THESIS PROPOSAL

USE OF AN MLP AND RBF NEURAL NETWORK TO DETECT IMPAIRED CHEATING ON THE COMPUTERIZED CERAD WORD LIST MEMORY TEST

Timothy Chan

January 13, 2008

I have read the attached thesis proposal and, in my opinion, it proposes work which is adequate in depth and scope to serve as the culminating experience for the Master's Degree in Computer Science. I would agree to chair this committee or serve thereon.

Date	Dr. Amar Raheja, Thesis Advisor
Date	Dr. Chung Lee, Committee Member
Date	Dr. Fang Tang, Committee Member

Contents

List of Figures

Chapter 1

Introduction

1.1 About the test

1.1.1 CERAD

The Consortium to Establish a Registry for Alzheimers Disease (CERAD) was established in 1986 by a grant from the National Institute on Aging (NIA), to standardize procedures for the evaluation and diagnosis of patients with Alzheimers disease (AD). Patients and nondemented control subjects were recruited from 24 NIA-sponsored Alzheimers Disease Research Centers and other university programs in the US. Using standardized diagnostic criteria and assessment instruments, CERAD subjects were examined at entry and annually thereafter, to observe the natural progression of AD.

The major standardized instruments developed by CERAD (CERAD battery) are now used by many AD research centers in the US and abroad, by physicians in clinical practice, and in population-based surveys. They have been translated into Bulgarian, Chinese, Dutch, Finnish, French, German, Italian, Japanese, Korean, Portuguese, and Spanish. Approximately 75 papers describing CERAD findings have been published in English-speaking scientific medical journals.

The CERAD battery includes demographic data on subject and informant, clinical history and examinations, extensive neuro-psychological exams, laboratory and imaging studies, and neuropathological studies. Of particular interest is the *Word List Memory* test.

The CERAD *Word List Memory* test has been recommended as a brief, efficient, and sensitive memory measure that can be used with a range of difficult patients (psychiatric patients vs. controls) [1]. It has also been showed to be one of the more sensitive tests in detecting Mild Cognitive Impairment (MCI) [2] and distinguishing early Alzheimer's disease from cognitively normal elderly control subjects [3].

1.1.2 Why detect MCI?

Patients with MCI are not disabled and usually have a single deficit – most often memory loss. MCI is a syndrome that is often due to AD. The reported rates of annual conversion from MCI to dementia due to AD range from 6-20%. Other studies have shown, as well, that patients with MCI are more likely to progress to AD than age-matched controls [4]. Since current therapies for AD can partially delay disease progression by up to 54%, detecting AD patients during the MCI stage slows their decline while patient function is still relatively preserved [5]. Early detection also allows patients and their families to more effectively plan for their future.

Potential adverse affects of screening are as follows: risk of depression and anxiety, and possible labeling effects[1][6].

[1]Once a diagnosis of dementia is given, the patient will not qualify for long-term care insurance or acceptance into a continuous care retirement community.

1.1.3 Structure of the test

To briefly describe the test, the subject is read a list of 10 words and asked to repeat each word and remember it. When finished, the subject is immediately asked to repeat as many of the 10 words as possible. The subject receives three such trials to learn the 10 word list (these trials are called the *immediate recall* trials).

A distractor task is given for 2-5 minutes to permit transfer of the information into episodic memory, which is processed in the hippocampus. The distractor task consist of 12 sets of comparison of the judged similarities among three animals at a time. The subject is then asked to freely recall as much of the 10 word list as they can (this is called the *delayed recall* trial).

1.2 Telephone testing

Testing the subject with the CERAD wordlist (CWL) over the telephone facilitates large scale screening, improves access and lowers cost. Other cognitive tests that have been developed for telephone administration include: the Brief Screen for Cognitive Impairment (BSCI) and the Telephone Interview of Cognitive Status (TICS). Research has demonstrated that psychological data obtained over the telephone are as reliable and valid as those obtained through face-to-face interaction [7, 8, 9].

1.2.1 BSCI

The Brief Screen for Cognitive Impairment (BSCI) consists of three questions administered by telephone (delayed recall, frequency of help with planning trips for errands, and frequency of help remembering to take medications). It has been asserted that the BSCI discriminates between demented patients and normal controls as well as two standard tests of dementia(the Mini Mental State Exam [MMSE], and

the Alzheimer's Disease Assessment Scale, [ADAS]), and may be a considered a valid screen for dementia [10].

1.2.2 TICS

The Telephone Interview for Cognitive Status (TICS) is an 11 item screening test and takes less than 10 minutes to administer and score. It has been shown to correlate well with the MMSE for post-stroke patients [11].

1.2.3 EMST

The Enhanced Mental Skills Test (EMST), uses the MCI Screen[2], a modification of the CWL, to detect MCI and mild dementia[12]. Screening tests are not needed to detect moderate or severe dementia. The MCI Screen analyzes the pattern of words recalled across the four CWL trials to give the highest accuracy for detecting MCI in the published literature[13]. The MCI Screen is 99% accurate in discriminating normal vs. mild dementia and 97% accurate in discriminating normal vs. MCI [13].

1.3 Cheating

The EMST is one of several tools used by the Long Term Care (LTC) insurance industry to screen applicants for MCI or dementia, which if detected, would result in denial of a policy. A recent industry study of telephone testing to screen 20,000 LTC applicants, found that 20% of them cheated to get a policy. Issuing an LTC policy to an impaired individual results in claims amounting to between $125,000 and $200,000 per such individual. Half of the LTC insurance companies went bankrupt in 2004 as a result of such errors.

[2]I am currently employed at the company that developed the MCI Screen.

1.4 Cheating detection

Currently, there are no systematic ways of evaluating whether an applicant taking the telephone test cheated. Occasionally it will be obvious to the tester, but the number of such applicants detected is negligible.

The insurance industry, needs to distinguish between normal applicants who cheat, and impaired cheaters. Given that approximately 1 million telephone tests are given annually, approximately 200,000 applicants cheat. Of these 200,000 cheaters, only 6% are impaired. Thus about 1.2% of 1 million applicants who take the telephone test are impaired and cheat to get a policy. Finding such individuals is analogous to finding a needle in a haystack.

We will employ the use of a Radial Basis Function Neural Network (RBF NN) to detect impaired cheaters. We will also compare our results with those obtained using logistic regression on the same data set.

Other applications of this research include a better understanding of the patterns of recall among normal individuals. We also hypothesize that the strategies used by impaired cheaters differ from those used by normal cheaters, and that RBF NN can detect such strategic differences.

Chapter 2

Literature Survey

2.1 The artificial neural network

The brain is often conceptualized to be a highly complex, nonlinear, and parallel computer.

Haykin[14] has said, a neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

1. Knowledge is acquired by the network through a learning process

2. Interneuron connection strengths known as synaptic weights are used to store the knowledge

According to Haykin[14], the benefits of neural networks include the following:

1. *Nonlinearity.* A neuron is basically a nonlinear device. As such, a neural network made up of an interconnection of neurons, is also nonlinear. This nonlinear property is highly important in cases where the underlying physical mechanism responsible for the generation of an input signal is inherently nonlinear.

2. *Input–Output Mapping.* A popular paradigm of learning, called supervised learning, involves the modification of synaptic weights of a neural network by applying a set of labeled training samples or task examples. Each example consists of a unique input signal and the corresponding desired response. The network is presented an example picked at random from the set, and the synaptic weights of the network are modified so as to minimize the difference between the desired response and the actual response of the network produced by the input signal in accordance with an appropriate statistical criteria. The training of the network is repeated for many examples in the set until the network reaches a steady state. The previously applied training examples may be reapplied during the training session in a different order. Thus the network learns from the examples by constructing an input–output mapping for the problem at hand.

3. *Evidential Response.* In the context of pattern classification, a neural network can be designed to provide information not only about which particular pattern to select, but also about the confidence in the decision made. This latter information may be used to reject ambiguous patterns, should they arise, and thereby improve the classification performance of the network.

The fundamental difference between a system that learns and one that merely memorizes is that the learning system generalizes to unseen examples[15].

An important design principle is that the network should only be as large as absolutely necessary. According to *The handbook of brain theory and neural networks* put out by MIT Press[15]:

> "Just as a polynomial of too high a degree is not useful for curve fitting, a network that is too large will fail to generalize well, and will re-

quire longer training times. Smaller networks, with fewer free parameters, enforce a smoothness constraint on the function found. For best performance, it is therefore desirable to find the smallest network that will 'fit' the training data."

2.2 Models of a neuron

A neuron is an information-processing unit that is fundamental to the operation of a neural network. We may identify three basic elements of a neuron model, as described here and can be see in [Fig. 2.1]:

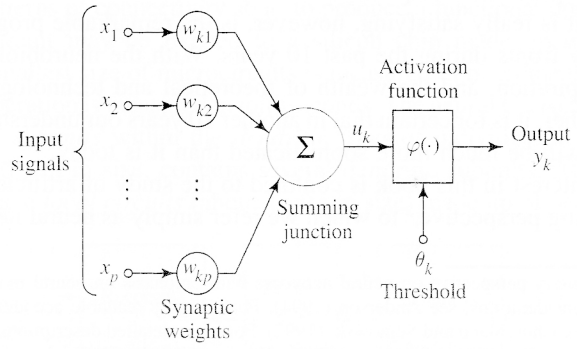

Figure 2.1 A nonlinear model of a neuron.

1. A set of synapses or connecting links, each of which is characterized by a weight or strength of its own.

2. An adder for summing the input signals, weighted by the respective synapses of the neuron.

3. An activation function for limiting the amplitude of the output of a neuron. The activation function is also referred to in the literature as a squashing function

in that it squashes (limits) the permissible amplitude range of the output signal to some finite value. Typically, the normalized amplitude range of the output of a neuron is written as a closed unit interval [0, 1] or alternatively [-1, 1].

2.3 The radial basis function neural network

The construction of a Radial Basis Function (RBF) network in its most basic form involves three entirely different layers. Each layer is comprised of a varying number of neurons. The input layer is made up of source neurons (sensory units). The output layer supplies the response of the network to the activation patterns applied to the input layer. In between the inputs and outputs there is a layer of processing units, called hidden units, which each implement a radial basis function. The transformation from the input space to the hidden-unit space is nonlinear, whereas the transformation from the hidden-unit space to the output space is linear. When a standard RBF network is used to perform a complex pattern classification task, it nonlinearly maps the input space onto a high dimensional space.

Finding the RBF weights is called network training. If we have a set of input-output pairs, called a training set, we optimize the network parameters in order to fit the network outputs to the given inputs. The fit is evaluated by means of a cost function, such as the mean squared error. After training, the RBF network can be used with data whose underlying statistics are similar to those of the training set.

If we take the approach of viewing the design of a neural network as a curve-fitting (approximation) problem in high dimensional space, then learning (training) is equivalent to finding a surface in multidimensional space that provides a best fit to the training data.

2.4 The difference between MLP and RBF

Another widely used neural network design is the multilayer perceptron (MLP); it is very similar to the RBF, thus warranting a discussion of how they are different. The MLP can be viewed as dividing the input space into regions bounded by hyperplanes, one for the thresholded output of each neuron of the output layer. Radial basis function networks describes an alternative approach to decomposition of a pattern space into regions, describing the clusters of data points in the space as if they were generated according to an underlying probability density function. Thus the multilayer perceptron method concentrates on class boundaries, while the radial basis function approach focuses on regions where the data density are highest, constructing global approximations to functions using combinations of basis functions centered around weight vectors. A succinct diagram [Fig. 2.2] describing the difference between a multilayer perception and radial basis function has been provided by Lowe [15]. We see in this figure an obvious clustering of data into 3 groups.

There are two ways of segregating these data. The first is to segregate them in polygonal cells. The straight lines in the figure illustrate this decomposition of the pattern space into regions, as would be obtained by a simple multilayer perceptron, where the lines represent the class boundaries. The second way of segregating these data is to describe the clusters of data themselves as if they were generated according to an underlying probability density function, modeled in the figure by elliptical distributions. Thus one method concentrates on class boundaries and the other focuses on regions where the data densities are highest.

Many types of nonlinear transfer functions, $\phi_j(\dots)$, may be used in an RBF NN. The output classes are biased by the final layer weights, $\{\lambda_{ij}\}$, connecting the jth hidden node to the kth output node [Fig. 2.3]. The input and hidden layers

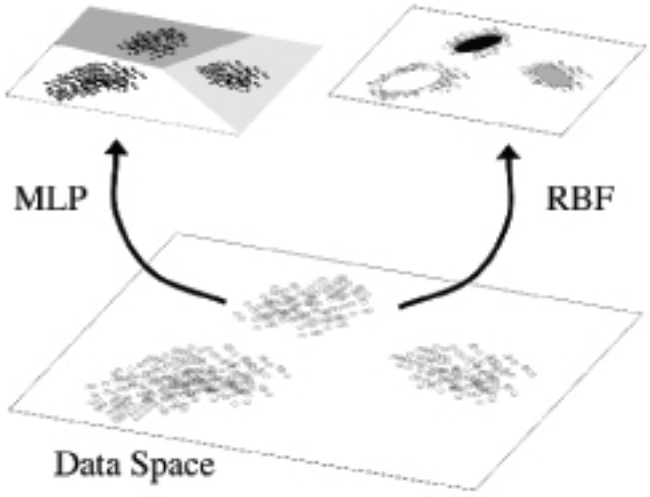

Figure 2.2 Dissection of pattern space by clusters and hyperplanes. The multilayer perceptron (MLP) exploits the logistic nonlinearity to create combinations of hyperplanes to dissect pattern space into separable regions. The radial basis function (RBF) dissects pattern space by modeling clusters of data directly and so is more concerned with data distributions.

are connected by weights, $\{\mu_{ij}\}$, which approximate the centers of clusters or the distribution of the input node.

The following is the mathematical embodiment of the RBF. The kth component of the output vector y_p corresponding to the pth input pattern x_p is expressed as

$$[y(x_p)]_k = \sum_{j=0}^{h} \lambda_{jk} \phi_j \left(\|x_p - \mu_j\|; \sum_j \right) \tag{2.1}$$

where $\phi_j(\ldots)$ denotes the nonlinear transfer function of hidden node j, ($\phi_0(\ldots) \equiv 1$ is the bias node), and the possible dependence on a smoothing matrix is left explicit. The most common example of the smoothing factor is in the use of a general Gaussian transfer fuction, i.e., $\phi(z) \approx exp - [z^T \sum^{-1} z]$. As can be seen from [Eq. 2.1], the main difference from a multilayer perceptron is that the output of the hidden node j, h_j is given as a radial function of the distance between each pattern vector and each hidden node weight vector (or prototype), $h_j^{RBF} = \phi_j(\|x_p - \mu_j\|)$, rather than as a

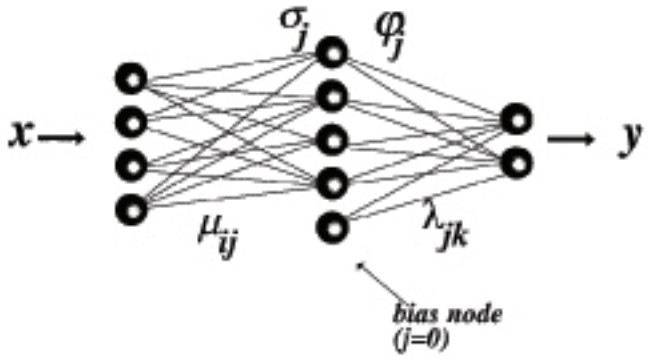

Figure 2.3 The basic radial basis function structure. x and y describe inputs and outputs, respectively. Layers are represented (not explicitly labeled) from left to right as i, j, and k. A nonlinear basis function $\phi_j(\dots)$ is centered on each hidden node weight vector μ, which also has a possible adaptable "Range of influence" σ_j. The output of the hidden node j, h_j is given as a radial function of the distance between each pattern vector and each hidden node weight vector. This is the main difference from a multiplayer perceptron. The network outputs are evaluated by a traditional scalar product between the vector of hidden node outputs and the weight vector attached to the output node k.

scalar product, $h_j^{MLP} = \phi_j(x \cdot \mu_j)$.

The original formulation of the RBF network was developed in order to produce a deterministic mapping of data by exploiting links with traditional function approximation. This approach attempted to introduce the notion that the training of neural networks could be described as curve fitting, hence,"generalization" consequently has a natural interpretation as interpolation along this fitting surface.

Lowe[15] gives a very clear and understandable example of this approach. Assume that we have a set of input/output pairs of input/target patterns representing data from an unknown but smooth surface in $\mathbb{R}^n \times \mathbb{R}^c$. For pedagogical purposes, lets use a simple example of a set of (x, y) pairs generated according to $y = x^2$. In this approach, the problem is to choose a function $y : \mathbb{R}^n \to \mathbb{R}^c$, which satisfies the interpolation conditions $y(x_p) = t_p, p = 1, 2, \dots, P$. This is a strict interpolation, in

which the function is constrained to pass through *all* the known data points. The strategy in interpolation theory was to construct a linear function space spanned by a set of nonorthogonal basis functions that depended on the positions of the known data points.

In the case of the simple $y = x^2$ example mentioned above, the inputs are x values, the targets are specific y values, and the RBF network is constructed so as to produce a fitting surface to the parabola $y = x^2$, which is a surface in $\mathbb{R}^1 \times \mathbb{R}^1 = \mathbb{R}^2$. Note that this is curve fitting to the *parabola* in the product space, not the data samples themselves. This parabola may be interpreted as the *generator* of the observed data, as locations on the parabola "produce" or "generate" the (x, y) input/output pairs.

As long as the generator of the data is smooth and nonlinear, we should be able to arbitrarily closely approximate it with an RBF network. This explains why networks can be successful in predicting deterministically chaotic time series: although the time series themselves may exhibit apparent randomness, the underlying map that produces the samples is itself usually very smooth.

The aim of the network is to approximate the underlying structure that generated the observed data, rather than the data itself. This is also how a mutlilayer perceptron operates. However, the RBF was introduced to make this link with curve fitting and interpolation explicit.

The strength and utility of the RBF derive from its simplicity and from a close relationship with other areas of signal and pattern processing and other neural network architectures.

In summary, the RBF network approach involves the expansion or pre-processing of input vectors into a high-dimensinal space. This attempts to exploit a theorem of Cover [16] which implies that a classification problem cast in a high-dimensional space is more likely to be linearly separable than would be the case in a low-dimensial

space.

2.5 Which is better?

From a historical perspective, MLP's became widely used in 1986 when a general backpropagation algorithm for the MLP was introduced by Rummelhart et al. [17]. RBF's were first introduced by Broomhead & Lowe in 1988[18]. Though one preceeds the other, there is no definitive answer to the question of which is better.

In the field of speaker recognition, many researchers have reported superior performance in recognition of the RBF over MLP networks[19, 20, 21]. Finan[22] confirms this. He also states "the overall superiority of performance reported in general for RBF networks over MLPs would appear to be due to the reduced sensitivity of the former to a poor training set when compared to the performance of an MLP for the same training set. When both networks are presented with their 'best' training sets, however, the RBF network still significantly out-performs the MLP."

The opposite was found to be true in the use of MLP and RBF for solutions to laser assisted paint stripping process of hybrid epoxy-polyester coatings on aluminum substrates[23]. In this application, MLPs were determined to be "more reliable and effective in modeling experimental results."

Mitra[24] showed that their RBF system trained itself much faster than their MLP based system used to classify Lidar data. He also found that though the RBF trained faster, it was found to have lower prediction accuracy than the MLP based system.

2.6 Applications

There have been many other real world applications of radial basis function neural networks, such as speech recognition [25], facial recognition [26], medical diagnosis [27], radar point-source location [28] and handwritten character recognition [29].

Because the patterns of recall used by impaired individuals differ from those of normals [30], their cheating strategies may also differ. My research will examine if RBF NN can detect such pattern differences.

2.7 Comparison with logistic regression

We will compare the performance of the RBF NN results with those obtained using logistic regression on the same data. Based on Sargent's meta-analysis of 28 studies [31], it is not possible to predict *a priori* whether RBF NN or logistic regression will do better.

Logistic regression has the advantage of being much more straightforward and computationally quicker and easier to replicate [32], while RBF NN extract nonlinear relations when they improve classification [31]. A drawback of neural networks is that the parameters in the model are not directly interpretable [33].

2.8 Certainty factors

2.8.1 Introduction

Certainty factors were first introduced by Buchanan and Shortliffe in their development of a medical diagnostic system, MYCIN, for Stanford University [34]. Certainty factors are based upon heuristic measurements of belief and disbelief of facts and are

loosely related to probability theory [35].

An illustrative example is as follows: suppose there is some fact X that is asserted to be true. If the certainty factor for this assertion is 1.0 then it is believed to be 100% true. If the certainty factor of this assertion is 0.0, then it is believed to be 0% true. The allowable range of certainty factor values is between 0.0 and 1.0, inclusively. This value can be interpreted to mean percent belief that a fact is true.

Wedding et al. [36] made it a point to explain that "certainty factors should not be confused with probabilistic measures, since there is only a small correlation between the two." For example, a man may assert that he is 90% certain that he will pick up milk on the way home from work. This does not mean the same thing as there is a 90% probability that the man will purchase the milk. In the first case, it has the meaning that the man is reasonably confident that he will buy the milk. The second meaning implies that all things being equal, the man will buy milk exactly 9 times out of 10.

2.8.2 Reliability measures of RBF's

RBF neural networks will give good results for input that it has seen before. It, however, will give poor results for unfamiliar input data. As stated by Leonard et al. [37]:

> "Because network models are not based on physical theory and contain nonlinearities, their predictions are suspect when extrapolating beyond the range of the original training data."

Therefore, to get the most accurate results from the RBF neural network it is essential to identify good and bad answers. Once identified, the good results are maintained while the bad results are discarded. Having the RBF neural network

itself indicate good and bad results is the most direct approach to this problem, according to Wedding et al. [36].

As previously mentioned, unlike MLP's, RBF's use a distance metric in the input space to determine the hidden layer activations. As a result, the contours of constant activation of the hidden layer are hyper-spheres instead of hyper-planes, [Fig. 2.2]. The contours are finite in length and form closed regions of significant activation, as opposed to MLPs where the contours are infinite in length and form semi-infinite regions of significant activation; this feature was exploited by Leonard et al. in their VI-net, an RBF based neural network that reports its own reliability [37].

Using RBF neural networks to generate a reliability measure is not an entirely novel concept. Lee [29] used this principle in a handwritten digit recognition program and achieved good results. Lee used as a measure of confidence the numeric difference between the two output nodes with the highest values. The disadvantage of this approach is that delta can, in some instances, be large even though the actual network familiarity with the input data is low [35].

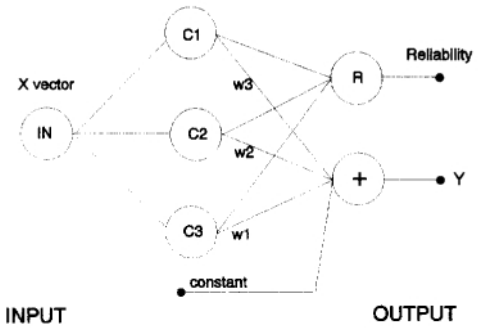

Figure 2.4 RBF neural network modified to generate a reliability measure along with normal output. Reliability (node R) and normal output are computed in parallel.

Leonard et al. [37] approached the problem differently. They suggested computing the reliability from the output of the RBF network's basis functions, i.e. the output

from the hidden layer. They proposed adding additional nodes to the output layer, see [Fig. 2.3]. This new network, shown in [Fig. 2.4], computes both output and reliability in parallel. Based on this numeric confidence measure, the user then has a choice to reject or accept the output. They explored two methods of measuring the reliability of prediction: the maximum activation value function and the Parzen estimator. They concluded the Parzen method was superior because it took into account the distribution of the training data. One flaw in this method is that when the dimensions of the input vector is large, the Parzen window approach is no longer capable of generating meaningful reliability measures [35].

2.8.3 Using certainty factors with RBF's

Wedding et al. also used the output of the RBF network's basis functions, $\phi(v)$, to calculate a certainty factor value. The certainty factor was calculated from the input vector's proximity to the hidden layer's node centers rather than the data distribution of the training vectors. It should be noted that this function returns values between 0.0 and 1.0, which is the same range as allowed for by certainty factors.

When the output of a hidden layer node is high (near 1.0), this indicates that a point lies near the center of a cluster. This means that the data is familiar to that particular node and therefore the overall network is confident in the answer. A value of 1.0 exactly indicates the data falls directly upon the center of a node. A value that is very small (near 0.0) will lie far outside of a cluster. This data is therefore unfamiliar to that particular node. If all nodes have a small $\phi(v)$ value, then the network's certainty in the answer is small and the network is not confident in the result.

The problem with this scheme is that N values of $\phi(v)$ are generated, or one for each node in the hidden layer of the RBF neural network. Wedding [36] describes that,

one approach is to take the maximum $\phi(v)$ and consider it to be the certainty factor. From a certainty factor perspective, the disadvantage of using only the maximum $\phi(v)$ value when determining the certainty factor is that it disregards the certainty of the other nodes. This may result in an instance when two or more of the hidden nodes are reasonably familiar with an input data, but no node is *extremely* familiar with it. This would lead to the calculated certainty factor to be lower than expected, which could cause good results to be discarded.

In order to factor in the certainty of the other nodes, a variation of the maximum $\phi(v)$ equation is used which calculates the certainty using the recursive formula:

$$CF_i = CF_{i-1} + (1 - CF_{i-1}) max_i(\phi(v)) \tag{2.2}$$

In [Eq. 2.2], the value of i refers to the number of nodes that will be used to determine the certainty factor of the RBF network. In general, the value of i should be set to N (the number of nodes in the hidden layer), but it can also be set to a lower value if computational time is critical to the application. The term CF_{i-1} refers to the certainty factor of the network using only $i - 1$ nodes to calculate the confidence. The value of CF_0 (the certainty when no nodes in the hidden layer are used) is defined to be 0.0. The term $max_i(\phi(v))$ is defined as the i^{th} largest value of $\phi(v)$. For instance, max_1 would mean the largest value of $\phi(v)$ and max_2 would be the second largest value of $\phi(v)$, etc.

2.9 Comparing classifier performance

2.9.1 Classifier performance

A classification model (or classifier) is a mapping from instances to predicted classes. Given a classifier and an instance, there are four possible outcomes. If the instance is positive and it is classified as positive, it is counted as a *true positive*. If the same instance is classified as negative, it is counted as a *false negative*. If another instance is negative and it is classified as positive, it is counted as a *false positive*. Lastly, if that same negative instance is classified as negative, it is counted as a *true negative*.

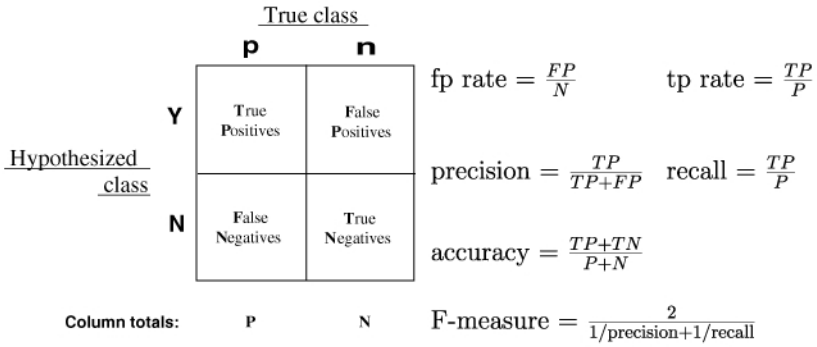

Figure 2.5 Confusion (contingency) matrix and common performance metrics calculated from it.

Given a classifier and a set of instances (the test set) a two-by-two matrix can be constructed. This contingency table, or as Fawcett has aptly named: *confusion matrix*[38], can be seen in [Fig. 2.5]. Two of these outcomes are successful (when the decision matches truth) and two are erroneous. Additional terms associated with the ROC are:

$$sensitivity = recall$$

$$specificity = \frac{true\ negatives}{false\ positives + true\ negatives} = 1 - fp\ rate$$

$$positive\ predictive\ value = precision$$

Even more terms can be seen in Appendix A.

2.9.2 The ROC curve

An ROC curve is a technique for visualizing, organizing, and selecting classifiers based on their performance. It plots the *true positive* rate of detection, against the corresponding *false positive* rate of error. The two numbers change in relation to each other as the detection threshold, or decision cutoff, varies.

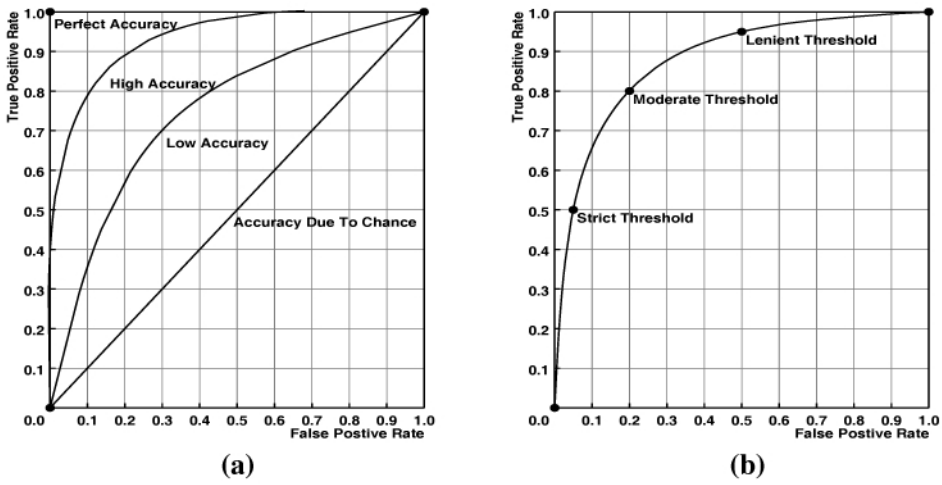

Figure 2.6 ROC curve interpretation. (a) A perfect ROC curve passes through the point (0,1) while an ROC curve no better than chance lies along the positive diagonal. (b) Each point along the ROC curve corresponds to a different decision threshold. Points closer to the origin (0,0) indicate a more exclusive criterion; points closer to the upper-right(1,0) indicate a more inclusive one.

ROC graphs have long been used in signal detection theory to depict the trade-

off between hit rates and false alarm rates of classifiers [39]. Such curves were first applied to asses how well radar equipment in World War II distinguished random interference from signals truly indicative of enemy planes [40]. There is extensive literature on the use of ROC graphs for diagnostic testing as Zou has shown with his online bibliography[41] as well as has been documented by Zweig[42]. Use of the ROC graphs in the machine learning community can be seen as early as 1989[43]. In addition to being a generally useful performance graphing method, they have properties that make them especially useful for domains with skewed class distributions and unequal classification error costs[38].

Two important aspects can be see in [Fig. 2.6]. In (a), different shapes of ROC curves indicate different levels of classifier accuracy. The better curves will bulge further outward to the up and to the left, nearing the point of perfection at (0,1). A perfect classifier will have a success rate of 1.0 for signals while having an error rate of 0.0 for noises. The worst ROC curve is the positive diagonal line stretching from (0,0) to (1,0); this accuracy is no better than random guessing. It is not possible for a classifier to do worse than chance (fall below the diagonal) because you could simply invert its output to achieve a better than chance result. We would reverse its classification decisions on every instance – its true positive classifications become false negative mistakes, and its false positives become true negatives.

In (b), three different threshold cutoffs are illustrated. Every ROC curve is based on the measurement of detector performance at various decision thresholds. On the ROC curve, the stricter thresholds appear closer to the point (0,0) and the more lenient thresholds appear closer to the point (1,0). The point (0,0) corresponds to wanting to say "No" all the time (leading to a low false-positive rate but also missing many true positives), and the point (1,0) corresponds to wanting to say "Yes" all the time (capturing nearly all the true positives, but at the expense of a high false-positive

rate).

2.9.3 Class skew

An attractive property that ROC curves have are the fact that they are insensitive to changes in class distribution. If the proportion of positive to negative instances changes in a test set, the ROC curves will not change. If we look at the contingency matrix in [Fig. 2.5], we see that the class distribution – the proportion of positive to negative instances – is the relationship of the left column (p) to the right column (n). Any performance metric that uses values from both columns will be inherently sensitive to class skews. Metrics such as accuracy, precision, lift and F score use values from both columns of the contingency matrix. As a class distribution changes these measures will change as well, even if the fundamental classifier performance does not. ROC graphs are based upon *true positive* rate and *false positive* rate, in which each dimension is a strict columnar ratio, so do not depend on class distributions.

2.9.4 Area under the ROC curve (AUC)

In order to compare classifiers, we may want to reduce ROC performance to a single scalar value representing expected performance. Bradley has recommended that the AUC be used in preference to overall accuracy when single number evaluation of machine learning algorithms is required [44]. Provost has also described the situation where accuracy is not a good measure: consider a domain where the classes appear in a 999:1 ratio [45, 46]. A simple rule, always, classify as the maxiumum likelihood class, gives a 99.9% accuracy. This is not satisfactory if a non-trivial solution is sought.

Since the AUC is a portion of the area of the unit square, its value will always be

between 0 and 1.0. As mentioned earlier, a random guessing classifier produces the diagonal line between (0,0) and (1,1), which has an area of 0.5, no realistic classifier should have an AUC less than 0.5.

The AUC has an important statistical property: the AUC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance [38].

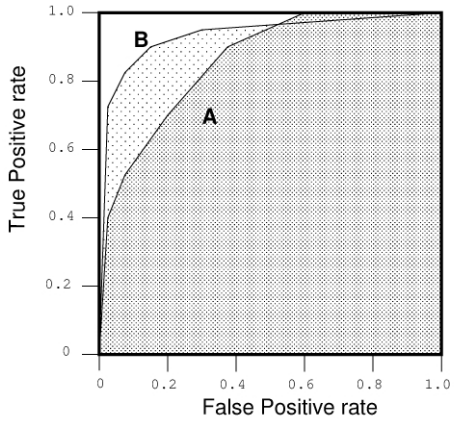

Figure 2.7 Which is the better classifier, A or B? Classifier B has greater area and therefore better average performance. It is generally better than A except at FP rate > 0.6, where A has a slight advantage.

One drawback of the AUC measure is that given a single AUC number, you cannot reconstruct the shape of the curve which produced it [47]. It should be noted that Maxion recommends "if there is a specific use in mind for the signal detector, then this may not be the best measure to use, because it gives only the most general picture of total accuracy" (similar to an average accuracy over all possible modes of the detector) [47]. He also notes that one ROC curve having a greater AUC may actually perform less well under some conditions than a different ROC curve having a smaller AUC number. We see an illustration of this in [Fig. 2.7]. Even though classifier B has a greater AUC, classifier A is actually superior when the false positive

rate is greater than 0.6. In situations where we want to use a more lenient threshold, classifier A would be the classifier of choice.

2.9.5 Convex hull

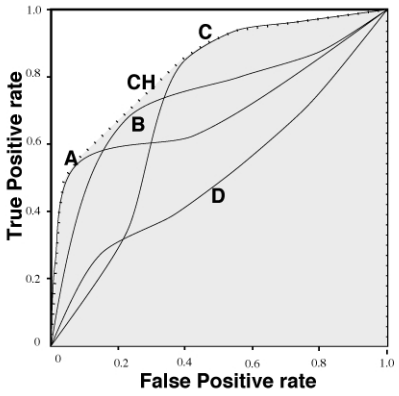

Figure 2.8 The ROC convex hull identifies potentially optimal classifiers.

The ROC convex hull (ROCCH) is a constructed "outer envelope" which includes that part of every ROC curve which dominates all other curves [Fig. 2.8]. If a single curve dominates all others over the entire graph, then it alone is the ROCCH. Otherwise, to construct the ROC convex hull you simply trace the outer envelope, making sure never to cave inwards or dip downwards, even if all other curves do at that point. The ROCCH represents the best possible performance of a group of detectors [47]. A classifier is potentially optimal if and only if it lies on the convex hull [48] of the set of points in ROC space. Since only classifiers on the convex hull are potentially optimal, no others need be retained.

We see in [Fig. 2.8] that D is clearly not optimal. B also can never be optimal because none of its points of its ROC curve lies on the convex hull. We can also remove from consideration any points of A and C that do not lie on the hull.

2.10 What we can learn from credit card fraud

As we move toward a cashless society, more and more transactions will be electronic in nature. Consumers are gradually moving away from paper payment instruments and toward electronic ones, especially payment cards[49]. Garcia-Swartz[49] finds that cash use dominates smaller transaction sizes but drops precipitously as transaction size increases. Fraud will occur during these large transactions when a criminal has the most to gain.

Our problem domain is similar to credit card fraud in the way that both data has the property of a skewed distribution. As stated earlier, 1.2% of applicants who take the telephone test are impaired and cheat to get a policy. From Brause's data [50], we see that fraud occurs at an even lower rate of 0.2%.

2.10.1 How to handle skewed distribution

Phua[51] states there are two typical ways to proceed when faced with this problem. The first approach is to apply different algorithm (meta-learning). Each algorithm has its unique strengths, so that it may perform better on particular data instances than the rest. The second approach is to manipulate the class distribution (sampling). The minority class training examples can be increased in proportion to the majority class in order to raise the chances of correct predictions by the algorithm(s). We plan to pursue the latter strategy.

According to Chan[52], the desired distribution of the data partitions belonging to a particular fraud detection data set must be determined empirically. They arrived at a determination that a 50:50 distribution gave the best models with their data[1].

[1]It should be noted that their data set had a fraud incident of 20%.

2.10.2 Many networks and the consensus vote

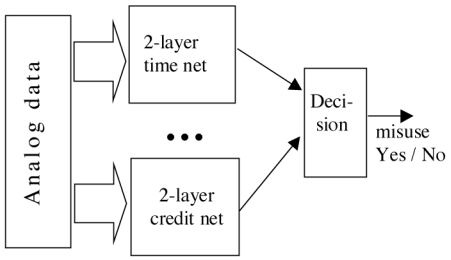

Figure 2.9 The neural network experts for analog data. Brause used several RBF neural networks, each one specialized on one topic.

Brause used a model [Fig. 2.9] in which they used one expert net for each feature group (time, credit, etc.) and grouped the experts together to form a common vote[50].

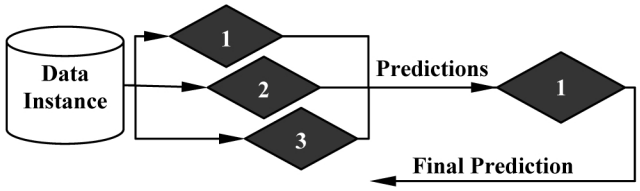

Figure 2.10 Predictions on a single data instance using precogs. Phua calls his components precogs, or precognitive elements. Each precog contains multiple black-box classification models, or classifiers, trained with one data mining technique.

In a similar vein, Phua[51] calls his units, precogs, short for precognitive elements [Fig. 2.10]. Instead of utilizing diversity of specialization, as Brause does, he utilizes diversity of classification models. For example, the first, second, and third precog are trained using the statistical paradigm, computer metaphor, and brain metaphor respectively. They require numerical inputs of past examples to output corresponding class predictions for new instances. As each precog outputs its many predictions for

each instance, all the predictions are fed back into one of the precogs, to derive a final prediction for each instance.

Chapter 3

Research Goal

The classes of this classification problem are as follows:

Class	True cog. stat.	Cheat stat.	Telephone cog. stat.	Face-to-face cog. stat.
1	impaired	*non*-cheater	impaired	impaired
2	*non*-impaired	*non*-cheater	*non*-impaired	*non*-impaired
3	impaired	cheater	*non*-impaired	impaired
4	*non*-impaired	cheater	*non*-impaired	*non*-impaired

Table 3.1 EMST cheating/impairment status classification.

Cheaters use external aids to get a normal result on the EMST. Because of the high future claims cost associated with issuing an LTC policy to an impaired individual, the primary goal of my research is to efficiently distinguish impaired cheaters (Class 3) from the rest. By efficiently, I mean that I want to maximize detection of impaired cheaters (true positives) while minimizing the inclusion of normal cheaters (false positives). With impaired cheaters comprising approximately 1.2% of all individuals taking the telephone EMST, a high specificity must be achieved, even at

the expense of lowering sensitivity. Specificity is increased when the number of false positives is reduced while sensitivity is lowered when the number of false negatives are increased.

3.1 Objectives

1. **Create a classifier that minimizes false positives, thereby maximizing specificity and positive predictive value.** To reiterate, our priority is to reduce the number of false positives. This has the effect of increasing specificity and positive predictive value, which is defined as $\frac{TN}{TN+FP}$ and $\frac{TP}{TP+FP}$, respectively. By minimizing false positives (FP) we maximize both specificity and positive predictive value. When looking at comparative ROC curves, [Fig. 2.7], to determine which is a better classifier, we prefer a curve that looks more like classifier B than classifier A. We prefer it not for its larger area under the curve (AUC), but for the property of having a curve that produces a greater true positive rate for any given *low*-valued false positive rate. We prefer a curve that leans more towards the upper left of the graph. In summary:

   (a) Create a classifier that minimizes false positives. We don't have to catch all the impaired cheaters, however the ones we do catch must truly be *impaired* and *cheating*.

   (b) Use comparative ROC curves (plots with more than one classifier) to assess which classifier is *better*.

2. **Compare created classifiers to a benchmark.** A statistical classifier, using logistic regression, has already been developed to predict impaired cheaters[1].

[1] Unpublished data

This is the benchmark we will compare our classifiers with. As mentioned in [Section 2.7], we can not predict *a priori* whether or not our neural networks will do better than this classifier.

3. **Use multiple design strategies.** In order to accomplish our first objective, we will design neural networks with varying properties. These networks will be compared with the logistic regression classifier as well as with each other.

   (a) Compare different learning algorithms.

   - Levenberg-Marquardt
   - Backpropagation

   (b) Compare different neural network architectures

   - Multi-layer preceptron (MLP)
   - Radial basis function (RBF)

   (c) Compare effect of using certainty factors with RBFs

   - RBF without certainty factors
   - RBF with certainty factors

4. **Create a simple GUI to tie it all together.** A simple GUI will be created to allow its user to see, first hand, the effects of the various neural network parameters on performance. The GUI will allow its user to do the following things:

   (a) load a datafile

   (b) select the type of network to be used

   - MLP

- RBF
- RBF with certainty factors

(c) train the network

- the user will see the results of the training in the form of a plot of training iteration vs. root mean squared error

(d) test the network and show the following results:

- an ROC plot comparing it against the logistic regression benchmark
- area under the curve
- specificity
- sensitivity
- positive predictive value

Chapter 4

Methodology

We will examine the performance of a 1-stage and 2-stage RBF's in distinguishing impaired cheaters from other subjects. We will also use the same design strategies to create a MLP so that we can compare their performance with each other. We will empirically determine the distribution of data that produces the best model, for all cases. For both the RBF and MLP, we will employ the use of two different learning algorithms for comparison: Levenberg-Marquardt and Backpropagation. In addition to varying the learning algorithms, we will add the use of certainty factors in the RBF's. An overview of the experimental design can be seen in [Table 4.2]. The data that we will collect for each permutation, as can also be seen in [Table 4.1], will be: ROC plots, area under the curve (AUC), sensitivity, positive predictive value (PPV). We will also generate comparative ROC plots for the various permutations.

If time permits, we will construct a GUI that will allow the user to select one of the permutations found in [Table 4.2], load data in to the neural network for training and then produce data output, such as the attributes in [Table 4.1].

Neural network attribute collected
ROC plots
area under the curve (AUC)
sensitivity
specificity
positive predictive value (PPV)

Table 4.1 For each permutation, the above will be collected

4.1 Data preparation

There are 7557 cases in our data-set. These cases are those that were assessed by telephone and deemed to be *non*-impaired [Fig. 4.1], which includes Classes 2, 3 and 4 of [Table 3.1]. From this set, there are 2228 that have been determined to be cheaters (Class 3 and 4). Of these cheaters, 34 of them are impaired (Class 4).

We will initially segregate this data-set into two pools: a training pool and a test (validation) pool. Cases will be selected at random to populate the two pools. We will set the ratio of training:test to be 80:20 so that we may maximize the number of cases the classifiers may learn from. A truly random selection process should yield similar proportions of cheaters and impaireds as that which was found in the original population of cases. This is our starting point.

After our initial investigation, we will re-generate these training and test pools, again in a random fashion, to verify reproducibility of our classifiers.

1-stage	MLP	RBF	RBF + CF
Levenberg-Marquardt			
Backpropagation			
2-stage	MLP	RBF	RBF + CF
Levenberg-Marquardt			
Backpropagation			

Table 4.2 Overview of experimental plan

4.2 Two stage method

Two RBF NNs will be created [Fig. 4.2]. RBF1 will be trained to distinguish between cheaters and *non*-cheaters. Subjects classified as cheaters will be passed into the second stage RBF NN, RBF2. RBF2 will be trained to distinguished between impaired and *non*-impaired cheaters. We will create the MLP NNs in the same fashion.

4.2.1 RBF1 and MLP1

Data that contain Classes 2, 3 and 4 of [Table 3.1] will be used to randomly select training data from a pool of previously verified *non*-cheaters, *non*-impaired cheaters and impaired cheaters. Initially we will not manipulate the distribution of the populations: *non*-cheaters, *non*-impaired cheaters and impaired cheaters. We will investigate the effect of manipulating the distribution of classes and its impact to the classifier model. In general, we will use 50:50 (cheater:*non*-cheater) as a starting point. Data sets: A, C, D of [Table 4.3] contain the aforementioned classes.

The outcome nodes will correspond to the classes of: cheater (Classes 3 and 4)

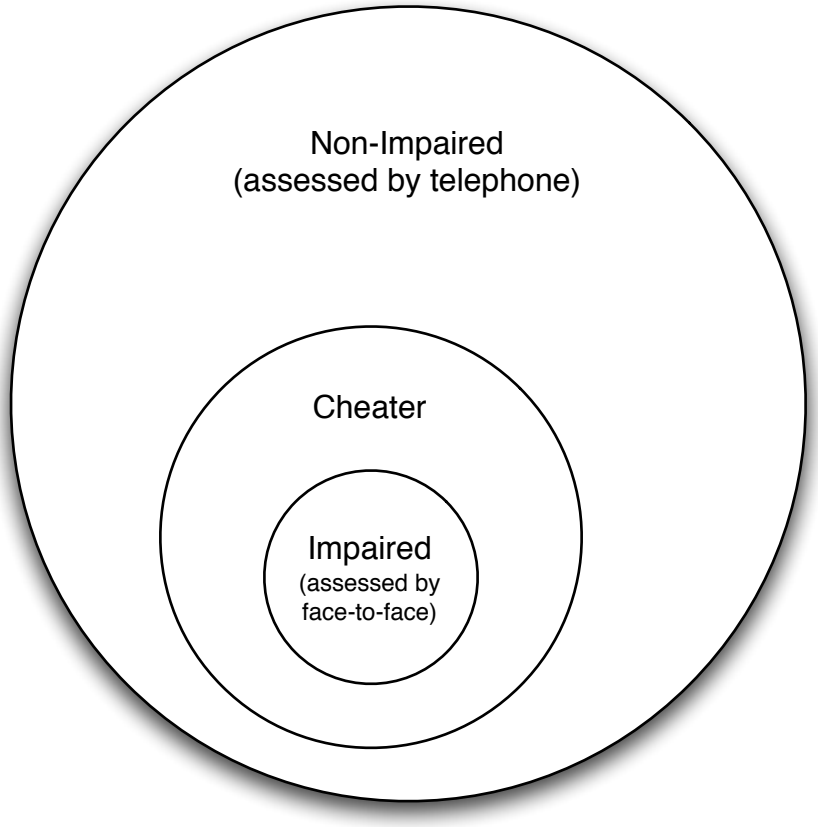

Figure 4.1 Venn diagram of classes in data. The *non*-impaired, assessed by telephone, set represents Classes 2, 3 and 4 of [Table 3.1]. The cheater subset of *non*-impaired, assessed by telephone, represents Classes 3 and 4. Lastly, the impaired, assessed by face-to-face, subset of cheater, represents Class 3, our class of interest. Note: diagram is not drawn to scale.

and *non*-cheater (Class 2).

4.2.2 RBF2 and MLP2

Data from Classes 3 and 4 of [Table 3.1] will be used to randomly select training data for RBF2 and MLP2. We will initially allow the distribution of the populations: *non*-impaired cheaters and impaired cheaters to be unaltered. Keep in mind the pool of impaired cheaters is heavily skewed. To investigate the effects of distribution to

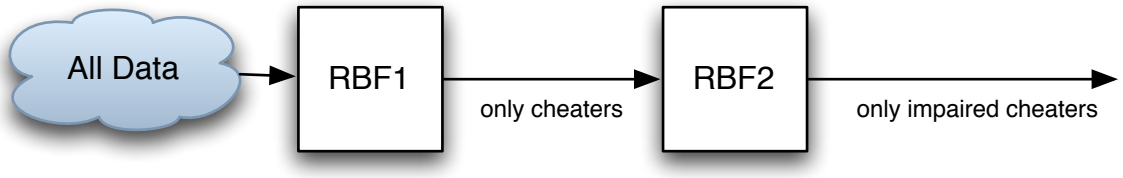

Figure 4.2 Two stage method. Two specialized RBF networks are applied serially. RBF1 filters data so that only cheaters are passed to RBF2. RBF2 applies its speciality of distinguishing impaired cheaters vs. *non*-impaired cheaters.

the classifier model, we will expose the classifiers to repeated instances of the skewed pool. Data sets: C and D of [Table 4.3] contain the aforementioned classes.

The outcome nodes will correspond to the classes of *non*-impaired cheater (Class 1) and impaired cheater (Class 3).

4.3 One stage method

The same technique used to select training samples for RBF1 and MLP1 will be used to train the one stage RBF NN and one stage MLP NN. The distribution of impaired cheater to *non*-cheater and unimpaired cheater will be determined empirically. Outcome nodes will correspond to the class of impaired cheater (Class 3 [Table 3.1]) and the rest, normal cheater or normal non-cheater (Classes 2 and 4 [Table 3.1]).

4.4 Learning algorithms

The Levenberg-Marquardt algorithm is a hybrid of the Gauss-Newton algorithm and the Steepest descent algorithm [59]. It is a compromise between very slow convergence, but converges from anywhere (Steepest descent) and very fast convergence, but converges only close to the optimum (Gauss-Newton).

We intend to compare the Levenberg-Marquardt algorithm with the Backpropagation algorithm. Backpropagation is similar to the Steepest descent algorithm except that step length is held constant. The optimum step length will be determined experimentally. In addition to step length, a momentum term may be specified. Momentum encourages movement in a fixed direction. Thus, if several steps are taken in the same direction, the algorithm "picks up speed", which gives it the ability to (sometimes) escape local minimum.

4.5 Specificity and positive predictive value

One of the important measures that we will look at are specificity and positive predictive value (PPV). A description of what these values represent can be found in [Appendix A]. For the purposes of detecting impaired cheaters for the insurance industry, specificity and PPV are important because they would like to *rule-in* whether an applicant is a cheater or not. The desire is to have the lowest rate of false positives (both specificity and PPV are greatest, when false positives go to 0). If the application were a public screening for early detection of dementia, then our desire would be for a high sensitivity. In this case, we would want to detect as many impaired people as possible, at the risk of having a higher rate of false positives.

4.6 Certainty factors

The certainty factors will be calculated according to [Eq. 2.2] developed by Wedding et al. [36]. An optimal cutoff value will be selected so as to maximize specificity and PPV.

4.7 GUI

We will attempt to create a GUI using Mathworks' MATLAB. The main purpose of the GUI will be to allow users to "re-run" some of our experiments and reconfirm some of the conclusions made in this paper. The GUI allow the user to initialize, train and run a sampling of pre-configured neural networks on-the-fly.

4.8 Tools that will be used

To build the initial RBF and MLP models, we will use Wolfram Research's Mathematica 5.2 with the Neural Networks add-on application library. The next step would be to re-create these models using MATLAB. This step is required in order to implement certainty factors. Mathematica's Neural Network add-on does not allow access to the hidden layer, which MATLAB does; this is necessary in order to calculate certainty factors.

The MATLAB will be used to plot Receiver Operating Characteristic Curves (ROC curves) and to calculate AUC.

The data will be housed in a Oracle 10g Express Edition database.

Set	Name	Description	Assumptions	Class [a]
A	Home assessments	Tests conducted face-to-face.	They could not cheat because the test administrator was physically present and watching.	**1**, **2**
B	Telephone assessments	Tests conducted over the telephone.	No assumptions made.	1, 2, 3, 4
C	Telephone with subsequent home [normal → impaired]	Applicants suspected of cheating on the telephone assessment were assessed again in a face-to-face setting. The results of the face-to-face testing were used to classify applicants as either normal cheaters (Class 4) or impaired cheaters (Class 3).	Discrepancy of test results is due to cheating	**3**, **4**
D	Cheating study	Employees of Lifeplans, Inc. were asked to take the EMST twice. Each employee was randomized to cheating on the first or second test. On the remaining test they were instructed not to cheat.	No assumptions made.	**2**, **4**

Table 4.3 Data that will be used in the analysis. Class numbers in bold indicate that the classes can be distinguished with certainty.

[a] see [Table 3.1]

Chapter 5

Evaluation of Results

To evaluate the performance of the RBF and MLP neural networks, we will plot ROC curves for each classifier on the same graph. We will use the area under the Receiver Operating Characteristic (ROC) curve as a general comparison between classifiers. If a dominating curve can not be visual discovered, we will use the ROC convex hull (ROCCH) to determine whether the classifier is an optimal in a specific region. We will also look closely at specificity and positive predictive value (PPV).

Chapter 6

Tentative Table of Contents for the Thesis

References

Appendices

Chapter 7

Tentative Timetable for Completion of the Thesis

July 1, 2006	Began modeling in Mathematica
September 1, 2007	Finished modeling in Mathematica
August 9, 2007	Submitted *final* draft of proposal
January 25, 2008	Complete research
February 1, 2008	Turn in *first* draft of thesis
February 28, 2008	Turn in *final* draft of thesis
March 15, 2008	Defend thesis

Bibliography

[1] G. Lamberty, C. Kennedy, and L. Flashman, "Clinical utility of the CERAD word list memory test.," Appl Neuropsychol **2,** 170–173 (1995).

[2] R. Larumbe, "[Detection of early cases of Alzheimer's disease. Application of the CERAD neuropsychological test battery]," Rev Med Univ Navarra **41,** 6–11 (1997).

[3] K. Welsh, N. Butters, J. Hughes, R. Mohs, and A. Heyman, "Detection of abnormal memory decline in mild cases of Alzheimer's disease using CERAD neuropsychological measures.," Arch Neurol **48,** 278–281 (1991).

[4] H. Kirshner, "Mild cognitive impairment: to treat or not to treat.," Curr Neurol Neurosci Rep **5,** 455–457 (2005).

[5] J. Morris, "Mild cognitive impairment and preclinical Alzheimer's disease.," Geriatrics **Suppl,** 9–14 (2005).

[6] M. Boustani, B. Peterson, L. Hanson, R. Harris, and K. Lohr, "Screening for Dementia in Primary Care: A summary of the evidence for the U.S. Preventive Services Task Force," Annals of Internal Medicine **138,** 927–937 (2003).

[7] V. Crooks, L. Clark, D. Petitti, H. Chui, and V. Chiu, "Validation of multi-stage telephone-based identification of cognitive impairment and dementia.," BMC Neurol **5,** 8 (2005).

[8] C. Lines, K. McCarroll, R. Lipton, and G. Block, "Telephone screening for amnestic mild cognitive impairment.," Neurology **60,** 261–266 (2003).

[9] R. Lipton, M. Katz, G. Kuslansky, M. Sliwinski, W. Stewart, J. Verghese, H. Crystal, and H. Buschke, "Screening for dementia by telephone using the memory impairment screen.," J Am Geriatr Soc **51,** 1382–1390 (2003).

[10] J. Hill, J. McVay, A. Walter-Ginzburg, C. Mills, J. Lewis, B. Lewis, and H. Fillit, "Validation of a brief screen for cognitive impairment (BSCI) administered by telephone for use in the medicare population.," Dis Manag **8,** 223–234 (2005).

[11] M. Barber and D. Stott, "Validity of the Telephone Interview for Cognitive Status (TICS) in post-stroke subjects.," Int J Geriatr Psychiatry **19,** 75–79 (2004).

[12] D. Liston, "LifePlans, Inc. Announces Innovative Approach to Detect Mild Cognitive Impairment and Early Alzheimer's Disease," http://www.lifeplansinc.com/news/EMST.htm (2005).

[13] W. Shankle, A. Romney, J. Hara, D. Fortier, M. Dick, J. Chen, T. Chan, and X. Sun, "Methods to improve the detection of mild cognitive impairment.," Proc Natl Acad Sci U S A **102,** 4919–4924 (2005).

[14] S. Haykin, *Neural Networks: A Comprehensive Foundation (2nd Edition)* (Prentice Hall, 1998).

[15] M. A. Arbib, *The handbook of brain theory and neural networks* (MIT Press, Cambridge, Mass, 2003), pp. xvii, 1290 p.

[16] T. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," IEEE Transactions on Electronic Computers **14,** 326–334 (1965).

[17] D. Rumelhart and J. McClelland, *Learning internal representations by error back propagation* (MIT Press, 1986), Vol. 1.

[18] D. Broomhead and D. Lowe, "Multivariable Functional Interpolation and Adaptive Networks," Complex Systems **2,** 321–355 (1988).

[19] S. Fredrickson and L. Tarassenko, in *Radial basis functions for speaker identification* (Martigny, Switzerland, 1994), pp. 107–110.

[20] M. Mak, W. Allen, and G. Sexton, "Speaker identificatoin using multilayer perceptrons and radial basis function networks," Neurocomputing **6,** 99–117 (1994).

[21] J. Oglesby and J. Mason, in *Radial basis function networks for speaker recognition* (Toronto, Canada, 1991), Vol. 1, pp. 393–396.

[22] R. Finan, A. Sapeluk, and R. Damper, in *Comparison of Multilayer and Radial Basis Function Neural Netowks for Text-dependent Speaker Recognition* (Washington, DC, USA, 1996), Vol. 4, pp. 1992–1997.

[23] M. Barletta and A. Gisario, "An application of neural network solutions to laser assisted paint stripping process of hybrid epoxy-polyester coatigns on aluminum subsctrates," Surface and Coatings Technology **200:24,** 6678–6689 (2006).

[24] V. Mitra, C. Wang, and S. Banerjee, "Lidar detection of underwater objects using a neuro-SVM-based architecture.," IEEE Trans Neural Netw **17,** 717–731 (2006).

[25] M. Niranjan and F. Fallside, "Neural networks and radial basis function in classifying static speech patterns," Computer Speech and Language **4,** 275–289 (1990).

[26] M. Er, S. Wu, J. Lu, and H. Toh, "Face recognition with radial basis function (RBF) neural networks," IEEE Transactions on Neural Networks **13:3,** 697–710 (2002).

[27] D. Lowe and A. Webb, "Exploiting prior knowledge in network optimization: an illustration from medical prognosis," Network **1,** 299–323 (1990).

[28] A. Webb, "Functional approximation by feed-forward networks: A least-squares approach to generalization," IEEE Transactions on Neural Networks **5:3,** 363–371 (1993).

[29] Y. Lee, "Handwritten digit recognition using K Nearest Neighbour, Radial Basis Function, and Backpropagation Neural Networks," Neural Computation **3,** 440–449 (1991).

[30] B. Wilson and P. Watson, "A practical framework for understanding compensatory behaviour in people with organic memory impairment.," Memory **4,** 465–486 (1996).

[31] D. Sargent, "Comparison of artificial neural networks with other statistical approaches: results from medical data sets.," Cancer **91,** 1636–1642 (2001).

[32] R. Chaveesuk, C. Srivaree-Ratana, and A. Smith, "Alternative neural network approaches to corporate bond rating," Journal of Engineering Valuation and Cost Analysis **2,** 117–131 (1999).

[33] S. Dreiseitl, L. Ohno-Machado, H. Kittler, S. Vinterbo, H. Billhardt, and M. Binder, "A comparison of machine learning methods for the diagnosis of pigmented skin lesions.," J Biomed Inform **34,** 28–36 (2001).

[34] B. Buchanan and E. Shortliffe, *Rule-based expert systems: the mycin experiments of the Standford heuristic programming project* (Addison-Wesley, 1984).

[35] D. Wedding and K. Cios, "Certainty factors versus Parzen windows as reliability measures in RBF networks," Neurocomputing **19,** 151–165 (1998).

[36] D. Wedding and K. Cios, "Time series forecasting by combining RBF networks, certainty factors, and the Box-Jenkins model," Neurocomputing **10,** 149–168 (1996).

[37] J. Leonard, M. Kramer, and L. Ungar, "Neural network architecture that computes its own reliability.," Computers & Chemical Engineering **16,** 819–835 (1992).

[38] T. Fawcett, "An introduction to ROC analysis," Pattern Recognition Letters **27,** 861–874 (2006).

[39] J. Egan, *Signal Detection Theory and ROC Analysis* (Academic Press, New York, 1975).

[40] J. Swets, R. Dawes, and J. Monahan, "Better Decisions through Science," Scientific American **283,** 82–87 (2000).

[41] K. Zou, "Receiver operating characteristic (ROC) literature research," Online bibliography (2002).

[42] M. Zweig and G. Cambell, "Receiver-Operating Characteristic (ROC) Plots: A Fundamental Evaluation Tool in Clinical Medicine," Clincal Chemistry **39:4,** 561–577 (1993).

[43] K. Spackman, *Signal detection theory: Valuable tools for evaluating inductive learning* (Morgan Kaufmann, 1989), pp. 160–163.

[44] A. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," Pattern recognition **30,** 1145–1159 (1997).

[45] F. Provost and T. Fawcett, in *Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distributions* (AAAI Press, Menlo Park, CA, 1997), pp. 43–48.

[46] F. Provost, T. Fawcett, and R. Kohavi, in *The Case Against Accuracy Estimation for Comparing Induction Algorithms* (Madison, WI, 1998).

[47] R. Maxion and R. Roberts, "Proper Use of ROC Curves in Intrusion/Anomaly Detection," National Technical Information Service **CS-TR-871,** 38 (2004).

[48] C. Barber, D. Dbkin, and H. Huhdanpaa, "The Quickhull Algorithm for Convex Hulls," ACM Transactions on Mathematical Software **22,** 469–483 (1996).

[49] D. Garcia-Swartz, R. Hahn, and A. Layne-Farrar, "The Move Toward a Cashless Society: A Closer Look at Payment Instrument Economics," Review of Network Economics **5,** 175–198 (2006).

[50] R. Brause, T. Langsdorf, and M. Hepp, in *Neural Data Mining for Credit Card Fraud Detection* (1999), Vol. ICTAI, pp. 103–106.

[51] C. Phua, D. Alahakoon, and V. Lee, "Minority Report in Fraud Detection: Classification of Skewed Data," ACM SIGKDD Explorations Newsletter **6,** 50–58 (2004).

[52] P. Chan, W. Fan, A. Prodromidis, and S. Stolfo, "Distributed data mining in credit card fraud detection," IEEE Intelligent Systems **Nov/Dec,** 67–74 (1999).

[53] A. Borque, G. Sanz, C. Allepuz, L. Plaza, P. Gil, and L. Rioja, "The use of neural networks and logistic regression analysis for predicting pathological stage in men undergoing radical prostatectomy: a population based study.," J Urol **166,** 1672–1678 (2001).

[54] Y. Li, L. Liu, W. Chiu, and W. Jian, "Neural network modeling for surgical decisions on traumatic brain injury patients.," Int J Med Inform **57,** 1–9 (2000).

[55] E. Bartfay, W. Mackillop, and J. Pater, "Comparing the predictive value of neural network models to logistic regression models on the risk of death for small-cell lung cancer patients.," Eur J Cancer Care (Engl) **15,** 115–124 (2006).

[56] F. Chun, M. Graefen, A. Briganti, A. Gallina, J. Hopp, M. Kattan, H. Huland, and P. Karakiewicz, "Initial Biopsy Outcome Prediction-Head-to-Head Comparison of a Logistic Regression-Based Nomogram versus Artificial Neural Network.," Eur Urol (2006).

[57] M. Green, J. Bjork, J. Forberg, U. Ekelund, L. Edenbrandt, and M. Ohlsson, "Comparison between neural networks and multiple logistic regression to predict acute coronary syndrome in the emergency room.," Artif Intell Med **38,** 305–318 (2006).

[58] I. Kaiserman, N. Kaiserman, and J. Pe'er, "Long term ultrasonic follow up of choroidal naevi and their transformation to melanomas.," Br J Ophthalmol **90,** 994–998 (2006).

[59] S. Roweis, "Levenberg-Marquardt Optimization," .

Appendix A

Test Characteristics

Sensitivity: The probability of the test finding disease among those who have the disease or the proportion of people with disease who have a positive test result.

$$Sensitivity = \frac{true\ positives}{true\ positives + false\ negatives}$$

Specificity: The probability of the test finding NO disease among those who do NOT have the disease or the proportion of people free of a disease who have a negative test.

$$Specificity = \frac{true\ negatives}{true\ negatives + false\ positives}$$

Positive Predictive Value (PPV): The percentage of people with a positive test result who actually have the disease.

$$Positive\ predictive\ value = \frac{true\ positives}{true\ positives + false\ positives}$$

Negative Predictive Value (NPV): The percentage of people with a negative test who do NOT have the disease.

$$Negative\ predictive\ value = \frac{true\ negatives}{true\ negatives + false\ negatives}$$

SnOut: Sensitive test to rule-out a disease. When a diagnostic test or sign has a high sensitivity, a negative result rules out the diagnosis.

Why **SnOut**? Sensitivity and Negative Predictive Value share FN. As Sensitivity increases toward 100%, FN decreases toward 0. As FN decreases toward 0, Negative Predictive Value increases toward 100%.

SpIn: Specific test to rule-in a disease. When a diagnostic sign or test has a high specificity, a Positive result rules in the diagnosis.

Why **SpIn**? Specificity and Positive Predictive Value share FP. As Specificity increases toward 100%, FP decreases toward 0. As FP decreases toward 0, Positive Predictive Value increases toward 100%.