91.4%	91.4%	91.4%
	25	82.4%	82.1%	84.0%	83.6%	83.6%
	50	89.8%	91.7%	91.7%	91.7%	91.7%
	100	88.9%	90.7%	90.1%	90.4%	90.4%
	324	87.7%	90.1%	88.6%	88.9%	88.3%
9	10	85.5%	82.7%	82.1%	82.4%	82.4%
	25	94.1%	94.8%	94.1%	93.5%	94.1%
	50	94.8%	95.4%	95.4%	95.1%	95.4%
	100	86.4%	85.8%	84.6%	84.9%	84.0%
	324	95.7%	93.8%	93.8%	94.4%	92.9%
10	10	61.7%	59.6%	58.0%	59.9%	59.6%
	25	66.7%	67.0%	66.7%	67.6%	66.4%
	50	66.0%	66.7%	64.8%	65.4%	65.4%
	100	69.8%	68.5%	70.1%	71.9%	72.2%
	324	69.1%	71.9%	73.1%	73.5%	71.3%

Table 7.7. Success rate of different Fuzzy ARTMAP ensembles

	Performance (Success Rate %)				
	Number of Fuzzy ARTMAP Modules				
	5	11	25	49	99
Ave. (Predictive Performance)	80.5%	81.2%	81.5%	81.6%	81.5%
Std. Dev.	10.7%	10.7%	10.5%	10.4%	10.5%
95% Confidence Upper	81.8%	82.5%	82.8%	82.9%	82.8%
95% Confidence Lower	79.1%	79.8%	80.1%	80.3%	80.1%
Frequency of best answer#	13	17	21	24	22
Frequency of best answer%	26%	34%	42%	48%	44%
95% Confidence Upper	39.6%	47.8%	55.8%	61.5%	57.7%
95% Confidence Lower	15.9%	22.4%	29.4%	34.8%	31.2%

Table 7.8. 95% confidence intervals for the success rate of the different ensembles

7.1.5 Implementations of Support Vector Machines

The implementations of SVMs have as a backbone the RHUL release 1.0 Software. The computational times were relatively fast. Using the kernel of Linear Splines the answer was approximately 3 seconds. The computational speed of the kernel of RBFs was dependent on γ and the size of the training dataset. It was very fast for the training datasets of 10, 25, 50, and 100 examples (approximately 3 seconds). However, the computational speed for the training dataset of 324 examples and a γ of 0.005 was approximately 7 minutes. The execution of a trained SVM is on the order of milliseconds. The scaling of the inputs (features) was performed from -1 to +1. The output for yes was represented by +1 and the output for no was represented by -1.

7.1.6 Support Vector Machines and Crossvalidation

This thesis developed a mechanism to provide the kernel and γ for RBFs. This mechanism uses genetic algorithms and crossvalidation. It was decided to test this mechanism against the utilization of a fixed kernel and γ. Figure 7.7 and Tables 7.9 and 7.10 show the predictive performance (testing datasets) of several SVMs using Linear Splines and RBFs with different γs. The results indicate that a developed mechanism using crossvalidation and genetic algorithms is a valid process with which to select kernels and parameters for SVMs.

		Performance (Success Rate %)								
Subject	**Size of**	**RBFs and γ**					**Splines**	**Crossvalidation**		
	Training File	**100**	**10**	**1**	**0.1**	**0.01**			**Kernel**	**γ**
1	**10**	49.4%	49.4%	52.2%	97.8%	98.8%	89.5%	98.8%	RBF	0.05
	25	49.4%	49.4%	74.4%	98.8%	98.8%	98.8%	94.4%	RBF	0.5
	50	50.6%	50.6%	94.1%	97.8%	97.8%	96.0%	98.1%	RBF	0.5
	100	49.4%	49.4%	96.9%	93.8%	93.8%	94.1%	97.2%	RBF	0.9
	324	49.4%	49.4%	98.1%	92.0%	89.8%	91.4%	98.5%	RBF	2
2	**10**	76.2%	76.2%	76.2%	76.9%	78.4%	75.9%	76.2%	RBF	1000
	25	76.2%	76.2%	76.2%	76.2%	76.5%	75.9%	76.2%	RBF	0.3
	50	76.2%	76.2%	75.6%	75.3%	78.4%	77.2%	77.2%	Splines	
	100	76.2%	76.2%	77.2%	75.6%	71.6%	71.3%	76.2%	RBF	0.65
	324	76.2%	76.2%	82.7%	77.8%	79.6%	76.9%	80.9%	RBF	0.8
3	**10**	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	RBF	1000
	25	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	90.7%	RBF	1000
	50	90.7%	90.7%	90.7%	89.2%	85.8%	88.0%	89.2%	RBF	0.1
	100	90.7%	90.7%	90.4%	86.4%	86.1%	86.4%	90.7%	RBF	1000
	324	90.7%	90.7%	91.4%	87.7%	85.8%	86.4%	91.7%	RBF	1.1
4	**10**	68.8%	38.9%	74.1%	68.2%	70.7%	70.4%	71.0%	RBF	0.15
	25	68.2%	68.2%	75.9%	75.9%	75.0%	74.1%	75.6%	RBF	0.09
	50	68.2%	68.2%	75.0%	71.0%	69.4%	70.1%	75.9%	RBF	0.5
	100	68.2%	68.2%	73.5%	71.0%	63.6%	64.5%	73.5%	RBF	1
	324	68.2%	68.2%	73.8%	71.9%	71.6%	68.8%	72.2%	RBF	1.75
5	**10**	70.7%	70.7%	70.7%	70.7%	70.7%	70.7%	70.7%	RBF	1000
	25	70.7%	70.7%	70.7%	76.5%	71.6%	76.9%	70.7%	RBF	1000
	50	70.7%	70.7%	74.7%	79.6%	78.7%	76.2%	74.7%	RBF	1
	100	70.7%	70.7%	78.4%	79.0%	77.5%	75.3%	79.9%	RBF	0.65
	324	70.7%	70.7%	82.4%	81.2%	78.1%	77.2%	82.7%	RBF	1.1
6	**10**	76.5%	62.3%	67.6%	79.3%	67.9%	83.0%	83.0%	Splines	
	25	48.8%	48.8%	63.6%	75.3%	77.2%	75.0%	77.2%	RBF	0.01
	50	48.8%	48.8%	74.1%	68.2%	67.3%	72.2%	74.1%	RBF	0.3
	100	51.2%	51.2%	85.8%	76.9%	72.8%	79.0%	85.5%	RBF	1.25
	324	51.2%	51.2%	86.4%	82.4%	82.4%	82.1%	85.8%	RBF	2.5
7	**10**	67.6%	67.6%	69.8%	68.8%	65.7%	66.7%	73.1%	RBF	0.4
	25	67.6%	67.6%	74.4%	60.5%	46.0%	69.8%	74.4%	RBF	1
	50	67.6%	67.6%	71.0%	70.7%	66.7%	71.0%	71.6%	RBF	0.9
	100	67.6%	67.6%	72.8%	71.6%	65.4%	72.2%	74.7%	RBF	0.7
	324	67.6%	67.6%	72.2%	65.7%	65.1%	65.7%	74.1%	RBF	1.5
8	**10**	43.2%	43.2%	46.0%	90.4%	91.4%	82.1%	91.4%	RBF	0.05
	25	56.8%	56.8%	75.0%	74.7%	80.6%	88.3%	88.3%	Splines	
	50	56.8%	56.8%	81.5%	88.3%	86.4%	89.2%	87.7%	RBF	0.45
	100	56.8%	56.8%	89.8%	89.8%	88.0%	90.1%	90.1%	RBF	0.7
	324	56.8%	56.8%	87.7%	87.7%	86.7%	85.5%	89.2%	RBF	1.75
9	**10**	94.8%	94.8%	94.8%	88.0%	82.4%	90.7%	94.8%	RBF	1000
	25	94.8%	94.8%	94.8%	94.8%	93.2%	90.7%	95.7%	RBF	0.6
	50	94.8%	94.8%	94.8%	93.8%	90.4%	90.7%	95.7%	RBF	0.6
	100	94.8%	94.8%	94.4%	88.9%	88.6%	83.6%	94.1%	RBF	0.8
	324	94.8%	94.8%	93.2%	90.1%	88.3%	88.6%	94.4%	RBF	1.25
10	**10**	45.1%	45.1%	46.9%	69.1%	69.1%	63.6%	67.3%	RBF	1000
	25	54.9%	54.9%	59.0%	71.9%	72.8%	63.6%	72.8%	RBF	0.01
	50	54.9%	54.9%	61.7%	69.8%	67.9%	63.0%	70.7%	RBF	0.075
	100	45.1%	45.1%	70.4%	70.7%	70.4%	66.7%	70.4%	RBF	0.1
	324	54.9%	54.9%	73.5%	65.1%	68.5%	67.0%	73.5%	RBF	1

Table 7.9. Success rate of different SVMs trained with different kernels and the crossvalidation process for the first 10 subjects. Kernel and γ under crossvalidation indicates the kernel and γ for those SVMs selected by crossvalidation and genetic algorithms.

	Performance (Success Rate %)						
	RBFs and γ					Splines	Crossvalidation
	100	10	1	0.1	0.01		
Ave. (Predictive Performance)	68.1%	67.2%	78.3%	80.4%	78.8%	79.3%	82.6%
Std. Dev.	15.8%	16.3%	12.5%	10.0%	11.0%	9.9%	9.7%
95% Confidence Upper	69.7%	68.8%	79.6%	81.7%	80.2%	80.7%	83.9%
95% Confidence Lower	66.5%	65.6%	76.8%	79.0%	77.4%	77.9%	81.3%
Frequency of best answer#	8	8	16	8	12	9	31
Frequency of best answer%	16.0%	16.0%	32.0%	16.0%	24.0%	18.0%	62.0%
95% Confidence Upper	28.5%	28.5%	45.8%	28.5%	37.4%	30.8%	74.1%
95% Confidence Lower	8.3%	8.3%	20.8%	8.3%	14.3%	9.8%	48.2%

Table 7.10. 95% confidence intervals for the success rate of the different SVMs and the crossvalidation/genetic algorithms process for the first 10 subjects

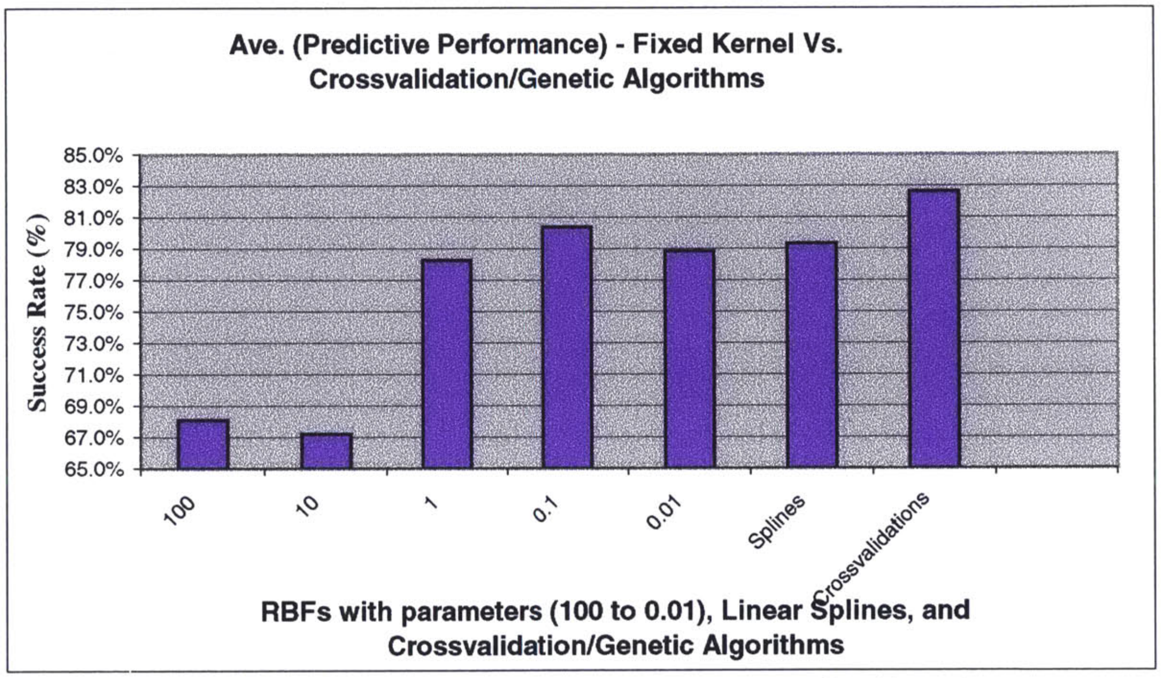


Figure 7.7. Crossvalidation/genetic algorithms vs. fixed kernels

7.3 Results

It is well known that the error from the training dataset is not expected to be a good indicator of future performance. To predict the performance of a learning mechanism, it is required to assess its error rate for a dataset that did not take part in the formation of the learning mechanism. This independent dataset is called the test dataset. It is essential that the test dataset was not used to generate the Backpropagation neural networks, Fuzzy ARTMAP, and SVMs. This thesis uses the testing datasets from the 125 subjects. The following tables show the summary of the predictive performance of the different machine learning paradigms selected (Appendix A shows the predictive performance for each subject and training dataset).

	Training File	Performance (Success Rate %)					
		SVM	FAM	Voting	BP	Pran	Fran
	10 examples						
Ave. (Predictive Performance)		77.1%	73.3%	74.2%	70.7%	50.1%	61.6%
Std. Dev.		13.3%	12.5%	12.5%	12.7%	2.8%	12.7%
95% Confidence Upper		77.5%	73.7%	74.6%	71.1%	50.6%	62.0%
95% Confidence Lower		76.7%	72.8%	73.8%	70.2%	49.7%	61.1%
Frequency of best answer#		84	30	33	27	0	9
Frequency of best answer%		67.2%	24.0%	26.4%	21.6%	0.0%	7.2%
95% Confidence Upper		74.8%	32.2%	34.7%	29.6%	3.0%	13.1%
95% Confidence Lower		58.6%	17.4%	19.5%	15.3%	0.0%	3.8%

Table 7.11. Predictive performance using a learning process with a training dataset of 10 examples for the entire testing dataset of 125 subjects. SVM is for Support Vector Machine, FAM is for Fuzzy ARTMAP, "Voting" is for an ensemble of Fuzzy ARTMAP modules, BP is for Backpropagation, Pran is for Pure Random Generator, and Fran is for Frequency Random Generator.

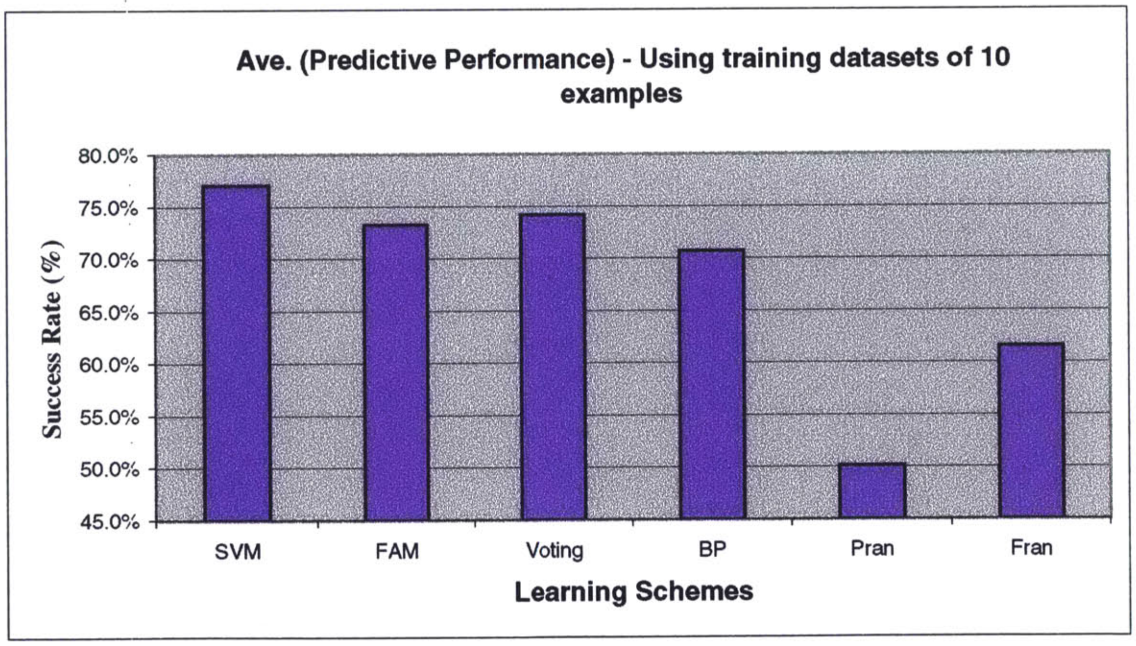


Figure 7.8. Success rates of the different techniques using training datasets of 10 examples and testing datasets of the 125 subjects

SVMs provide a good solution when a reduced number of training examples are available (See Figure 7.8 and Table 7.11). The ensemble of Fuzzy ARTMAP modules outperformed in success rate a single Fuzzy ARTMAP module by 1.23%. The performance of Backpropagation neural networks under conditions of reduced training datasets is very limited. SVMs outperformed Backpropagation by 9.05%. In addition, Fuzzy ARTMAP neural networks outperformed Backpropagation by 3.7%.

	Training	Performance (Success Rate %)					
	File	SVM	FAM	Voting	BP	Pran	Fran
	25 examples						
Ave. (Predictive Performance)		77.8%	74.7%	76.6%	74.2%	50.2%	63.1%
Std. Dev.		12.0%	12.1%	12.6%	12.9%	3.2%	13.5%
95% Confidence Upper		78.2%	75.1%	77.0%	74.6%	50.7%	63.5%
95% Confidence Lower		77.4%	74.3%	76.2%	73.7%	49.7%	62.6%
Frequency of best answer#		67	16	40	25	1	3
Frequency of best answer%		53.6%	12.8%	32.0%	20.0%	0.8%	2.4%
95% Confidence Upper		62.1%	19.8%	40.6%	27.9%	4.4%	6.8%
95% Confidence Lower		44.9%	8.0%	24.5%	13.9%	0.1%	0.8%

Table 7.12. Predictive performance using a learning process with a training dataset of 25 examples for the entire testing dataset of 125 subjects.

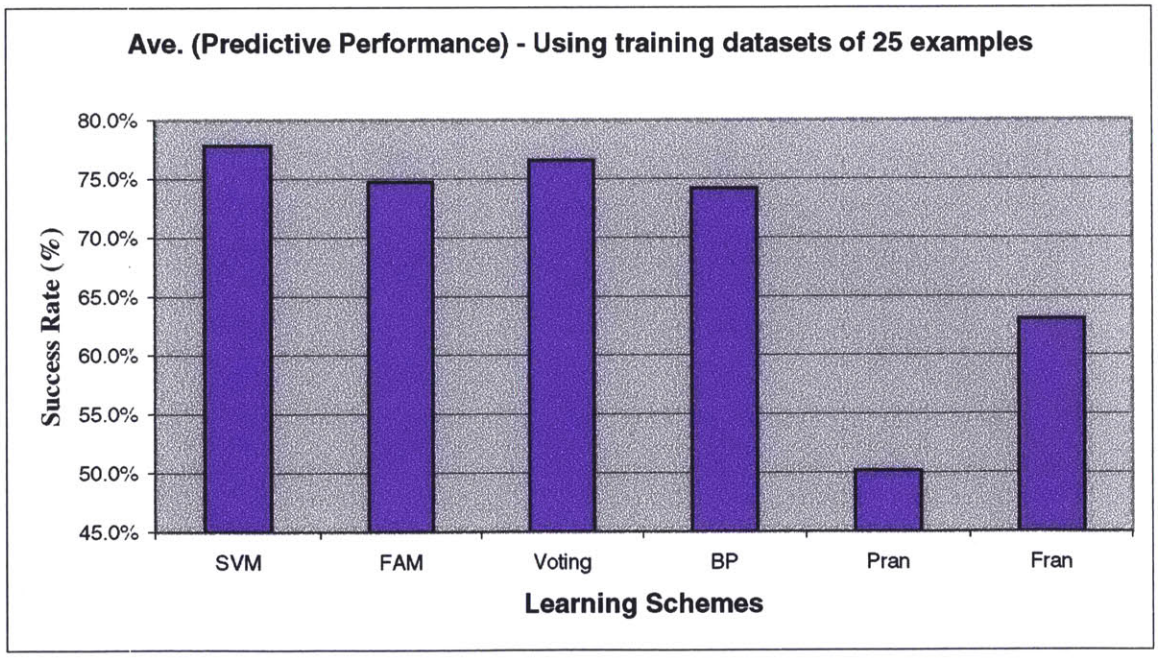


Figure 7.9. Success rates of the different techniques using training datasets of 25 examples and testing datasets of the 125 subjects

The increment of training examples from 10 to 25 increased the performance of SVMs, Fuzzy ARTMAP, the ensemble of Fuzzy ARTMAP modules (49 modules), and Backpropagation. The increase in performance of Backpropagation was very substantial from 70.7% to 74.2% (5%). Again, the ensemble of Fuzzy ARTMAP modules outperformed in success rate a single Fuzzy ARTMAP module, this time by 2.54%. SVMs increased their predictive performance but at a lesser rate. However, SVMs still have the best performance followed by the ensemble of Fuzzy ARTMAP modules. Fuzzy ARTMAP has a better performance than Backpropagation; however, this lead has been reduced from 3.7% to only 0.7%.

	Training File 50 examples	Performance (Success Rate %)					
		SVM	FAM	Voting	BP	Pran	Fran
Ave. (Predictive Performance)		79.0%	76.5%	78.4%	76.2%	49.9%	63.6%
Std. Dev.		11.5%	12.5%	12.4%	11.8%	2.7%	13.4%
95% Confidence Upper		79.4%	76.9%	78.8%	76.6%	50.4%	64.0%
95% Confidence Lower		78.6%	76.1%	78.0%	75.8%	49.5%	63.1%
Frequency of best answer#		58	23	50	17	1	1
Frequency of best answer%		46.4%	18.4%	40.0%	13.6%	0.8%	0.8%
95% Confidence Upper		55.1%	26.1%	48.8%	20.7%	4.4%	4.4%
95% Confidence Lower		37.9%	12.6%	31.8%	8.7%	0.1%	0.1%

Table 7.13. Predictive performance using a learning process with a training dataset of 50 examples for the entire testing dataset of 125 subjects.

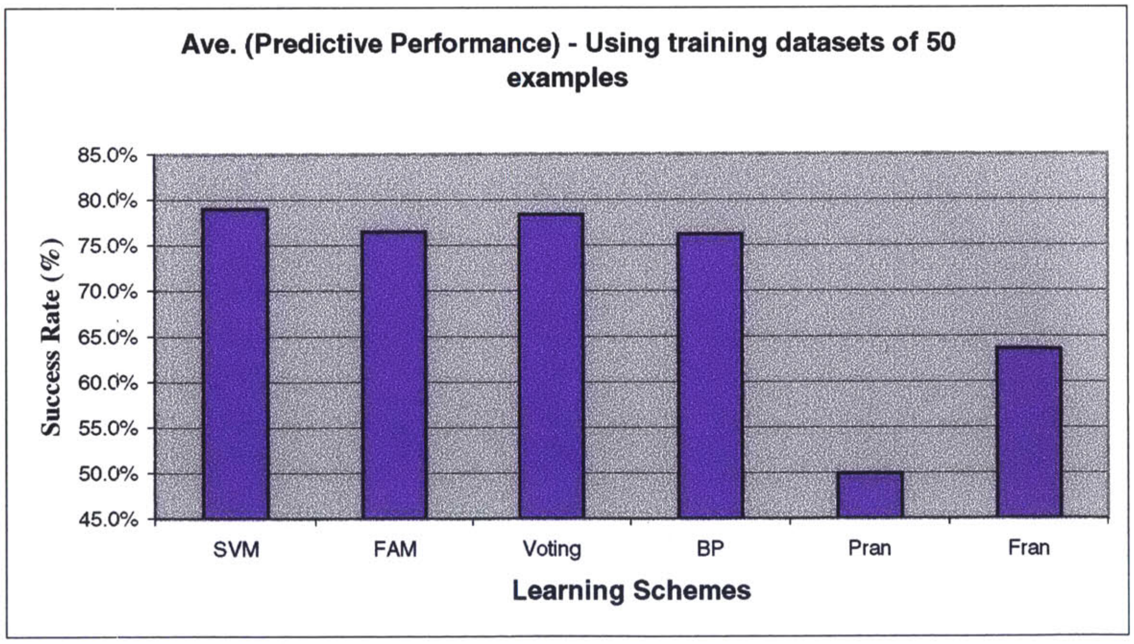


Figure 7.10. Success rates of the different techniques using training datasets of 50 examples and testing datasets of the 125 subjects

The increment of training examples from 25 to 50 increased the performance of SVMs, Fuzzy ARTMAP, the ensemble of Fuzzy ARTMAP modules (49 modules), and Backpropagation (see Figure 7.10 and Table 7.13). The increase in performance of Backpropagation (2.7%), the ensemble of Fuzzy ARTMAP modules (2.35%), and Fuzzy ARTMAP (2.4%) was more pronounced than the increase in performance for SVMs (1.5%). The ensemble of Fuzzy ARTMAP modules outperformed in success rate a single Fuzzy ARTMAP module by 2.5%. The difference in performance between SVMs and BP shrunk from 9.05% with a training dataset of 10 examples to 3.7% with a training dataset of 50 examples. SVMs still have the best performance followed by the ensemble of Fuzzy ARTMAP modules.

	Training	Performance (Success Rate %)					
	File	SVM	FAM	Voting	BP	Pran	Fran
	100 examples						
Ave. (Predictive Performance)		79.2%	76.3%	79.4%	78.4%	49.8%	62.6%
Std. Dev.		11.3%	12.0%	11.9%	11.4%	2.9%	12.9%
95% Confidence Upper		79.6%	76.7%	79.8%	78.8%	50.3%	63.1%
95% Confidence Lower		78.8%	75.9%	79.0%	78.0%	49.3%	62.2%
Frequency of best answer#		42	8	56	24	0	1
Frequency of best answer%		33.6%	6.4%	44.8%	19.2%	0.0%	0.8%
95% Confidence Upper		42.3%	12.1%	53.5%	27.0%	3.0%	4.4%
95% Confidence Lower		25.9%	3.3%	36.4%	13.3%	0.0%	0.1%

Table 7.14. Predictive performance using a learning process with a training dataset of 100 examples for the entire testing dataset of 125 subjects.

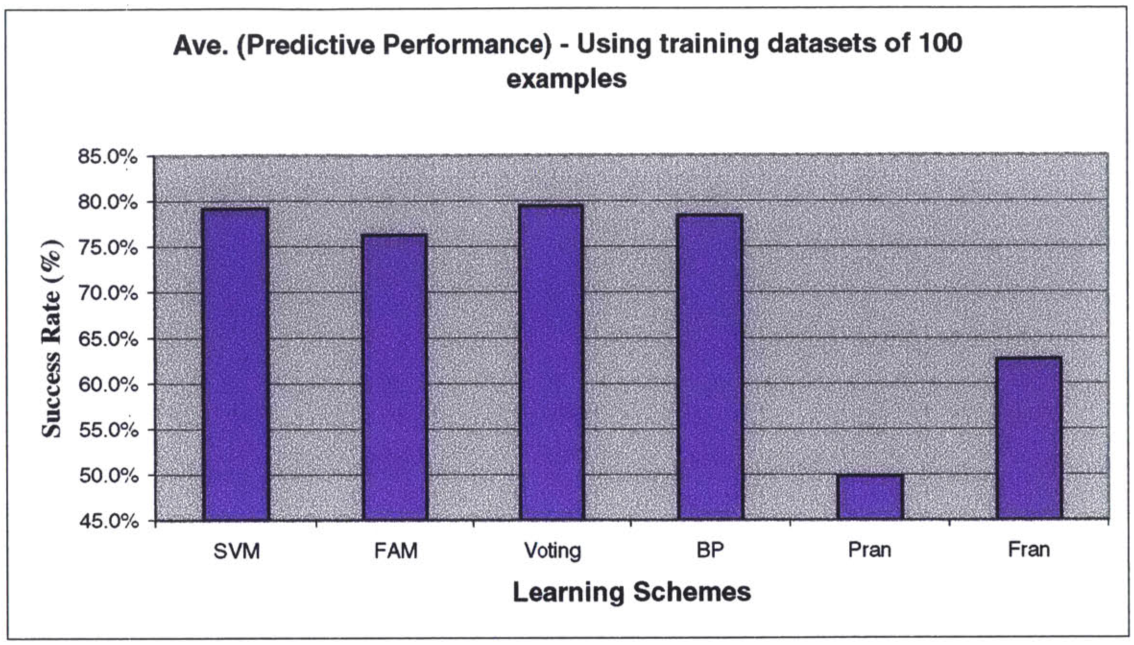


Figure 7.11. Success rates of the different techniques using training datasets of 100 examples and testing datasets of the 125 subjects

The increment of training examples from 50 to 100 increased the performance of SVMs, the ensemble of Fuzzy ARTMAP modules (49 modules), and Backpropagation (see Table 7.14 and Figure 7.11). Fuzzy ARTMAP suffered a decrease in performance (-0.25%, not statistically significant). The increase in performance of Backpropagation (2.9%), and the ensemble of Fuzzy ARTMAP modules (1.28%), was stronger than the increase in performance for SVMs (0.25%). The ensemble of Fuzzy ARTMAP modules outperformed in success rate a single Fuzzy ARTMAP module by 4.07%. The difference in performance between SVMs and BP shrunk from 9.05% with a training dataset of 10 examples to 1.02% with a training dataset of 100 examples. The ensemble of Fuzzy ARTMAP modules has the best performance followed very closed by SVMs (see 95% confidence intervals for both predictive performance and frequency of best answer). In addition, Backpropagation, for the first time, outperformed Fuzzy ARTMAP by 2.75%.

	Training File 324	Performance (Success Rate %)					
		SVM	FAM	Voting	BP	Pran	Fran
Ave. (Predictive Performance)		80.9%	76.2%	80.0%	78.6%	50.2%	63.0%
Std. Dev.		11.0%	12.3%	12.2%	11.6%	2.5%	12.9%
95% Confidence Upper		81.3%	76.6%	80.4%	79.0%	50.7%	63.5%
95% Confidence Lower		80.6%	75.8%	79.6%	78.2%	49.8%	62.5%
Frequency of best answer#		62	7	44	26	0	1
Frequency of best answer%		49.6%	5.6%	35.2%	20.8%	0.0%	0.8%
95% Confidence Upper		58.2%	11.1%	43.9%	28.7%	3.0%	4.4%
95% Confidence Lower		41.0%	2.7%	27.4%	14.6%	0.0%	0.1%

Table 7.15. Predictive performance using a learning process with a training dataset of 324 examples for the entire testing dataset of 125 subjects.

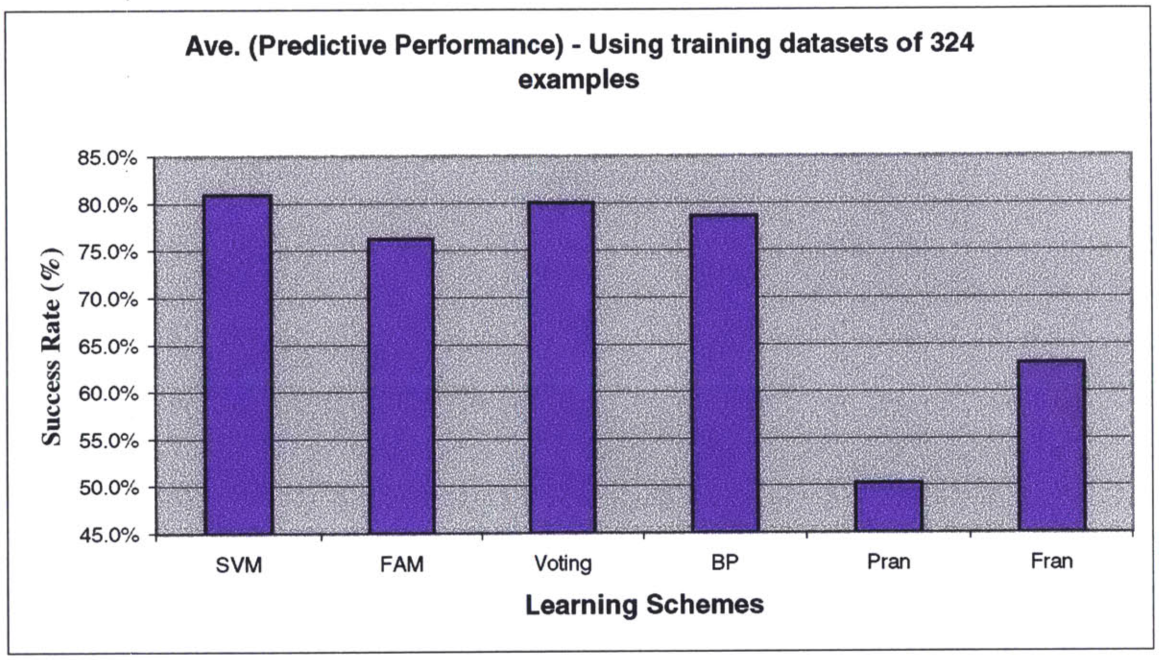


Figure 7.12. Success rates of the different techniques using training datasets of 324 examples and testing datasets of the 125 subjects

The increment of training examples from 100 to 324 increased the performance of SVMs, the ensemble of Fuzzy ARTMAP modules (49 modules), and Backpropagation (see Table 7.15 and Figure 7.12). Fuzzy ARTMAP suffered another decrease in performance (-0.13%, not statistically significant). The increase in performance of SVMs (2.15%) was stronger than the increase in performance for the ensemble of Fuzzy ARTMAP modules (0.8%) and Backpropagation (0.26%). The ensemble of Fuzzy ARTMAP modules outperformed in success rate a single Fuzzy ARTMAP module by 5%. SVMs have the best performance followed by the ensemble of Fuzzy ARTMAP modules. In addition, Backpropagation outperformed Fuzzy ARTMAP again, this time by 3.14%.

	Performance (Success Rate %)					
	SVM	**FAM**	**Voting**	**BP**	**Pran**	**Fran**
Ave. (Predictive Performance)	78.8%	75.4%	77.7%	75.6%	50.1%	62.8%
Std. Dev.	11.9%	12.3%	12.5%	12.4%	2.8%	13.1%
95% Confidence Upper	79.2%	75.8%	78.1%	76.0%	50.6%	63.2%
95% Confidence Lower	78.4%	75.0%	77.3%	75.2%	49.6%	62.3%
Frequency of best answer#	313	84	223	119	2	15
Frequency of best answer%	50.1%	13.4%	35.7%	19.0%	0.3%	2.4%
95% Confidence Upper	54.0%	16.3%	39.5%	22.3%	1.2%	3.9%
95% Confidence Lower	46.2%	11.0%	32.0%	16.2%	0.1%	1.5%

Table 7.16. Predictive performance of the different machine learning paradigms selected for the entire testing dataset of 125 subjects.

In general, it can be said that SVMs have the best success rate (see Table 7.16 and Figure 7.13). The ensemble of Fuzzy ARTMAP modules is able to perform close to SVMs. The ensemble is able to mitigate the problem of the generation of different weights and recognition categories for different orderings of a given training set. The ensemble of Fuzzy ARTMAP modules outperforms a single Fuzzy ARTMAP neural network by 3.05%. This result agrees with the results published by Carpenter et al. [21]. SVMs outperform Backpropagation by approximately 4.23%. This also agrees with the results published by other researchers [9,66]. The performance of Backpropagation is very close to the performance of Fuzzy ARTMAP. However, it is very clear that Backpropagation performance is superior when trained with relatively large datasets (see Figure 7.14).

We would like to mention that, regarding memory requirements, the Backpropagation neural networks (approximately 21 Kbytes) and Fuzzy ARTMAP neural networks (approximately 23 Kbytes), and SVMs (approximately 28Kbytes), had small memory requirements. The ensembles of Fuzzy ARTMAP modules (approximately 250 Mbytes) have medium size memory requirements. In term of runtime performance, Backpropagation neural networks and Fuzzy ARTMAP neural networks were able to generate recommendations in microseconds. The Fuzzy ARTMAP ensembles and SVMs were able to generate recommendations in milliseconds. The learning time was faster for Fuzzy ARTMAP and slower for Backpropagation neural networks and SVMs. Learning times for the different individual models used in these experiments ranged from 3 seconds for Fuzzy ARTMAP neural networks, to minutes for SVMs and Backpropagation neural networks. However, taking into consideration the crossvalidation/search mechanisms and relatively large training datasets, learning ranged from 10 minutes for Fuzzy ARTMAP neural networks, up to 2 hours for Backpropagation neural networks, and up to 4 hours for SVMs.

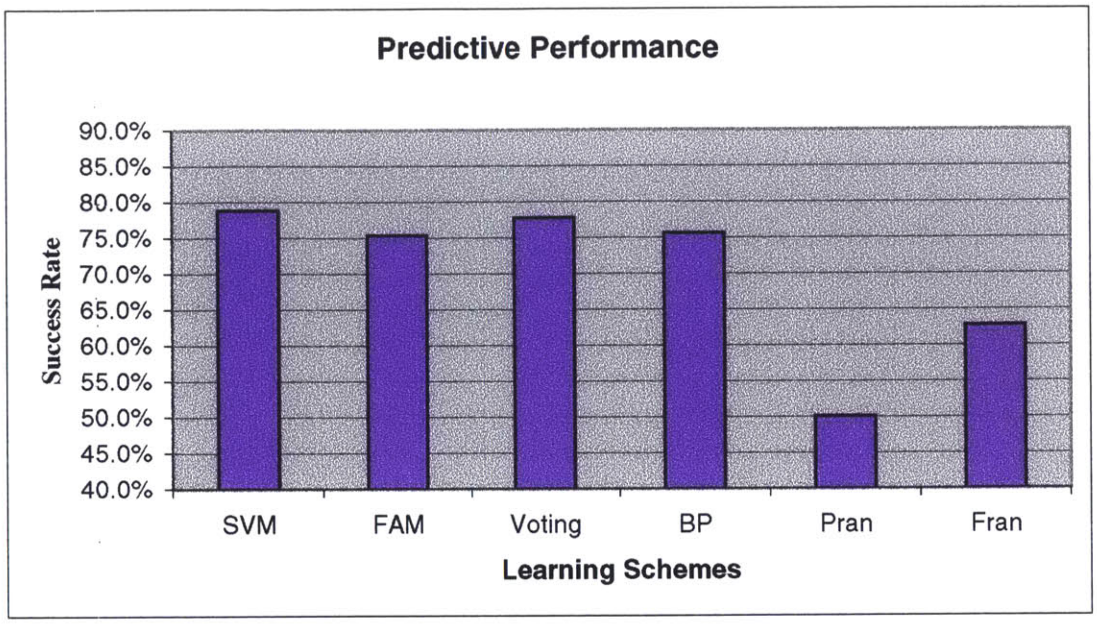


Figure 7.13. Predictive performance (average) of the selected predictive paradigms for the entire experiment

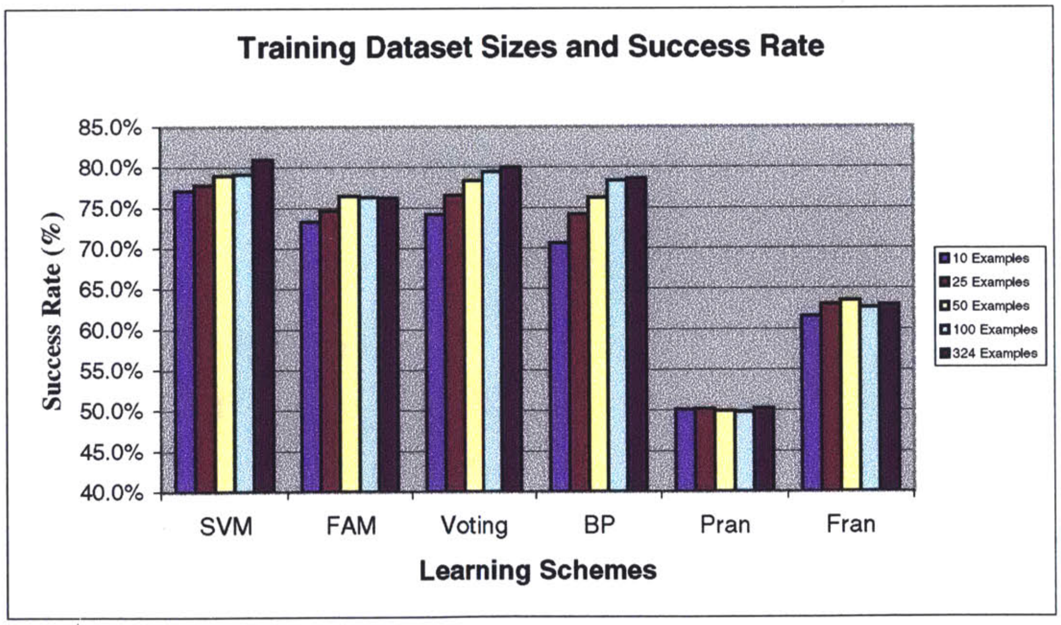


Figure 7.14. Predictive performance of the selected predictive paradigms using different sizes of training datasets.

8 Conclusion

The implementation of intelligent recommendation systems will require the application of sophisticated Internet and machine learning technologies in order to improve the systems' performance. The application of these new technologies will enhance the decision-making capabilities to respond to ever changing environments.

The major focus of this research was the application and comparison of three machine learning techniques – Backpropagation neural networks, Fuzzy ARTMAP neural networks, and Support Vector Machines – to recommendation systems in a representative problem. The different techniques offer several advantages, such as learning and self-improvement.

In this section, the conclusions of this research effort are presented. The section also describes the contributions and recommendations of this research. It closes by discussing future research directions.

8.1 Thesis Summary and Conclusions

This thesis draws from both engineering and management. Domain expertise and solution technologies from systems engineering, software engineering, and electrical engineering are used to provide mechanisms to increase the effectiveness of recommender systems. These intelligent recommendation systems are designed to develop the use of customer-centered management approaches in electronic commerce. The economics of the development and commercialization of these systems are based on the promises of higher levels of customer satisfaction and customer loyalty. The work reported in this research describes the predictive performance of Backpropagation neural networks, Fuzzy ARTMAP neural networks, and Support Vector Machines in a representative problem of feature-based filtering.

Backpropagation neural networks offer execution speed, strong nonlinear learning capabilities, and convenient ways to represent product and user knowledge. Backpropagation neural networks provide a good performance level (even close to that of Support Vector Machines when enough data is available). On the other hand, the learning times are longer than desired for real-time problems. It is very difficult to implement on-line learning schemes with them (therefore, they are reduced to a batch mode). In addition, the learning process is still more "craft" than science. There are a variety of algorithms, from simple gradient-descent to more complex implementations using Bayesian neural network approaches. There are decisions to make such as learning parameters and architectures. In addition, it is very difficult to know which strategy will be the fastest for a given problem. It will depend on many factors such as the complexity of the problem, the number of data points, the number of weights, and the learning goal. And again, the problem of getting trapped in local minima is a possible one! Therefore, to automate its process knowledge-based systems have to be implemented. Tenfold crossvalidation and other statistical schemes can help to complement these knowledge-based systems. Nevertheless, specialized hardware will have to be developed to support the developments of smart agents using Backpropagation. In summary, Backpropagation neural networks are good candidates to implement smart agents. They are not limited to classifier problems but can deal also with regression type problems. The solutions addressed by this approach should be directed to problems where there is enough data, and when changes in the environment are not so frequent, allowing learning to occur in a batch mode.

Fuzzy ARTMAP neural networks offer execution speed and learning capabilities. Product features and user knowledge are able to be represented in its neural network structure (somewhat limited due to the fact that its architecture accepts inputs and outputs scaled between 0 and 1). Fuzzy ARTMAP neural networks provide a competitive performance level; however, they are less accurate than Backpropagation neural networks and Support Vector Machines. On the other hand, the learning times are very fast, meeting the requirements of real-time problems. It is not difficult to implement on-line learning schemes with them. Fuzzy ARTMAP does not have architectural problems. The decision about the vigilance factor is the only problem. This thesis made a contribution to solving this problem by devising an innovative approach that uses crossvalidation and simulated annealing to automate this process. Fuzzy ARTMAP always converge. The problems of lower performance and accuracy can be mitigated by using ensembles of different Fuzzy ARTMAP modules. These Fuzzy ARTMAP modules are trained using the same training dataset but using a different ordering of the examples. These ensembles are able to outperform a single Fuzzy ARTMAP neural network by a 2.5% to 5% margin in success rate. In addition, these Fuzzy ARTMAP ensembles are able to match Backpropagation neural networks and get closer to Support Vector Machines' performance. Another advantage of these ensembles and their voting mechanism is that the user can assign probability estimates to competing predictions given small, noisy or incomplete training sets. This can be a good scheme with which to implement serendipity procedures and conflict resolution strategies. The basic problem is that Fuzzy ARTMAP neural networks are good as classifiers but are not good at implementing regression type solutions. Therefore, their performance using continuous values is questionable. In summary, ensembles of Fuzzy ARTMAP neural networks are good candidates for implementing smart agents. The solutions addressed by this approach should be directed to problems where there are middle levels of historical data on the user's behavior, to where the problems are a classification ones, and to where changes in the environment are frequent (learning can occur on-line).

Support Vector Machines offer execution speed, strong nonlinear learning capabilities, and convenient ways to represent product and user knowledge (We have some reservations here ...We still do not know of any effective approach that can "inject" symbolic rules into the structure of a Support Vector Machine in spite of the virtual support vectors approach [9]). Support Vector Machines provide the best performance level (success rate). This level of increased performance depends on the problem. With smaller training data sets, Support Vector Machines outperformed Backpropagation by almost 10%. With larger training sets, the increase in performance over other machine learning paradigms ranged from 3% to 5%. On the other hand, the learning times are longer than desired for real-time problems. It is very difficult to implement on-line learning schemes with them. In addition, the learning process still is, at some level, more a "craft" (not as bad as Backpropagation neural networks, though) than science despite the claims of researchers. The kernel and the different parameters of the kernel have to be decided in order to achieve high levels of performance. This thesis contributed an innovative methodology using genetic algorithms and tenfold crossvalidation to decide these parameters, using only the training data as guidance. Support Vector Machines converge for the classifier problem and assure the global minima. Support Vector Machines work well with classification and regression type problems. In addition, specialized hardware will have to be developed to support the developments of smart agents using Support Vector Machines. In summary, Support Vector Machines are excellent candidates with which to implement smart agents. One of the advantages is that the solution is based on the support vectors. These support vectors are a subset of the training dataset. Therefore, schemes to implement serendipity and/or increase performance by studying the positions of the hyperplanes and convex hulls (e.g., using orthogonality, studying the densities and distributions of the support vectors in hyperspace) are possible. The solutions addressed by this approach should be directed to problems where there are middle levels of information; however, even with smaller datasets the performance is very good, and changes in the environment are not so frequent, allowing learning to occur in a batch mode.

Table 8.1 shows a summary of the performance of the different machine learning schemes utilized in this thesis. As was stated before, "it is well known that the error on the training dataset is not expected to be a good indicator of future performance." However, the reality is that it sometimes is difficult to have enough information to split the data into training and testing datasets. Tenfold crossvalidation is a good procedure to make these decisions based on a training dataset. Tenfold crossvalidation can even lead to the development of more sophisticated methodologies to decide architectures and learning parameters for the different machine learning techniques (as developed in this thesis by combining crossvalidation with simulated annealing and genetic algorithms). Two important cautions, learned from this work, to take into consideration are (1) getting a good measure of performance is a computation-intensive undertaking [100], and (2) tenfold crossvalidation is not a silver bullet (it might not be enough to get a reliable error estimate).

	BP*	**FAM***	**Voting**	**SVM***
Predictive Performance **(Depending on Training Dataset Size)**				
10 Examples	70.7%	73.3%	74.2%	77.1%
25 Examples	74.2%	74.7%	76.6%	77.8%
50 Examples	76.2%	76.5%	78.4%	79.0%
100 Examples	78.4%	76.3%	79.4%	79.2%
324 Examples	78.6%	76.2%	80.0%	80.9%
Predictive Performance **(Overall)**	75.6%	75.4%	77.7%	78.8%
Learning Speed*+ **(Training dataset of 10 Examples)** ***without specialized hardware**	2 Minutes	20 Seconds	40 Seconds	2 Minutes
Learning Speed* **(Training dataset of 324 Examples)** ***without specialized hardware**	2 Hours	10 Minutes	12 Minutes	4 Hours
Execution Speed* **(to provide a recommendation)** ***without specialized hardware**	Microseconds	Microseconds	Milliseconds	Milliseconds
Memory Requirements **(Learning)**	Medium	Low	Low	Medium
Memory Requirements **(Execution)**	Low	Low	Medium	Medium-Low

Table 8.1. Relative performance of the various learning schemes to develop recommendation systems

8.2 Recommendations for Further Research

The remarkable advances in technological systems such as the Internet will require remarkable changes to existing approaches and methodologies for their design and management. Complexity is the expression that describes the different concepts, forms, functions, interdependencies, and challenges of systems such as recommender software agents. This thesis is only a small step toward a more holistic approach. This work has created the infrastructure and the willingness to start pursuing a more ambitious research agenda to make intelligent agents smarter.

This research has presented a comparison in predictive performance of several machine learning techniques to develop smart agents. The experience gained from this research also suggests some areas of allied research. Follow-up studies are needed to perform an in-depth comparative analysis using a variety of operational conditions. First, the literature survey indicated that feature-based filtering agents learn more slowly initially. This opens the hypothesis that the ideal learning session should be able to provide initial knowledge based on collaborative filtering to these feature-base filtering approaches. Second, the approach presented in this thesis attempts to optimize the short-term performance. Therefore, there are compelling benefits to try to optimize long-term performance (and compensate for the loss of serendipity). Finally, the willingness of the target user to provide an answer immediately after the recommendation was made is an assumption utilized in this research; however, the interactions between smart agents and target users can involve temporal issues and delay effects too. Therefore, there is also value in studying the temporal and delay effects on the learning. These areas of allied research are explained in the next subsections.

8.2.1. Study of the Performance of Backpropagation, Fuzzy ARTMAP, and Support Vector Machines

The experiments conducted in this research form just a single branch of analysis. An in-depth study of the performance of the different algorithms under different conditions should be initiated. These scenarios should include different scenarios, continuous values to assign user preferences, operational conditions, statistical sensitivity analysis to know what product attributes are important to a target user (before the machine learning scheme is even trained), the effect of adding initial knowledge based on collaborative filtering, the synergistic effect of intrinsic symbolic techniques (e.g., expert systems), and stochastic factors. In addition, to improve these schemes, a more comprehensive study that includes not only the past behavior of the user but other information such as [83]

- *Current shopping needs*
- *E-commerce preferences (buying methods, merchants, products)*
- *Travel information (airline, car rental, hotel, restaurant, itinerary)*
- *Grocery/consumable purchases*
- *Other purchases (books, software, music, clothing, electronics, 'luxury items')*
- *Wish list*
- *Interests, preferences, favorite activities*
- *Browsing habits (history, cookies, clickstream data)*
- *Chat/ICQ/Zephyr logs*
- *Credit card records*
- *Financial records*
- *Yearly salary*
- *Address*
- *Email address*
- *Demographic info*
- *Date of birth*

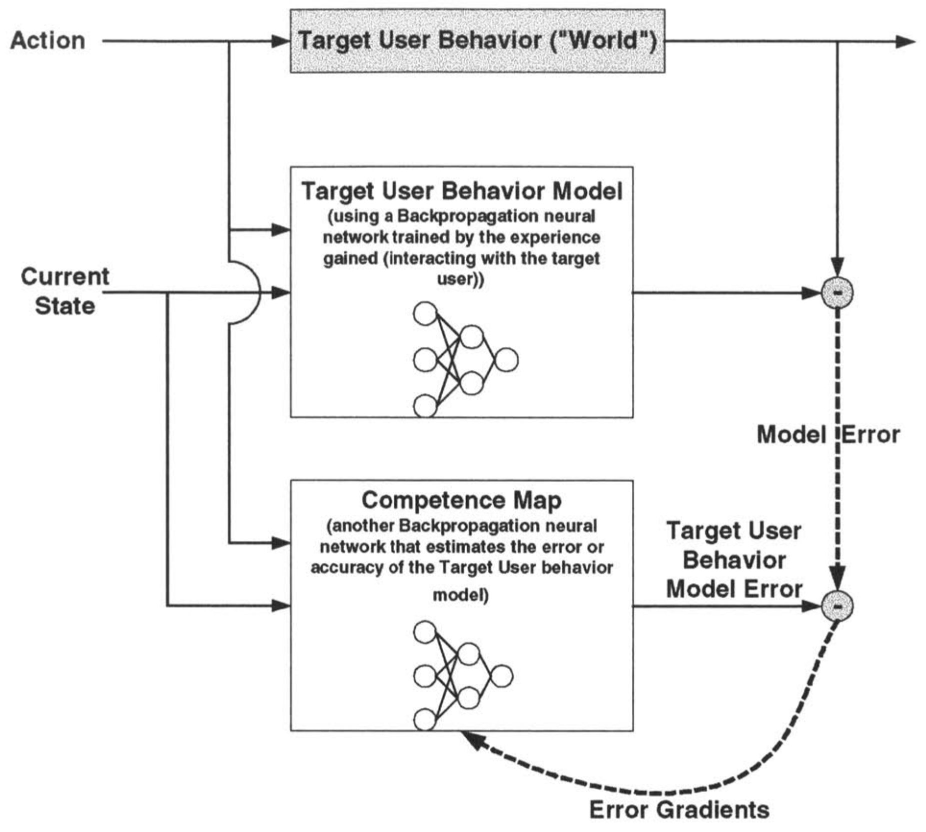
should be addressed. This will allow the system to build a better profile of the target user. The approach must be flexible to fuse the information that is available in order to get higher predictive performance. This will bring in other issues, such as privacy and consumer protection, to be investigated and added to the smart agent's framework. Privacy laws to protect the privacy and rights of the consumers on the Internet must be respected.

8.2.2. Exploring an Unknown Environment (Compensating for the Loss of Serendipity)

Whenever a smart agent learns, two opposing principles have to be combined: exploration (long-term optimization) and exploitation (short-term optimization). This thesis during the testing phase has only used the side of exploitation in order to get higher performance levels (this might lead to the loss of serendipity). The idea that could be researched in future work is to make an agent explore unknown regions in a more directed manner. In addition, this research can compensate for the loss of serendipity and increase the ability to discover new products and unexpected joys!

8.2.2.1. Backpropagation Neural Networks.

Thrun and Moller [86] developed the concept of a competence map which is used for guiding exploration. Based on their ideas, a bistable system enables the smooth switching of attention between the two behaviors – exploration and exploitation – depending on knowledge gain.



Action
Target User Behavior ("World")
Target User Behavior Model
(using a Backpropagation neural network trained by the experience gained (interacting with the target user))
Current State
Model Error
Competence Map
(another Backpropagation neural network that estimates the error or accuracy of the Target User behavior model)
Target User Behavior Model Error
Error Gradients

Figure 8.1. Exploring an unknown environment using Backpropagation

Therefore, the competence map estimates the accuracy of the Target User Behavior Model neural network based on Backpropagation. This estimation is used for exploring the world by selecting actions (i.e., product selections) which minimize the expected competence of the Target User behavior model, and thus maximize the resulting learning effect (and create nice surprises for the target user!).

8.2.2.2. Fuzzy ARTMAP Neural Networks

It is possible to implement a similar mechanism by using the ensemble of Fuzzy ARTMAP modules. We have explained that the reason for these ensembles is to train several Fuzzy ARTMAP neural networks to mitigate the problem of the formation of different weights structures due to the sequence of examples of the training dataset. The final prediction for a given test set item is the one made by the largest number of simulations. Since the set of items making erroneous predictions varies from one simulation to the next, voting cancels many of the errors. The user can assign probability estimates to competing predictions given small, noisy or incomplete training sets. This can be a good scheme with which to implement serendipity procedures and conflict resolution strategies. For example, Figure 8.2 details the results of voting for 324 testing examples using an ensemble of 25 Fuzzy ARTMAP modules. 25 means (Y Axis) that all 25 Fuzzy ARTMAP modules voted for the "yes" category and 0 means that all 25 Fuzzy ARTMAP modules voted for the "no" category. A number of 12 means 12 Fuzzy ARTMAP modules voted for the "yes" category and 13 voted for the "no" category. This voting scheme can tell us where the system needs to explore more in order to solve conflicts of knowledge (i.e., the system is not sure of the answer!).

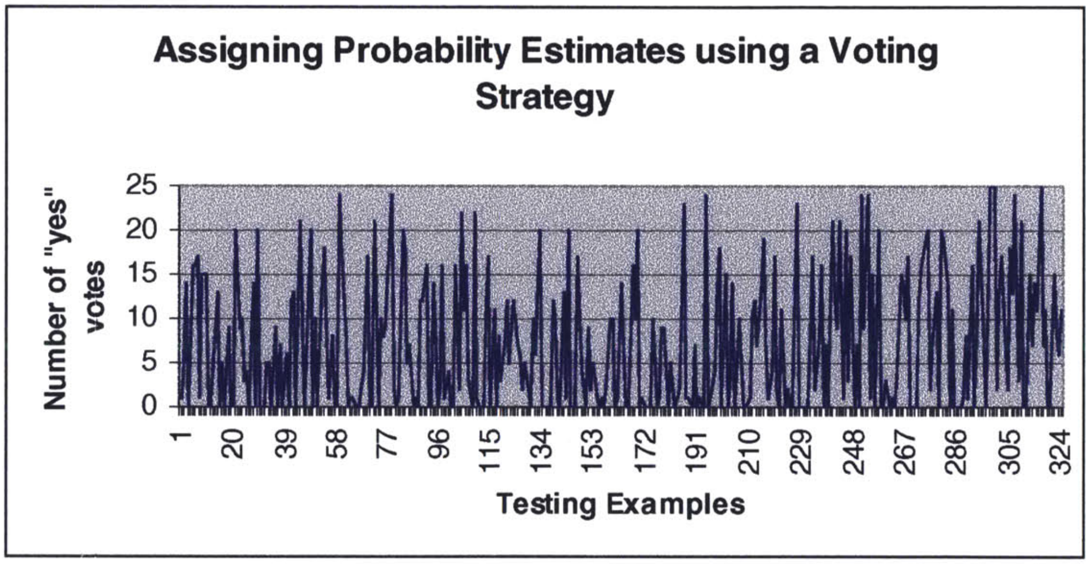


Figure 8.2. Assigning probability estimates using a voting strategy and an ensemble of 25 Fuzzy ARTMAP modules (For instance, the testing example #20 has 0 "yes" votes; therefore, it has a probability of 0 of being category "yes." On the other hand, testing example #305 has 18 "yes" votes; therefore, it has a probability of 0.72 of being category "yes")

8.2.2.3. Support Vector Machines

SVMs select support vectors to implement their classification strategy. These support vectors are a subset of the training dataset. Therefore, schemes to implement serendipity and/or increase performance by studying the positions of the hyperplanes and convex hulls (e.g., using orthogonality, studying the densities and distributions of the support vectors in the hyperspace) are possible.

For instance, the densities and distributions of the support vectors in the hyperspace can be studied. Those regions with low densities should be studied and see how new support vectors can expand a convex hull and/or redefine boundaries. Orthogonality is a good concept to start looking for candidate recommendations to the user and thus expands the knowledge base of the system and improving the accuracy of prediction.

8.2.3 Reinforcement Learning and Temporal Reasoning Using Neural Networks

Reinforcement learning is different from supervised learning. Supervised learning is learning from examples provided by some knowledgeable external supervisor. Supervised learning is not adequate for learning when temporal delays from the external supervisor are present. As expressed by Sutton and Barto [85], "In interactive problems it is often impractical to obtain examples of desired behavior that are both correct and representative of all the situations in which the agent has to act. In uncharted territory---where one would expect learning to be most beneficial---an agent must be able to learn from its own experience." This type of learning could be useful in situations such as intelligent recommendation systems that must forget past user experiences that are too old. In addition, the target user might make a decision sometime after the product was recommended (e.g., Amazon.com recommends to the target user a book, and the user buys this book sometime (e.g., 1 month) after the recommendation has been made).

In the reinforcement learning scheme, an agent continually senses the environment, selects actions (e.g., specific product recommendations) to affect the environment and, after each action, receives from the environment a reinforcement signal (the user bought the recommended product (positive reinforcement) or the user did not buy the product (negative reinforcement)). This reinforcement signal can be positive (a reward), negative (a punishment), or simply "nothing." The objective of learning is to construct an optimal action selection policy that maximizes the agent's performance. As expressed by Sutton and Barto [85], the discounted cumulative reinforcement learning is explained as follows:

$$\mathbf{R_t = r_{t+1} + \gamma\, r_{t+2} + \gamma^2\, r_{t+3} + \ldots = \Sigma\, \gamma^k\, r_{t+k+1},}$$

Where $\mathbf{R_t}$ is the discounted cumulative reinforcement starting from time **t** throughout the future, and γ is a parameter called the discount rate. The discount rate (also called the forgetting factor) determines the present value of future rewards: "a reward received **k** time steps in the future is worth only γ^{k-1} times what it would be worth if it were received immediately."

These techniques have been extended to neural networks. Backpropagation neural networks can be trained by using two different algorithms: Q-Learning from Watkins [94] and Heuristic Dynamic Programming (HDP) from Paul Werbos [98]. For instance, Q-Learning (see Figure 8.3) offers a procedure for a smart agent to learn to select desirable products given the characteristics (profile) of a consumer. Q-learning attempts to estimate value functions for selecting the best product for a particular consumer profile. The values reflect the immediate and (discounted) long term expected reward for making a given selection. These "Q" values are conceptually, and often in practice, stored in a table that enumerates the possible state/action pairs [75,95]. Again, Q-Learning may be seen as online stochastic dynamic programming. Lacking a world model to allow offline search, the Q learner uses the world as its own model. However, versions that develop internal models of the world are available. These versions of Q-Learning with internal models are considered superior to the original Q-Learning formulation.

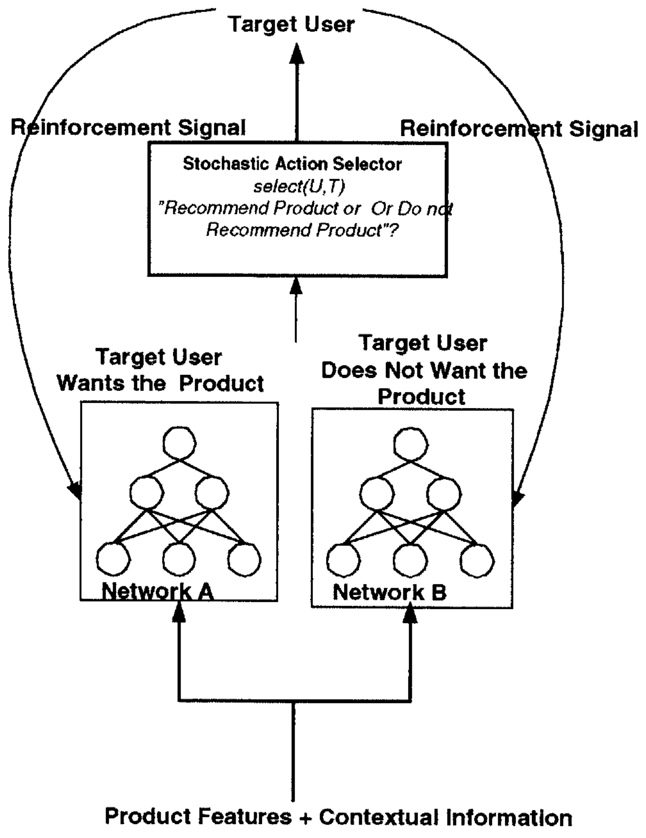

Target User
Reinforcement Signal
Reinforcement Signal
Stochastic Action Selector
select(U,T)
"Recommend Product or Or Do not Recommend Product"?
Target User Wants the Product
Target User Does Not Want the Product
Network A
Network B
Product Features + Contextual Information

Figure 8.3. Q-learning for intelligent recommendation systems

9 Bibliography

[1] A. Akaike. "Statistical predictor identification," *Annals of the Institute of Statistics and Mathematics*, vol. 22, pp. 203-217, 1970.

[2] D. Ariely. *Career Proposal*, MIT Sloan School of Management, 2000.

[3] D. Ariely, J. Lynch and M. Aparicio. "Which Intelligent Agents Are Smarter? An Analysis of Relative Performance of Collaborative and Individual Based Recommendation Agents," MIT Sloan School of Management, 2000.

[4] A. Barron. "Predicted Squared Error: A Criterion for Automatic Model Selection," *Self-Organizing Methods in Modeling*, Marcel Dekker, Inc., 87-102, 1984.

[5] J. Bradshaw (Editor), *Software Agents*, MIT press, 1997.

[6] J. Breese, D. Heckerman and C. Kadie. "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 1998.

[7] E. Brynjolfsson and M. Smith. The Great Equalizer? Consumer Choice Behavior at Internet Shopbots, MIT Sloan White Paper, 2000.

[8] C. Burges. "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, No. 2, pp. 121-167, 1998.

[9] C. Burges and B. Scholkopf. "Improving the Accuracy and Speed of Support Vector Machines," *Advances in Neural Information Processing Systems 9*, Edited by M. Mozer, M. Jordan and T. Petsche, Morgan Kaufmann Publishers, pp.375-381, 1998.

[10] G. Carpenter. "Neural Network Models for Pattern Recognition and Associative Memory," *Neural Networks*, 1989, vol. 2, pp. 243-257.

[11] G. Carpenter. "Analysis of ART 2," *Neural Networks: From Foundations to Applications*, 1991, vol. 1, Lecture 15, pp. 361-386.

[12] G. Carpenter. "Distributed Learning, Recognition, and Prediction by ART and ARTMAP Neural Networks," *Neural Networks*, vol. 10, No. 8, pp. 1473-1494, 1997.

[13] G. Carpenter and S. Grossberg. "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine," *Computer Vision, Graphics, and Image Processing*, 1987, vol. 37, pp. 54-115.

[14] G. Carpenter and S. Grossberg. "ART 2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns," *Applied Optics*, 1987, vol. 26, pp. 4919-4930.

[15] G. Carpenter and S. Grossberg. "ART 2:Self-Organization of Stable Category Recognition Codes for Analog Input Patterns," *Proceedings of the IEEE First International Conference on Neural Networks*, Edited by M. Caudill and C. Butler, 1987, vol. 2, pp. 727-736.

[16] G. Carpenter and S. Grossberg. "The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network," *Computer*, March 1988, pp. 77-88.

[17] G. Carpenter and S. Grossberg. "ART 3: Hierarchical Search Using Chemical Transmitters in Self-Organizing Pattern Recognition Architectures," *Neural Networks*, 1990, vol. 3, pp. 129-152.

[18] G. Carpenter, S. Grossberg, and J. Reynolds. "ARTMAP: Supervised Real-Time Learning and Classification of Nonstationary Data by a Self-Organizing Neural Network," *Pattern Recognition By Self-Organizing Neural Networks*, Edited by G. Carpenter and S. Grossberg, MIT Press, 1991, pp. 503-544.

[19] G. Carpenter, S. Grossberg and D. Rosen. "Fuzzy ART," Poster paper presented at the *Neural Networks for Vision and Image Processing Conference*, Wang Institute of Boston University, May 10-12, 1991.

[20] G. Carpenter, S. Grossberg and D. Rosen. "ART2-A: An Adaptive Resonance Algorithm for Rapid Category Learning and Recognition," *Neural Networks*, vol. 4, pp. 493-504, 1991.

[21] G. Carpenter, S. Grossberg, N. Markuzon, J. Reynolds and D. Rosen. "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps," *Technical Report CAS/CNS-TR-91-016*, 1991.

[22] G. Carpenter, S. Grossberg, N. Markuzon, J. Reynolds and D. Rosen. "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps," *IEEE Transactions on Neural Networks*, vol. 3, No. 5, pp. 698-713, 1992.

[23] G. Carpenter, S. Grossberg and J. Reynolds. "A Fuzzy ARTMAP Nonparametric Probability Estimator for Nonstationary Pattern Recognition

Problems," *IEEE Transactions on Neural Networks*, vol. 6, No. 6, pp. 1330-1336, 1995.

[24] G. Carpenter and N. Markuzon. "ARTMAP-IC and medical diagnosis: Instance counting and inconsistent cases," *Neural Networks*, vol. 11, No. 2, pp. 323-336, 1998.

[25] G. Carpenter, B. Milenova and B. Noeske. "Distributed ARTMAP: a neural network for fast distributed supervised learning," *Neural Networks*, vol. 11, No. 5, pp. 793-813, 1998.

[26] J-C. Charlet and E. Brynjolfsson, "Firefly Network," Case S-OIT-22, Graduate School of Business, Stanford University, 1998.

[27] J-C. Charlet and E. Brynjolfsson, "Rule-Based Systems," Graduate School of Business, Stanford University, 1998.

[28] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*, Cambridge University Press, 2000.

[29] V. Darley, Towards a Theory of Autonomous, Optimising Agents. Ph. D. Dissertation, Harvard University, 1999.

[30] S. Davies, A. Marshall and R. Martin. "Fusing Multiple Sources with Bayesian Networks to Achieve Accurate Object Descriptions," SPIE, Vol. 2589, pp. 79-90, 1995.

[31] Defense Advanced Research Projects Agency, Information Systems Office, Control of Agents-Based Systems (CoABS) Program, http://dtsn.darpa.mil/iso/index2.asp?mode=9.

[32] H. Demuth and M. Beale, *Neural Network Toolbox (MATLAB),* The MATH WORKS Inc., 1998.

[33] R. Fletcher and C. Reeves, "Function minimization by conjugate gradients," *Computer Journal*, vol. 7, pp. 149-154, 1964.

[34] S. Geisser. "The predictive sample reuse method with applications," *Journal of the American Statistical Association,* vol. 70, No. 350, *1975.*

[35] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Co., 1989.

[36] N. Good, J. Schafer, J. Konstan, A. Borchers, B. Sarwar, J. Herlocker and J. Riedl. "Combining Collaborative Filtering with Personal Agents for Better

Recommendations," Proceedings of the 1999 Conference of the American Association of Artificial Intelligence, 1999.

[37] D. Greening. "Building Consumer Trust with Accurate Product Recommendations," A White Paper on LikeMinds Personalization Server, 2000.

[38] S. Grossberg. "Some Nonlinear Networks Capable of Learning a Spatial Pattern of Arbitrary Complexity," *Proceedings of the National Academy of Sciences USA*, 1968, vol. 59, pp. 368-372.

[39] S. Grossberg. "Adaptive Pattern Classification and Universal Recoding, I: Parallel Development and Coding of Neural Feature Detectors," *Biological Cybernetics*, 1976, vol. 23, pp. 121-134.

[40] S. Grossberg. "Competitive Learning: From Interactive Activation to Adaptive Resonance," *Connectionist Models and Their Implications: Readings From Cognitive Science*, Edited by D. Waltz and J. Feldman, Ablex Publishing Corporation, 1988, pp. 243-283.

[41] S. Grossberg. "Nonlinear Neural Networks: Principles, Mechanisms, and Architectures," *Neural Networks*, 1988, vol. 1, pp. 17-61.

[42] M. Hagan, H. Demuth, and M. Beale, *Neural Network Design*, PWS Publishing Company, 1995.

[43] W. Hanson. *Principles of Internet Marketing*, South-Western College Publishing, 2000.

[44] A. Hodgkin and A. Huxley, "A Quantitative Description of Membrane Current and Its Applications to Conduction and Excitation in Nerve," *The Journal of Physiology*, vol. 117, 1952, pp. 500-544.

[45] J. Holland. *Adaptation in Natural and Artificial Systems*, MIT Press, 2 edition, 1975,1992.

[46] E. Horvitz, J. Breese, D. Heckerman, D. Hovel and K. Rommelse. "The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users," Microsoft Research, 1999.

[47] R. Kohavi. "A study of crossvalidation and bootstrap for accuracy estimation and model selection," *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence,* Morgan Kaufman Publishers, Inc., pp. 1137-1143, 1995.

[48] A. Krogh and J. Hertz. "A Simple Weight Decay Can Improve Generalization," *Advances in Neural Information Processing System 4*, Edited by J. Moody, S. Hanson, and R. Lippmann, Morgan Kaufman Publishers, Inc., pp. 950-957, 1994.

[49] Y. Le Cun, J. Denker and S. Solla. "Optimal Brain Damage," *Neural Information Processing Systems*, Edited by D. Touretzky, vol. 2, 1990.

[50] H. Lieberman. "Letizia: An Agent that Assists Web Browsing," *Proceeding of IJCAI 95*, AAAI Press, 1995.

[51] C. Lim and R. Harrison. "Modified Fuzzy ARTMAP approaches Bayes optimal classification rates: an empirical demonstration," *Neural Networks*, vol. 10, No. 4, pp. 755-774, 1997.

[52] C. Lueg and C. Landolt. "A Java-Based Approach to Active Collaborative Filtering," *CHI 1998 Workshop Proceedings*, 1998.

[53] P. Lyman and H. Varian. *Project Report: How Much Information?* University of California - Berkeley's School of Information Management and Systems, 2000.

[54] D. Mackay. *Bayesian Methods for Adaptive Models*, Ph.D. Dissertation, California Institute of Technology, 1992.

[55] D. Mackay. "Bayesian Model Comparison and Backprop Nets," *Advances in Neural Information Processing System 4*, Edited by J. Moody, S. Hanson, and R. Lippmann, Morgan Kaufman Publishers, Inc., 839-846, 1994.

[56] D. Mackay. "Probable networks and plausible predictions - a review of practical Bayesian methods for supervised neural networks," *Network: Computation in Neural Systems*, Vol. 6, 469-505, 1995.

[57] P. Maes. "Agents that Reduce Work and Information Overload," *Communications of the ACM*, vol. 37, No. 7, pp. 31-40. 1994.

[58] P. Maes. "Artificial Intelligent meets Entertainment: Lifelike Autonomous Agents," *Communications of the ACM*, vol. 38, No. 11, pp. 108-114. 1995.

[59] S. Marriot and R. Harrison. " A modified Fuzzy ARTMAP architecture for the approximation of Noisy Mappings," *Neural Networks*, vol. 8, No. 4, pp. 619-641, 1995.

[60] B. Miller, J. Riedl and J. Konstan. "Experiences with GroupLens: Making Usenet Useful Again," *Proceedings of the 1997 Usenix Winter Technical Conference*, 1997.

[61] J. Moody."The Effective Number of Parameters: An Analysis of Generalization and Regularization in Nonlinear Learning Systems," *Advances in Neural Information Processing System 4*, Edited by J. Moody, S. Hanson, and R. Lippmann, Morgan Kaufman Publishers, Inc., 683-690, 1994.

[62] J. Moody and J. Utans."Principled Architecture Selection for Neural Networks: Application to Corporate Bond rating Prediction," *Advances in Neural Information Processing System 4*, Edited by J. Moody, S. Hanson, and R. Lippmann, Morgan Kaufman Publishers, Inc., 683-690, 1994.

[63] N. Morgan and H. Bourlard. *Generalization and Parameter Estimation in Feedforward Nets: Some Experiments*, International Computer Science Institute, Technical Report, TR-89-017, 1989.

[64] R. Murch and J. Johnson. *Intelligent Software Agents*, Prentice Hall, 1998.

[65] H. Nwana. "Software Agents: An Overview," *Knowledge Engineering Review*, vol. 11, No. 3, pp. 205-244, 1996.

[66] E. Osuna, R. Freund and F. Girosi. "An Improved Training Algorithm for Support Vector Machines," *Proceedings of the IEEE Workshop on Neural Networks and Signal Processing*, 1997.

[67] C. Petrie. "Agent-based Engineering, the Web, and Intelligence," IEEE Expert, 1996.

[68] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 1988.

[69] T. Plate, P. Band, J. Bert and J. Grace. "A comparison between neural networks and other statistical techniques for modeling the relationship between tobacco and alcohol and cancer," *Advances in Neural Information Processing System 9*, Edited by M. Mozer, M. Jordan, and t. Petsche, The MIT Press, 967-973, 1997.

[70] J. Platt. "How to implement SVMs," *IEEE Intelligent Systems*, July/August 1998.

[71] J. Platt. "Fast training of support vector machines using sequential minimal optimization," *Advances in Kernel Methods – Support Vector Learning*,

Edited by B. Scholkopf, C Gurges, and A. Smola, The MIT Press, pp. 185-208, 1999.

[72] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C*, Cambridge University Press, 1993.

[73] D. Rumelhart, J. McClelland and the PDP Research Group. *Parallel Distributed Processing: explorations in the microstructure of cognition*, Vol. 1: Foundations, Cambridge, MIT Press/Bradford Books.

[74] K. Ryan. Success Measures of Accelerated Learning Agents for e-Commerce, MS Thesis, MIT, September 1999.

[75] T. Samad, S. Harp, B. Wollenberg, B. Morton, L. Pires, S. Brignonne. Simulation of complex systems for the power industry with adaptive agents. *EPRI Technical Report TR112816*, 1999.

[76] B. Sarwar, G. Karypis, J. Konstan and J. Riedl. "Application of Dimensionality Reduction in Recommender System – A Case Study," *ACM E-Commerce 2000 Conference*, 2000.

[77] B. Sarwar, G. Karypis, J. Konstan and J. Riedl. "Analysis of Recommender Algorithms for E-Commerce," *ACM WebKDD 2000 Web Mining for E-Commerce Workshop,* 2000.

[78] C. Saunders. *Support Vector Machines User Manual*, March 23, 1998.

[79] C. Saunders, M. Stitson, and J. Weston. *Support Vector Machines Reference Manual*, July 10, 1998.

[80] B. Scholkopf. *Supervised Vector Learning*, R. Oldenbourg Verlag, 1997.

[81] B. Scholkopf. "SVMs – A pratical consequence of learning theory," *IEEE Intelligent Systems*, July/August 1998.

[82] Y. Seo and B. Zhang. "Learning User's Preferences by Analyzing Web-Browsing Behaviors," *Proceedings of the Agents 2000*, 2000.

[83] S. Shearing and P. Maes. "Representation and Ownership of Electronic Profiles," *CHI 2000 Workshop Proceedings Designing Interactive Systems for 1-to-1 E-commerce*, 2000.

[84] S. Solla, D. Schwartz, N. Tishby and E. Levin. "Supervised Learning: a Theoretical Framework," *Neural Information Processing Systems*, Edited by D. Touretzky, vol. 2, 1990.

[85] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*, Cambridge, MA: MIT Press, 1998.

[86] S. Thrun and K. Moller. "Active Exploration in Dynamic Environments," *Advances in Neural Information Processing System 4*, Edited by J. Moody, S. Hanson, and R. Lippmann, Morgan Kaufman Publishers, Inc., 531-538, 1994

[87] G. Urban, F. Sultan and W. Qualls. "Trust Based Marketing on the Internet," MIT Sloan W.P. 4035-98, 1998.

[88] V. Vapnik. *Estimation of Dependencies Based on Empirical Data*, Springer-Verlag, 1982.

[89] V. Vapnik. "Principles of Risk Minimization for Learning Theory," *Advances in Neural Information Processing System 4*, Edited by J. Moody, S. Hanson, and R. Lippmann, Morgan Kaufman Publishers, Inc., 831-838, 1994.

[90] V. Vapnik. *The Nature of Statistical Learning Theory*, Springer Verlag, 1995.

[91] V. Vapnik, S. Golowich, and A. Smola. *Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing*, 1997.

[92] A. Wasfi. "Collecting User Access Patterns for Building User Profiles and Collaborative Filtering," *Proceedings of the IUI 99*, Redondo Beach, CA, 1999.

[93] G. Wahba. "Spline Models for Observational Data," *Volume 59 of Regional Conference Series in Applied Mathematics*, SIAM Press, 1990.

[94] C. Watkins, *Learning from Delayed Rewards*, Ph.D. Thesis, Cambridge University, 1989.

[95] C. Watkins and P. Dayan. "Q-learning," *Machine Learning*, vol. 8, 279-292, 1992.

[96] A. Weigend, D. Rumelhart and B. Huberman. "Back-Propagation, Weight Elimination, and Time Series Prediction," *Proceedings of the Connectionist Models Summer School*, Morgan Kaufmann Publishers, Inc., 105-116, 1991.

[97] A. Weigend, D. Rumelhart and B. Huberman. "Generalization by Weight-Elimination with Application to Forecasting," *Advances in Neural*

Information Processing System 3, Edited by R. Lippmann, J. Moody, and D. Touretzky, Morgan Kaufman Publishers, Inc., 875-882, 1993.

[98] P. Werbos. “Approximate Dynamic Programming for Real-Time Control and Neural Modeling,” *Handbook of Intelligent Control*, Edited by D. White and D. Sofge, Van Nostrand Reinhold, 1992.

[99] P. Werbos. *The Roots of Backpropagation*, John Wiley & Sons, 1994.

[100] I. Witten and E. Frank. *Data Mining*, Morgan Kaufman Publishers, Inc., 2000.

[101] M. Wooldridge and N. Jennings. “Intelligent Agents: Theory and Practice,” *The Knowledge Engineering Review*, vol. 10, No. 2, pp. 115-152, 1995.

[102] L. Zadeh. "Fuzzy Sets,” *Information and Control*, vol. 8, pp. 338-353, 1965.

10 Terminology

This thesis assumes a basic understanding of computer engineering, machine learning, marketing, and electronic commerce and uses standard terminology without including definitions except where deemed necessary. The reader is referred to standard computer engineering, machine learning, optimization, marketing, and electronic commerce texts for those terms with which he/she is not familiar:

N. Cristianini and J. Shawe-Taylor, **An Introduction to Support Vector Machines**. Cambridge University Press, 2000.

D. Goldberg. **Genetic Algorithms in Search, Optimization and Machine Learning**. Addison-Wesley Publishing Co., 1989.

M. Hagan, H. Demuth, and M. Beale, **Neural Network Design**. PWS Publishing Company, 1995.

W. Hanson, **Principles of Internet Marketing**. South-Western College Publishing, 2000.

J. Holland, **Hidden Order**, Perseus Pr., 1996.

P. Kotler, **Marketing Management**. Prentice Hall, 1999.

W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, **Numerical Recipes in C**. Cambridge University Press, 1993.

11 Appendix A: Results for 125 Subjects

Appendix A contains the predictive performance (success rate) of the algorithms selected (Backpropagation neural networks, Fuzzy ARTMAP Neural Networks, Ensembles of Fuzzy ARTMAP Neural Networks, and Support Vector Machines) for the 125 subjects and the different training datasets. SVM is for Support Vector Machine, FAM is for Fuzzy ARTMAP, "Voting" is for an ensemble of 49 Fuzzy ARTMAP modules, BP is for Backpropagation, Pran is for Pure Random Generator, and Fran is for Frequency Random Generator.

Subject	Size of	Performance (Success Rate %)					
	Training File	SVM	FAM	Voting	BP	Pran	Fran
1	10	98.8%	98.8%	98.8%	98.8%	49.7%	49.7%
	25	94.4%	98.8%	98.8%	98.8%	50.6%	50.6%
	50	98.1%	98.8%	98.8%	98.8%	49.4%	45.1%
	100	97.2%	98.5%	98.5%	96.9%	50.9%	48.8%
	324	98.5%	94.4%	97.5%	97.8%	49.1%	42.9%
2	10	76.2%	77.8%	78.4%	78.1%	49.1%	72.8%
	25	76.2%	75.9%	75.0%	76.2%	53.1%	71.6%
	50	77.2%	76.2%	74.1%	76.5%	46.6%	63.9%
	100	76.2%	72.2%	79.6%	75.3%	52.2%	60.5%
	324	80.9%	81.2%	83.3%	84.0%	49.4%	62.0%
3	10	90.7%	90.7%	90.7%	90.7%	53.1%	90.7%
	25	90.7%	90.7%	90.7%	90.7%	50.0%	90.7%
	50	89.2%	90.1%	90.1%	86.4%	51.5%	85.2%
	100	90.7%	86.7%	90.4%	89.2%	47.2%	83.3%
	324	91.7%	87.0%	90.7%	92.0%	49.4%	81.2%
4	10	71.0%	66.0%	67.9%	67.6%	49.7%	49.7%
	25	75.6%	72.2%	71.9%	71.6%	47.5%	55.9%
	50	75.9%	72.5%	75.6%	74.7%	49.1%	58.6%
	100	73.5%	69.4%	71.3%	73.1%	53.1%	52.2%
	324	72.2%	71.9%	72.5%	70.7%	46.9%	57.4%
5	10	70.7%	70.7%	70.7%	70.7%	49.1%	70.7%
	25	70.7%	74.7%	75.0%	75.3%	48.8%	68.5%
	50	74.7%	78.1%	82.1%	80.6%	46.3%	62.0%
	100	79.9%	74.4%	83.6%	80.9%	44.8%	59.9%
	324	82.7%	73.8%	83.3%	82.1%	44.4%	55.6%
6	10	83.0%	73.1%	70.1%	64.8%	48.5%	53.4%
	25	77.2%	76.5%	76.9%	83.3%	47.5%	49.4%
	50	74.1%	75.9%	79.6%	78.7%	47.5%	49.4%
	100	85.5%	81.8%	85.8%	88.0%	48.5%	51.9%
	324	85.8%	79.0%	88.0%	88.9%	48.1%	54.6%
7	10	73.1%	69.4%	66.4%	67.0%	49.7%	57.4%
	25	74.4%	70.7%	72.5%	59.3%	52.5%	57.7%
	50	71.6%	74.7%	72.2%	73.1%	54.3%	58.3%
	100	74.7%	72.5%	73.1%	73.1%	43.5%	59.0%
	324	74.1%	70.4%	73.5%	75.9%	50.0%	56.8%
8	10	91.4%	91.4%	91.4%	91.4%	51.2%	48.5%
	25	88.3%	83.6%	83.6%	89.5%	47.2%	52.2%
	50	87.7%	91.7%	91.7%	89.2%	51.5%	48.8%
	100	90.1%	91.0%	90.4%	88.3%	50.3%	51.9%
	324	89.2%	83.0%	88.9%	84.9%	54.3%	51.5%
9	10	94.8%	83.3%	82.4%	82.4%	47.2%	86.1%
	25	95.7%	91.4%	93.5%	94.1%	58.6%	90.4%
	50	95.7%	96.0%	95.1%	95.7%	53.1%	91.0%
	100	94.1%	85.2%	84.9%	90.1%	46.3%	88.3%
	324	94.4%	89.8%	94.4%	92.3%	50.9%	88.6%
10	10	67.3%	60.5%	59.9%	67.6%	47.2%	49.4%
	25	72.8%	65.4%	67.6%	74.5%	49.7%	55.2%
	50	70.7%	65.4%	65.4%	73.5%	54.3%	48.1%
	100	70.4%	70.7%	71.9%	72.8%	51.9%	48.8%
	324	73.5%	72.5%	73.5%	72.2%	51.2%	51.9%

Subject	Size of	Performance (Success Rate %)					
	Training File	SVM	FAM	Voting	BP	Pran	Fran
11	10	90.7%	85.2%	83.6%	68.8%	50.3%	80.2%
	25	90.7%	90.7%	90.7%	81.5%	49.7%	87.7%
	50	90.7%	87.0%	88.6%	85.8%	49.4%	83.3%
	100	90.7%	86.1%	92.0%	88.0%	49.1%	79.6%
	324	90.7%	85.8%	89.8%	83.0%	48.1%	81.8%
12	10	88.3%	66.7%	66.7%	63.3%	50.3%	66.0%
	25	88.3%	78.1%	81.5%	75.9%	50.0%	73.1%
	50	88.3%	79.9%	84.3%	81.2%	51.9%	81.2%
	100	88.3%	84.0%	86.7%	87.0%	48.8%	77.8%
	324	88.3%	81.8%	84.6%	83.0%	53.4%	76.5%
13	10	90.4%	84.3%	87.0%	86.7%	50.3%	81.8%
	25	90.4%	81.8%	80.6%	82.4%	47.8%	81.8%
	50	90.4%	83.3%	83.0%	83.0%	50.3%	84.3%
	100	88.3%	87.0%	86.4%	87.7%	48.8%	81.5%
	324	90.4%	84.3%	86.7%	84.0%	50.6%	84.3%
14	10	76.5%	56.2%	63.3%	52.8%	46.9%	59.0%
	25	77.5%	75.3%	77.8%	67.6%	48.5%	57.4%
	50	79.9%	73.8%	78.7%	75.0%	48.5%	60.8%
	100	80.6%	76.2%	79.3%	77.2%	50.0%	61.1%
	324	81.2%	77.2%	81.2%	78.7%	51.5%	61.1%
15	10	89.8%	83.3%	84.9%	82.7%	50.6%	67.3%
	25	85.5%	85.2%	86.7%	75.6%	50.9%	72.5%
	50	90.7%	86.7%	90.1%	74.4%	51.2%	72.5%
	100	87.7%	82.7%	88.9%	87.7%	50.3%	73.1%
	324	84.6%	81.2%	85.8%	82.4%	48.5%	73.8%
16	10	74.7%	67.3%	66.0%	57.4%	46.0%	63.6%
	25	74.7%	68.8%	68.5%	68.5%	53.1%	67.6%
	50	74.7%	70.4%	72.8%	66.0%	54.3%	64.5%
	100	74.7%	63.6%	70.7%	74.7%	49.1%	66.0%
	324	74.7%	62.3%	69.1%	66.7%	51.2%	63.3%
17	10	42.9%	53.1%	53.7%	49.4%	46.0%	51.2%
	25	53.7%	51.9%	53.7%	48.5%	48.5%	54.9%
	50	60.5%	58.0%	53.1%	58.3%	49.7%	47.2%
	100	63.0%	56.8%	58.3%	54.0%	50.6%	48.1%
	324	61.1%	52.8%	58.6%	56.8%	48.5%	52.8%
18	10	95.7%	95.7%	95.7%	95.7%	47.8%	95.7%
	25	95.7%	92.3%	93.8%	91.0%	53.7%	92.9%
	50	95.7%	95.7%	95.7%	95.1%	48.1%	92.3%
	100	95.7%	92.6%	95.1%	91.7%	50.0%	92.9%
	324	95.7%	95.7%	95.4%	90.7%	46.9%	93.5%
19	10	76.5%	63.9%	65.4%	64.5%	53.1%	66.0%
	25	73.1%	60.8%	71.0%	69.1%	47.8%	66.4%
	50	76.5%	67.3%	71.6%	68.8%	51.2%	67.6%
	100	75.0%	66.4%	70.4%	71.6%	51.9%	63.9%
	324	76.5%	63.9%	65.1%	70.1%	47.8%	63.3%
20	10	77.8%	64.8%	69.4%	73.1%	49.7%	50.3%
	25	75.9%	72.2%	72.2%	69.1%	55.9%	67.3%
	50	77.8%	70.7%	70.7%	63.9%	53.4%	62.0%
	100	71.0%	63.9%	67.3%	67.6%	49.1%	60.8%
	324	75.9%	71.6%	72.2%	73.1%	52.2%	68.8%

Subject	Size of	Performance (Success Rate %)					
	Training File	SVM	FAM	Voting	BP	Pran	Fran
21	10	50.3%	45.1%	46.0%	47.8%	46.9%	51.9%
	25	44.1%	46.0%	42.9%	44.4%	52.2%	47.8%
	50	49.4%	46.3%	48.1%	49.7%	44.4%	50.0%
	100	49.4%	53.4%	50.6%	50.0%	50.9%	48.8%
	324	50.3%	50.3%	51.9%	46.0%	48.8%	52.8%
22	10	59.6%	49.7%	55.9%	54.3%	49.7%	48.5%
	25	59.6%	57.4%	58.6%	55.9%	49.1%	51.9%
	50	59.6%	57.7%	59.0%	56.5%	50.6%	51.9%
	100	59.6%	52.8%	60.2%	57.1%	53.4%	57.1%
	324	56.8%	51.2%	51.2%	50.6%	51.2%	47.5%
23	10	51.2%	53.7%	50.9%	53.7%	48.8%	48.5%
	25	50.9%	49.1%	52.2%	50.0%	45.7%	51.9%
	50	53.4%	50.3%	51.9%	51.9%	51.2%	50.9%
	100	48.1%	46.9%	52.3%	50.9%	51.9%	51.9%
	324	43.2%	43.8%	46.0%	50.0%	48.1%	46.6%
24	10	54.3%	54.0%	54.0%	54.0%	53.1%	55.9%
	25	57.4%	51.5%	54.6%	52.2%	49.4%	50.6%
	50	57.4%	52.2%	51.9%	55.6%	48.1%	50.3%
	100	53.1%	52.5%	53.1%	58.0%	49.4%	47.2%
	324	57.4%	50.9%	52.2%	55.2%	48.1%	50.6%
25	10	51.2%	54.3%	55.2%	50.0%	53.7%	49.7%
	25	51.2%	59.0%	56.8%	51.5%	50.9%	48.8%
	50	56.8%	54.3%	55.9%	48.1%	49.7%	51.5%
	100	48.5%	54.3%	52.8%	51.5%	51.9%	46.0%
	324	50.9%	53.4%	52.8%	54.3%	48.5%	48.5%
26	10	75.9%	72.5%	76.2%	73.5%	51.2%	68.5%
	25	75.9%	61.4%	68.5%	66.7%	55.9%	69.4%
	50	70.4%	64.8%	64.8%	66.4%	49.4%	66.0%
	100	75.9%	53.7%	68.5%	74.7%	46.0%	60.5%
	324	75.9%	55.2%	65.7%	71.0%	46.9%	67.0%
27	10	86.1%	76.9%	82.7%	80.2%	46.0%	78.4%
	25	86.1%	81.8%	86.1%	85.2%	46.6%	84.0%
	50	86.1%	84.9%	84.9%	84.9%	49.4%	84.0%
	100	86.1%	87.0%	87.7%	89.8%	51.9%	81.8%
	324	92.0%	91.0%	94.1%	91.0%	47.8%	77.5%
28	10	58.6%	58.0%	59.9%	60.8%	55.9%	55.2%
	25	58.6%	65.1%	66.0%	71.0%	49.4%	55.6%
	50	72.5%	65.1%	73.5%	76.5%	52.5%	54.9%
	100	70.7%	73.5%	77.8%	78.1%	50.0%	50.3%
	324	79.0%	74.1%	77.5%	75.0%	51.9%	55.2%
29	10	52.5%	63.6%	64.2%	51.2%	47.8%	51.2%
	25	62.0%	64.2%	63.3%	69.1%	54.9%	46.3%
	50	63.3%	62.7%	63.6%	58.0%	51.5%	49.7%
	100	68.8%	68.5%	70.7%	69.8%	48.8%	52.5%
	324	74.1%	66.0%	72.5%	73.1%	47.2%	49.7%
30	10	87.3%	81.2%	80.9%	60.8%	51.5%	50.3%
	25	82.4%	85.5%	88.3%	75.3%	57.4%	58.3%
	50	91.0%	92.3%	93.5%	88.3%	53.1%	57.1%
	100	90.4%	92.6%	92.9%	87.7%	56.2%	53.4%
	324	91.7%	86.4%	91.7%	88.0%	50.3%	57.1%

Subject	Size of	Performance (Success Rate %)					
	Training File	SVM	FAM	Voting	BP	Pran	Fran
31	10	79.0%	75.9%	76.9%	68.5%	49.4%	61.4%
	25	82.1%	78.7%	80.9%	76.5%	48.1%	62.7%
	50	82.1%	80.2%	82.1%	79.0%	42.9%	71.9%
	100	84.3%	83.6%	88.3%	76.9%	52.2%	70.1%
	324	81.5%	79.0%	81.5%	85.2%	51.9%	72.5%
32	10	84.9%	67.6%	72.2%	73.5%	54.6%	66.0%
	25	84.9%	71.3%	75.6%	75.3%	50.3%	68.8%
	50	84.9%	72.2%	79.6%	67.9%	45.1%	75.0%
	100	80.6%	74.4%	83.3%	74.1%	45.1%	69.8%
	324	85.5%	77.5%	83.0%	84.6%	49.7%	72.5%
33	10	77.5%	73.8%	73.8%	75.9%	51.5%	63.3%
	25	71.6%	78.1%	77.2%	75.3%	47.8%	54.9%
	50	80.6%	77.2%	81.2%	74.7%	50.3%	54.6%
	100	84.0%	82.1%	87.0%	75.3%	51.2%	58.3%
	324	85.5%	85.8%	85.8%	79.3%	47.2%	58.3%
34	10	49.7%	63.9%	63.9%	62.7%	55.6%	50.9%
	25	67.9%	71.3%	72.2%	68.5%	47.2%	47.8%
	50	63.3%	60.2%	63.6%	62.7%	50.6%	48.5%
	100	64.2%	63.6%	67.6%	71.9%	50.6%	48.8%
	324	65.7%	59.3%	68.2%	68.8%	53.1%	47.8%
35	10	60.2%	60.2%	61.4%	51.2%	52.5%	46.3%
	25	65.7%	70.7%	68.8%	71.9%	49.4%	49.7%
	50	80.6%	67.9%	83.3%	81.5%	47.5%	54.3%
	100	79.9%	74.4%	88.3%	81.5%	50.3%	50.0%
	324	80.2%	80.6%	87.7%	84.9%	47.5%	53.7%
36	10	88.9%	82.7%	83.0%	78.7%	48.8%	73.5%
	25	86.4%	81.2%	83.0%	83.3%	50.9%	76.2%
	50	86.1%	82.1%	85.8%	84.3%	51.2%	81.5%
	100	88.9%	84.0%	87.0%	85.2%	51.2%	79.9%
	324	88.6%	81.8%	85.5%	81.2%	48.5%	78.1%
37	10	75.0%	73.8%	72.2%	76.9%	51.9%	54.6%
	25	79.3%	73.5%	78.1%	78.7%	51.9%	51.5%
	50	73.1%	75.6%	75.3%	72.8%	55.2%	50.3%
	100	78.4%	73.1%	75.9%	77.8%	48.5%	52.8%
	324	79.6%	81.8%	85.2%	76.9%	50.0%	51.9%
38	10	96.0%	96.0%	96.0%	96.0%	50.6%	48.1%
	25	92.3%	96.0%	96.0%	96.0%	56.8%	52.5%
	50	96.0%	96.0%	96.0%	96.0%	51.2%	52.8%
	100	92.3%	92.3%	93.5%	87.3%	48.8%	46.0%
	324	95.1%	92.6%	94.4%	94.1%	54.0%	53.7%
39	10	94.1%	94.1%	94.1%	92.3%	49.1%	43.8%
	25	77.2%	80.6%	84.0%	69.8%	50.0%	49.7%
	50	88.6%	92.0%	91.7%	83.0%	49.1%	51.9%
	100	88.9%	88.3%	89.5%	87.3%	52.2%	50.6%
	324	92.9%	92.6%	92.6%	92.0%	46.0%	53.1%
40	10	72.5%	65.7%	68.5%	66.7%	50.9%	49.7%
	25	65.1%	70.7%	70.7%	72.2%	49.7%	57.4%
	50	61.4%	73.1%	73.8%	76.9%	44.4%	55.2%
	100	75.6%	73.8%	80.2%	75.9%	46.9%	51.9%
	324	81.5%	77.2%	76.5%	83.3%	46.3%	57.1%

Subject	Size of	Performance (Success Rate %)					
	Training File	SVM	FAM	Voting	BP	Pran	Fran
51	10	75.3%	71.9%	75.9%	65.1%	50.9%	65.4%
	25	76.2%	71.6%	77.5%	77.2%	54.9%	66.0%
	50	74.4%	74.1%	75.6%	73.5%	55.6%	63.6%
	100	79.9%	77.2%	80.9%	78.4%	48.8%	58.3%
	324	83.0%	79.6%	84.9%	85.5%	54.6%	65.4%
52	10	88.9%	82.1%	85.2%	85.5%	54.9%	80.6%
	25	88.9%	87.3%	90.4%	84.3%	48.8%	82.7%
	50	91.4%	90.4%	87.7%	90.4%	43.5%	84.9%
	100	88.0%	86.1%	88.6%	88.0%	54.9%	81.5%
	324	88.6%	88.6%	88.3%	89.2%	52.8%	82.4%
53	10	76.9%	74.1%	72.2%	75.6%	50.9%	50.6%
	25	77.2%	73.5%	74.1%	78.1%	52.5%	55.2%
	50	76.9%	70.7%	79.6%	74.7%	47.5%	50.3%
	100	70.7%	72.2%	76.5%	74.7%	45.1%	51.5%
	324	75.9%	66.0%	77.8%	75.0%	51.5%	54.0%
54	10	91.3%	82.7%	87.7%	83.3%	48.8%	73.8%
	25	90.4%	89.5%	89.8%	83.0%	45.1%	83.6%
	50	90.4%	90.1%	90.4%	86.4%	48.8%	84.3%
	100	88.9%	88.0%	88.6%	87.3%	42.9%	80.9%
	324	90.4%	88.6%	90.7%	88.6%	50.0%	83.3%
55	10	81.2%	69.1%	69.4%	61.4%	48.8%	54.9%
	25	81.2%	73.5%	72.2%	70.7%	51.5%	66.0%
	50	81.2%	73.8%	74.4%	78.4%	54.6%	73.1%
	100	81.8%	80.6%	78.7%	83.3%	50.3%	70.7%
	324	82.1%	78.7%	80.9%	78.4%	52.8%	68.2%
56	10	70.7%	70.7%	70.7%	70.7%	53.7%	63.9%
	25	74.4%	74.4%	72.8%	67.3%	44.4%	52.2%
	50	75.6%	82.7%	82.4%	75.6%	51.5%	57.1%
	100	81.8%	76.9%	84.0%	78.7%	46.9%	61.4%
	324	84.6%	83.0%	86.1%	83.0%	49.4%	59.0%
57	10	63.3%	59.9%	63.6%	63.3%	45.4%	50.0%
	25	67.3%	67.6%	65.7%	69.1%	54.9%	52.2%
	50	69.1%	70.4%	67.9%	73.8%	48.8%	45.4%
	100	67.9%	67.3%	69.4%	76.9%	51.5%	51.2%
	324	71.0%	69.1%	73.8%	70.1%	46.0%	50.0%
58	10	71.3%	75.0%	72.5%	67.0%	52.5%	63.9%
	25	71.3%	73.8%	73.8%	74.7%	52.2%	66.7%
	50	71.3%	73.1%	74.1%	71.3%	45.4%	64.8%
	100	73.1%	76.5%	77.2%	71.6%	50.6%	59.6%
	324	75.9%	77.8%	76.5%	76.9%	50.0%	61.1%
59	10	82.7%	79.3%	84.0%	77.5%	47.8%	60.2%
	25	89.2%	87.0%	89.8%	89.5%	49.4%	61.1%
	50	80.6%	77.5%	84.3%	75.0%	49.7%	59.3%
	100	76.9%	86.1%	86.7%	86.4%	50.3%	60.5%
	324	86.7%	82.4%	86.4%	84.6%	52.8%	62.0%
60	10	83.3%	84.6%	84.6%	72.8%	50.6%	51.9%
	25	84.0%	77.8%	78.1%	77.5%	56.5%	50.0%
	50	87.3%	84.3%	88.9%	85.5%	46.6%	57.1%
	100	83.6%	84.9%	88.3%	84.0%	51.2%	61.1%
	324	84.9%	85.2%	88.0%	85.5%	53.7%	56.2%

Subject	Size of	Performance (Success Rate %)					
	Training File	SVM	FAM	Voting	BP	Pran	Fran
61	10	88.9%	86.1%	85.8%	88.0%	50.3%	71.0%
	25	90.1%	88.6%	89.5%	88.3%	50.6%	78.7%
	50	88.0%	88.3%	89.5%	86.4%	47.5%	77.8%
	100	88.9%	88.3%	89.5%	87.7%	46.9%	75.0%
	324	88.6%	88.6%	88.3%	85.2%	46.3%	80.2%
62	10	96.0%	78.7%	90.1%	65.4%	51.5%	85.8%
	25	96.0%	83.3%	95.1%	91.4%	50.0%	88.3%
	50	96.0%	92.9%	94.1%	90.4%	50.6%	91.0%
	100	96.0%	91.0%	93.5%	92.3%	45.4%	91.4%
	324	96.0%	94.4%	96.0%	90.4%	52.2%	93.5%
63	10	70.4%	66.7%	67.3%	68.5%	50.0%	53.1%
	25	77.2%	75.6%	79.0%	77.2%	46.3%	51.2%
	50	81.2%	74.1%	84.0%	75.3%	51.5%	57.7%
	100	75.6%	80.6%	81.8%	79.9%	45.7%	56.2%
	324	79.6%	81.5%	83.6%	79.3%	49.4%	58.6%
64	10	60.8%	63.9%	63.3%	65.4%	43.2%	48.8%
	25	71.6%	63.3%	67.3%	72.8%	47.8%	52.8%
	50	74.4%	69.8%	75.3%	74.1%	52.2%	51.9%
	100	75.6%	66.7%	79.9%	75.3%	50.9%	49.7%
	324	76.2%	67.6%	76.5%	71.3%	48.1%	52.2%
65	10	81.2%	75.3%	80.2%	75.9%	54.0%	58.0%
	25	93.2%	82.7%	92.9%	93.2%	49.7%	56.8%
	50	92.9%	92.0%	93.2%	93.2%	47.8%	51.9%
	100	90.4%	84.3%	92.0%	92.0%	50.6%	59.3%
	324	87.3%	85.8%	92.0%	87.7%	47.2%	51.5%
66	10	80.9%	72.8%	73.8%	66.0%	50.6%	65.7%
	25	80.9%	79.3%	81.8%	71.9%	47.2%	71.6%
	50	82.7%	83.6%	83.0%	84.6%	54.3%	70.1%
	100	83.6%	86.4%	87.0%	85.5%	50.9%	68.8%
	324	91.7%	85.2%	92.0%	88.0%	54.6%	67.0%
67	10	92.6%	96.6%	93.8%	80.6%	45.1%	50.0%
	25	90.7%	82.4%	94.8%	86.1%	46.3%	50.6%
	50	96.0%	96.9%	96.9%	95.4%	48.5%	51.9%
	100	94.4%	92.3%	96.6%	95.4%	49.1%	48.5%
	324	96.0%	93.5%	95.4%	92.6%	50.9%	53.1%
68	10	71.6%	66.0%	64.8%	62.3%	47.8%	60.5%
	25	71.6%	60.2%	60.5%	66.7%	49.4%	59.9%
	50	71.6%	61.4%	61.1%	67.3%	50.0%	57.1%
	100	70.1%	58.3%	60.5%	57.4%	46.6%	55.6%
	324	69.8%	59.9%	68.8%	66.4%	50.6%	56.2%
69	10	84.9%	77.5%	77.5%	81.2%	50.3%	67.0%
	25	82.1%	84.3%	85.8%	81.8%	52.2%	77.2%
	50	85.8%	81.2%	87.3%	84.9%	48.5%	78.4%
	100	86.7%	87.3%	87.3%	92.9%	49.7%	72.8%
	324	90.1%	87.7%	87.7%	87.0%	50.9%	72.8%
70	10	79.9%	79.9%	84.9%	76.2%	53.7%	69.4%
	25	79.9%	79.0%	79.9%	78.7%	46.9%	68.8%
	50	79.9%	76.5%	79.3%	79.6%	49.1%	66.4%
	100	79.9%	79.6%	80.9%	79.6%	46.6%	69.1%
	324	81.8%	76.2%	81.8%	79.3%	47.8%	67.6%

Subject	Size of	Performance (Success Rate %)					
	Training File	SVM	FAM	Voting	BP	Pran	Fran
71	10	73.8%	71.9%	75.6%	71.9%	50.6%	65.7%
	25	76.2%	71.6%	76.9%	73.5%	49.4%	65.4%
	50	74.4%	74.1%	76.2%	70.4%	44.1%	59.9%
	100	79.9%	77.2%	81.8%	79.3%	52.2%	59.9%
	324	83.0%	79.6%	85.5%	83.0%	49.4%	60.8%
72	10	88.9%	82.1%	86.4%	77.2%	48.8%	79.6%
	25	88.9%	87.3%	88.6%	87.7%	50.3%	84.0%
	50	91.4%	90.4%	87.7%	87.7%	44.4%	86.4%
	100	88.0%	86.1%	88.6%	85.8%	55.9%	78.7%
	324	88.6%	88.6%	88.3%	88.9%	48.1%	79.9%
73	10	76.9%	74.1%	73.1%	74.4%	50.0%	47.5%
	25	77.2%	73.5%	75.0%	67.9%	47.2%	45.4%
	50	76.9%	70.7%	78.7%	75.9%	51.5%	48.1%
	100	70.7%	72.2%	75.6%	74.7%	48.5%	42.3%
	324	75.9%	66.0%	77.5%	77.5%	50.9%	51.2%
74	10	90.4%	82.7%	84.6%	80.2%	46.9%	74.7%
	25	90.4%	89.5%	89.5%	86.7%	54.9%	84.6%
	50	90.4%	90.1%	90.4%	90.7%	49.7%	85.5%
	100	88.9%	88.0%	88.3%	84.3%	48.5%	82.4%
	324	90.4%	88.6%	89.8%	88.3%	51.5%	83.0%
75	10	81.2%	69.1%	69.4%	51.2%	50.0%	59.3%
	25	81.2%	73.5%	73.8%	74.1%	51.9%	67.3%
	50	81.2%	73.8%	73.5%	78.4%	46.6%	65.7%
	100	81.8%	80.6%	78.4%	83.3%	54.0%	63.6%
	324	82.1%	78.7%	80.6%	78.7%	51.5%	68.5%
76	10	70.7%	70.7%	70.7%	70.7%	58.6%	63.9%
	25	74.4%	74.4%	73.5%	67.6%	48.8%	50.0%
	50	75.6%	82.7%	83.0%	75.6%	52.5%	53.7%
	100	81.8%	76.9%	84.3%	79.6%	49.7%	62.0%
	324	84.6%	83.0%	86.4%	78.7%	50.3%	58.6%
77	10	63.3%	59.9%	64.5%	67.9%	52.2%	47.8%
	25	67.3%	67.6%	65.7%	70.1%	46.3%	49.1%
	50	69.1%	70.4%	67.0%	70.1%	49.7%	48.5%
	100	67.9%	67.3%	69.4%	67.6%	48.5%	51.9%
	324	74.1%	69.1%	75.9%	69.1%	50.3%	49.1%
78	10	71.3%	75.0%	72.5%	78.4%	49.4%	63.0%
	25	71.3%	73.8%	74.7%	75.0%	56.2%	60.8%
	50	71.3%	73.1%	75.3%	72.5%	52.5%	64.2%
	100	73.1%	76.5%	75.9%	74.1%	53.1%	64.2%
	324	75.9%	77.8%	76.9%	77.5%	50.6%	58.0%
79	10	87.9%	79.3%	85.2%	81.2%	46.9%	60.5%
	25	89.2%	87.0%	89.5%	85.8%	46.0%	61.1%
	50	81.8%	77.5%	82.4%	79.3%	49.4%	60.8%
	100	83.3%	86.1%	88.0%	86.4%	47.2%	58.0%
	324	86.7%	82.4%	89.2%	85.5%	54.6%	63.9%
80	10	83.3%	84.6%	83.6%	64.2%	49.1%	47.2%
	25	84.0%	77.8%	78.7%	71.9%	50.6%	56.5%
	50	87.3%	84.3%	89.2%	84.9%	47.8%	56.8%
	100	83.6%	84.9%	88.0%	87.7%	48.8%	58.0%
	324	84.9%	85.2%	88.6%	85.5%	51.5%	56.8%

Subject	Size of	Performance (Success Rate %)					
	Training File	SVM	FAM	Voting	BP	Pran	Fran
81	10	88.9%	86.1%	86.1%	72.8%	52.5%	75.0%
	25	90.1%	88.6%	89.2%	83.3%	51.5%	74.1%
	50	88.0%	88.3%	90.1%	83.0%	50.9%	80.2%
	100	88.9%	88.3%	88.6%	88.6%	44.4%	77.5%
	324	88.6%	88.6%	88.9%	87.0%	46.0%	82.4%
82	10	96.0%	78.7%	86.7%	90.1%	50.6%	83.6%
	25	96.0%	83.3%	94.4%	92.0%	50.6%	88.3%
	50	96.0%	92.9%	94.4%	92.0%	49.7%	90.4%
	100	96.0%	91.0%	93.8%	90.4%	49.4%	90.7%
	324	96.0%	94.4%	96.0%	95.1%	52.8%	93.8%
83	10	70.4%	66.7%	68.5%	69.4%	48.8%	52.5%
	25	77.2%	75.6%	80.2%	75.6%	49.4%	54.9%
	50	81.2%	74.1%	81.8%	76.9%	51.9%	60.5%
	100	75.6%	80.6%	81.5%	79.9%	52.8%	64.5%
	324	79.6%	81.5%	81.8%	77.2%	50.9%	59.3%
84	10	60.8%	63.9%	63.3%	63.6%	49.7%	46.0%
	25	71.6%	63.3%	67.0%	68.2%	48.8%	46.0%
	50	74.4%	69.8%	75.0%	73.5%	50.9%	53.7%
	100	75.6%	66.7%	79.9%	68.8%	51.9%	47.2%
	324	76.2%	67.6%	75.6%	73.8%	51.9%	52.2%
85	10	81.2%	75.3%	80.2%	76.9%	46.3%	54.3%
	25	93.2%	82.7%	88.9%	93.2%	54.9%	54.9%
	50	92.9%	92.0%	93.2%	93.2%	48.1%	51.2%
	100	90.4%	84.3%	92.0%	91.0%	53.1%	58.6%
	324	87.3%	85.8%	92.0%	92.9%	51.9%	54.6%
86	10	80.9%	72.8%	70.7%	77.8%	52.5%	65.7%
	25	80.9%	79.3%	82.7%	69.8%	52.2%	68.5%
	50	82.7%	83.6%	83.3%	70.7%	52.2%	74.1%
	100	83.6%	86.4%	87.7%	85.5%	57.4%	71.6%
	324	91.7%	85.2%	92.6%	92.0%	47.5%	69.8%
87	10	70.3%	73.1%	72.2%	72.2%	48.1%	53.4%
	25	77.2%	66.4%	68.2%	72.8%	52.5%	46.3%
	50	74.1%	69.1%	78.1%	74.1%	55.2%	53.7%
	100	72.8%	72.2%	73.8%	75.3%	50.6%	48.1%
	324	73.8%	68.8%	78.1%	73.5%	49.1%	49.4%
88	10	78.4%	75.3%	76.5%	74.7%	49.1%	73.5%
	25	78.4%	75.9%	76.5%	76.2%	45.4%	78.1%
	50	78.4%	76.5%	77.8%	75.0%	52.5%	70.4%
	100	78.4%	71.6%	74.1%	77.5%	47.5%	64.2%
	324	78.4%	71.6%	73.5%	74.4%	53.4%	66.7%
89	10	77.5%	77.5%	77.5%	74.1%	49.7%	55.6%
	25	63.6%	62.0%	64.2%	62.3%	56.8%	55.9%
	50	70.4%	67.9%	67.6%	68.2%	50.9%	55.2%
	100	69.4%	67.9%	70.7%	76.5%	49.1%	53.4%
	324	73.8%	69.4%	75.9%	73.8%	54.6%	54.9%
90	10	82.1%	82.1%	81.8%	81.5%	48.8%	54.3%
	25	71.6%	76.2%	74.7%	78.4%	52.5%	49.7%
	50	74.1%	71.9%	74.1%	73.5%	47.2%	50.3%
	100	76.2%	75.6%	78.7%	76.5%	50.3%	48.8%
	324	75.6%	70.4%	75.3%	72.5%	48.1%	50.6%

Subject	Size of	Performance (Success Rate %)					
	Training File	SVM	FAM	Voting	BP	Pran	Fran
91	10	72.5%	67.6%	70.4%	64.2%	50.3%	43.8%
	25	75.9%	71.9%	72.8%	71.3%	52.8%	61.7%
	50	77.5%	78.4%	79.3%	76.9%	50.6%	61.4%
	100	80.2%	79.9%	83.3%	81.8%	51.5%	62.0%
	324	83.0%	79.6%	85.2%	78.4%	51.9%	60.2%
92	10	89.5%	83.3%	85.2%	79.3%	46.9%	82.1%
	25	89.5%	83.6%	85.2%	76.2%	49.1%	82.4%
	50	89.5%	83.0%	84.3%	79.9%	50.0%	83.6%
	100	85.2%	84.3%	84.3%	82.4%	53.7%	79.9%
	324	84.0%	86.4%	84.9%	86.1%	50.9%	79.3%
93	10	58.6%	51.9%	47.8%	47.5%	46.6%	50.3%
	25	58.6%	58.3%	60.2%	53.7%	42.6%	48.8%
	50	65.4%	66.7%	69.8%	70.4%	46.9%	50.6%
	100	70.4%	68.2%	76.2%	58.0%	46.0%	56.5%
	324	71.9%	63.6%	68.8%	74.7%	50.0%	50.9%
94	10	99.1%	99.1%	99.1%	99.1%	50.9%	49.4%
	25	94.1%	99.1%	99.1%	97.2%	45.7%	51.2%
	50	99.1%	99.1%	99.1%	99.1%	51.9%	47.8%
	100	98.8%	99.1%	99.1%	99.1%	50.3%	52.8%
	324	97.8%	96.9%	98.8%	98.8%	45.4%	52.5%
95	10	82.4%	82.4%	82.4%	80.2%	46.0%	58.0%
	25	75.3%	78.7%	74.7%	81.8%	49.1%	55.2%
	50	73.5%	82.4%	82.1%	76.9%	49.4%	52.2%
	100	73.8%	76.5%	75.9%	73.8%	50.0%	53.4%
	324	75.3%	75.3%	78.1%	82.4%	49.7%	48.8%
96	10	80.6%	82.4%	83.3%	77.2%	50.6%	63.3%
	25	85.5%	85.8%	89.5%	86.1%	48.1%	62.3%
	50	87.7%	88.0%	88.0%	85.2%	46.0%	64.5%
	100	85.8%	85.8%	88.3%	85.2%	56.2%	67.6%
	324	85.8%	80.9%	85.2%	84.9%	49.7%	60.8%
97	10	82.1%	77.8%	78.4%	50.0%	52.2%	50.6%
	25	84.3%	68.5%	81.8%	78.7%	51.9%	55.6%
	50	82.4%	86.4%	83.0%	81.8%	47.8%	55.9%
	100	81.5%	82.1%	81.8%	82.1%	47.8%	54.3%
	324	86.1%	83.0%	85.5%	80.6%	53.1%	51.5%
98	10	92.6%	92.6%	92.6%	92.6%	52.8%	50.6%
	25	92.3%	92.0%	92.0%	91.0%	49.1%	51.5%
	50	89.2%	92.0%	92.0%	86.4%	50.6%	58.3%
	100	85.2%	83.3%	89.2%	86.7%	44.8%	48.8%
	324	90.7%	84.6%	90.4%	90.7%	49.7%	50.9%
99	10	67.9%	71.3%	69.8%	60.8%	50.6%	54.0%
	25	65.1%	66.7%	66.4%	60.5%	50.6%	50.9%
	50	69.8%	65.7%	71.9%	67.6%	52.8%	56.8%
	100	65.1%	66.7%	69.4%	74.1%	49.1%	54.3%
	324	74.1%	69.4%	74.4%	71.3%	54.0%	55.6%
100	10	62.0%	67.6%	67.3%	66.4%	50.0%	59.6%
	25	62.0%	67.0%	65.1%	54.3%	48.5%	56.2%
	50	62.0%	66.0%	65.4%	67.9%	51.5%	53.4%
	100	67.0%	66.4%	69.8%	73.5%	48.1%	49.1%
	324	73.8%	69.4%	74.1%	75.9%	45.4%	51.2%

Subject	Size of	Performance (Success Rate %)					
	Training File	SVM	FAM	Voting	BP	Pran	Fran
101	10	94.8%	94.8%	94.8%	94.8%	50.0%	94.8%
	25	94.8%	90.7%	92.9%	93.2%	53.7%	88.3%
	50	94.8%	88.0%	91.4%	81.2%	58.0%	83.0%
	100	94.8%	89.8%	88.3%	85.8%	46.0%	88.3%
	324	93.8%	88.6%	92.0%	86.7%	53.7%	88.9%
102	10	92.6%	92.6%	92.6%	85.2%	45.1%	92.6%
	25	92.6%	80.6%	84.3%	83.0%	48.8%	81.8%
	50	92.6%	85.8%	88.3%	85.5%	50.0%	84.3%
	100	92.6%	90.1%	90.7%	86.1%	48.5%	86.1%
	324	92.6%	79.0%	87.7%	85.8%	53.7%	85.5%
103	10	60.2%	60.8%	60.5%	61.4%	47.8%	53.7%
	25	54.9%	52.2%	57.1%	63.9%	53.7%	52.8%
	50	54.9%	59.3%	59.3%	60.8%	48.1%	47.2%
	100	55.2%	57.1%	57.1%	60.8%	47.8%	52.5%
	324	63.3%	60.5%	63.9%	62.3%	51.5%	59.3%
104	10	53.7%	53.1%	53.4%	50.9%	48.8%	56.2%
	25	50.3%	52.5%	48.8%	49.1%	48.8%	47.2%
	50	50.9%	47.5%	47.5%	47.2%	51.9%	48.5%
	100	51.2%	48.8%	49.4%	49.7%	48.8%	43.8%
	324	53.7%	54.3%	56.2%	51.5%	49.1%	46.9%
105	10	85.5%	76.2%	75.3%	75.9%	47.5%	67.3%
	25	85.5%	74.4%	75.6%	65.7%	51.2%	70.7%
	50	85.5%	71.3%	80.6%	85.5%	49.1%	71.9%
	100	85.5%	71.9%	79.3%	82.4%	49.1%	74.1%
	324	85.2%	76.2%	81.8%	75.0%	53.7%	76.9%
106	10	66.4%	60.2%	62.7%	64.5%	54.6%	61.4%
	25	66.4%	56.2%	56.2%	58.3%	48.5%	54.9%
	50	64.5%	59.9%	62.7%	54.3%	52.2%	57.7%
	100	66.4%	54.9%	54.6%	56.2%	50.9%	53.1%
	324	63.0%	55.9%	56.5%	51.9%	50.6%	54.3%
107	10	63.6%	58.3%	59.9%	58.3%	50.0%	59.6%
	25	62.0%	54.9%	55.2%	50.0%	54.0%	52.5%
	50	63.6%	54.6%	54.6%	55.2%	49.4%	60.5%
	100	63.6%	55.6%	58.0%	63.3%	49.4%	58.0%
	324	63.6%	56.2%	54.0%	63.6%	55.9%	58.6%
108	10	58.0%	57.1%	56.2%	58.0%	51.9%	51.9%
	25	56.5%	51.5%	54.9%	44.1%	51.2%	49.7%
	50	57.1%	48.1%	53.1%	49.1%	54.3%	52.5%
	100	50.9%	47.8%	50.3%	53.1%	48.8%	54.6%
	324	57.1%	50.9%	49.1%	42.3%	48.1%	47.2%
109	10	72.2%	53.1%	53.4%	51.2%	55.2%	51.9%
	25	72.2%	53.1%	54.0%	45.7%	50.9%	55.9%
	50	72.2%	57.7%	56.2%	51.5%	48.5%	61.4%
	100	72.2%	56.2%	60.8%	56.8%	50.6%	58.3%
	324	72.2%	55.6%	58.6%	54.3%	50.6%	59.3%
110	10	55.0%	46.3%	43.2%	46.9%	54.6%	42.3%
	25	71.9%	56.2%	55.6%	51.9%	47.2%	56.2%
	50	71.9%	63.3%	64.5%	64.2%	48.5%	60.2%
	100	71.9%	63.6%	64.5%	72.5%	54.9%	61.7%
	324	71.9%	55.6%	59.0%	59.3%	50.3%	55.9%

Subject	Size of	Performance (Success Rate %)					
	Training File	SVM	FAM	Voting	BP	Pran	Fran
111	10	66.4%	56.8%	55.9%	59.3%	52.5%	56.8%
	25	66.4%	55.9%	53.1%	55.2%	47.8%	53.4%
	50	66.4%	50.6%	57.4%	54.6%	52.5%	54.9%
	100	65.7%	53.1%	55.2%	50.0%	46.6%	56.2%
	324	66.4%	56.8%	58.3%	55.6%	48.8%	56.2%
112	10	83.3%	74.4%	77.2%	76.5%	43.2%	59.3%
	25	83.3%	74.7%	78.4%	75.3%	50.6%	72.5%
	50	83.3%	77.5%	81.5%	71.3%	51.9%	74.7%
	100	83.3%	76.5%	81.5%	77.2%	55.6%	70.1%
	324	83.3%	78.4%	78.1%	75.9%	50.9%	68.8%
113	10	93.8%	88.3%	88.0%	83.3%	52.8%	88.6%
	25	93.8%	93.8%	93.8%	91.4%	54.3%	88.9%
	50	92.3%	93.8%	93.8%	88.0%	52.2%	87.7%
	100	93.8%	85.8%	89.8%	88.6%	48.5%	88.3%
	324	94.1%	94.1%	91.0%	86.1%	46.9%	87.0%
114	10	61.7%	71.3%	73.8%	55.6%	51.9%	54.6%
	25	75.3%	75.3%	81.2%	74.1%	50.0%	63.3%
	50	80.6%	78.7%	85.2%	76.5%	49.7%	63.0%
	100	80.6%	80.6%	86.4%	81.5%	45.7%	59.0%
	324	86.7%	83.0%	90.1%	83.3%	51.9%	60.8%
115	10	94.8%	76.5%	83.0%	73.1%	51.9%	75.6%
	25	94.8%	89.5%	96.0%	88.6%	49.4%	88.6%
	50	94.8%	90.4%	95.1%	93.5%	47.8%	87.7%
	100	94.8%	93.5%	93.8%	93.8%	48.1%	88.3%
	324	94.8%	91.7%	92.9%	95.1%	54.3%	90.4%
116	10	61.1%	65.4%	66.0%	69.4%	49.4%	46.6%
	25	66.7%	63.6%	71.6%	70.1%	53.1%	51.2%
	50	74.4%	66.0%	78.7%	77.5%	47.2%	47.5%
	100	79.0%	71.9%	80.6%	80.2%	56.2%	53.7%
	324	78.7%	70.4%	76.5%	79.0%	49.7%	51.9%
117	10	94.1%	94.1%	94.1%	94.1%	52.2%	94.1%
	25	94.1%	94.1%	94.1%	94.1%	46.0%	94.1%
	50	94.1%	94.1%	94.1%	90.4%	49.4%	91.7%
	100	94.1%	93.8%	93.8%	92.3%	50.3%	91.4%
	324	94.8%	89.8%	94.4%	93.5%	49.4%	88.9%
118	10	73.8%	86.1%	76.9%	71.9%	51.2%	67.0%
	25	80.2%	87.0%	83.3%	77.5%	47.8%	69.4%
	50	78.4%	81.5%	83.3%	81.5%	49.7%	67.0%
	100	79.9%	79.9%	83.6%	82.7%	43.8%	63.3%
	324	87.7%	83.3%	89.8%	85.8%	48.8%	62.7%
119	10	44.1%	49.4%	52.5%	58.6%	49.4%	46.0%
	25	60.2%	56.8%	59.0%	56.8%	47.5%	50.9%
	50	62.7%	60.8%	61.7%	56.8%	50.6%	54.6%
	100	65.4%	66.7%	72.2%	64.5%	48.8%	49.7%
	324	67.0%	63.6%	70.1%	76.2%	51.2%	50.3%
120	10	58.3%	51.5%	53.7%	48.1%	50.6%	48.8%
	25	50.0%	52.8%	55.9%	44.1%	47.2%	50.3%
	50	57.4%	66.4%	55.9%	60.2%	52.5%	53.7%
	100	64.8%	63.0%	62.7%	75.3%	47.2%	51.5%
	324	68.2%	57.7%	58.3%	74.4%	53.4%	47.2%

Subject	Size of Training File	Performance (Success Rate %)					
		SVM	FAM	Voting	BP	Pran	Fran
121	**10**	88.9%	82.1%	79.9%	61.7%	44.8%	82.4%
	25	88.9%	87.0%	87.7%	88.3%	57.4%	84.9%
	50	88.9%	89.2%	88.6%	86.1%	55.6%	83.0%
	100	88.9%	81.2%	85.8%	85.2%	47.8%	79.0%
	324	89.2%	81.5%	87.0%	83.3%	47.8%	79.0%
122	**10**	90.4%	90.4%	90.4%	90.4%	48.1%	50.6%
	25	81.5%	89.2%	85.2%	81.2%	48.8%	46.3%
	50	87.3%	86.7%	86.7%	83.3%	49.1%	52.5%
	100	84.3%	80.9%	87.3%	79.3%	56.2%	48.8%
	324	87.7%	84.0%	89.5%	84.9%	56.8%	55.2%
123	**10**	80.6%	76.5%	79.9%	78.4%	49.4%	76.2%
	25	80.6%	78.4%	80.6%	74.7%	48.8%	73.5%
	50	82.4%	79.6%	81.5%	77.5%	46.3%	67.0%
	100	78.7%	78.1%	83.0%	78.4%	50.9%	61.7%
	324	81.8%	79.6%	80.6%	83.3%	53.1%	66.0%
124	**10**	85.2%	82.4%	82.1%	75.3%	53.4%	79.6%
	25	90.1%	82.1%	84.6%	72.2%	50.9%	78.1%
	50	90.1%	82.4%	86.1%	84.0%	46.9%	84.6%
	100	90.1%	88.3%	87.7%	88.6%	48.8%	82.4%
	324	90.1%	79.0%	85.5%	86.1%	50.3%	78.4%
125	**10**	81.5%	70.7%	71.3%	65.7%	53.4%	63.0%
	25	81.8%	77.2%	81.8%	67.9%	50.9%	68.5%
	50	79.9%	78.4%	79.6%	72.2%	51.5%	67.0%
	100	81.5%	74.4%	76.9%	75.6%	50.9%	65.4%
	324	82.1%	70.1%	75.6%	68.5%	50.9%	70.1%