1995

Using back propagation neural networks for prediction of individual cell performance in a long string lead-acid peak shaving battery

Richard E. Young
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_theses

Recommended Citation

Young, Richard E., "Using back propagation neural networks for prediction of individual cell performance in a long string lead-acid peak shaving battery" (1995). *Master's Theses*. 1044.
DOI: <https://doi.org/10.31979/etd.bb6z-pp5s>
https://scholarworks.sjsu.edu/etd_theses/1044

This Thesis is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Theses by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

**A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600**

**USING BACK PROPAGATION NEURAL NETWORKS FOR
PREDICTION OF INDIVIDUAL CELL PERFORMANCE IN A
LONG STRING LEAD-ACID PEAK SHAVING BATTERY**

A Thesis

Presented to

The Faculty of the Department of

Chemistry

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Richard E. Young

May 1995

UMI Number: 1374635

UMI Microform 1374635

Copyright 1995, by UMI Company. All rights reserved.

**This microform edition is protected against unauthorized
copying under Title 17, United States Code.**

UMI

**300 North Zeeb Road
Ann Arbor, MI 48103**

© 1995

Richard E. Young

ALL RIGHTS RESERVED

APPROVED FOR THE DEPARTMENT OF CHEMISTRY

A handwritten signature in black ink, appearing to read "Sam Perone", written over a horizontal line.

Dr. Sam Perone

A handwritten signature in black ink, appearing to read "Joseph Pesek", written over a horizontal line.

Dr. Joseph Pesek

A handwritten signature in black ink, appearing to read "Viorica Lopez-Avila", written over a horizontal line.

Dr. Viorica Lopez-Avila

APPROVED FOR THE UNIVERSITY

A handwritten signature in black ink, appearing to read "Serena H. Stanford", written over a horizontal line.

Abstract

USING BACK PROPAGATION NEURAL NETWORKS FOR PREDICTION OF INDIVIDUAL CELL PERFORMANCE IN A LONG STRING LEAD-ACID PEAK SHAVING BATTERY

By Richard E. Young

Neural networks were explored for the first time to predict the performance of individual cells in a large lead-acid battery. A simple time based index of routine maintenance events from a three year period were used as inputs into the neural networks. Using the back propagation algorithm, the neural networks were trained to recognize three performance classes of cells (low, medium, and high), as measured by their capacity. The objective was to accurately identify the low and high performing cells. The high performing cells could be predicted with 100% accuracy with less than 9% false positive identification of the poor performing cells. Alternatively, the number of false positives could be kept to less than 5% of the poor performing cells but the high performing cells were predicted with 95% accuracy. The low performing cells could be predicted with 95.7% accuracy with no false positives due to high performing cells.

Dedication and Acknowledgments

This work is dedicated to my wife, Diana, who put up with months and months of late night “neuraling.” With her encouragement and support this project saw fruition.

My sincere thanks to all of those who helped me complete this project. Foremost, I am grateful to Dr. Sam Perone, Professor of Chemistry, San Jose State University, who introduced me to neural networks and provided me with productive advice and direction. My thanks to Dean Miklavic of NeuralWare, Inc., who patiently answered several questions concerning the application of the NeuralWorks Professional II/PLUS software. I am indebted to Dr. Viorica Lopez-Avila, Director of Midwest Research Institute - California Operations, who volunteered to serve on my graduate committee and provided me with many valuable suggestions. And sincere appreciation goes to Dr. Joseph Pesek, San Jose State University Chemistry Department Chair, for taking time out of his busy schedule to serve on my graduate committee.

Table Of Contents

Abstract	iv
Dedication and Acknowledgments	v
List of Tables	ix
List of Figures	x

Chapter 1 - Introduction

1.1	Electrical Energy Peak Shaving with Lead-Acid Batteries	1
1.1.1	Peak Shaving: What Is It and Why Do It?	1
1.1.2	The EPRI Battery: Purpose and History	3
1.1.3	The Chemistry of Lead-Acid Batteries	6
1.2	Previous Studies of Predicting and Modeling Cell Performance	9
1.2.1	First Exploration: Predicting Nickel-Cadmium Cell Performance Using Initial Characteristics	9
1.2.2	Extending the Concept to Predicting Lead-Acid Cell Performance Using Initial Characteristics	10
1.2.3	Investigations Into Predicting Cell Performance for EPRI's GNB Lead-Acid Battery	11
1.2.3.1	Investigation in the Use of EPRI's GNB Battery Fabrication/ Test Data for Cell Performance Prediction	12
1.2.3.2	Investigation in the Use of EPRI's GNB Battery Routine Maintenance Data for Cell Performance Prediction	13
1.2.3.3	Investigation in the Use of EPRI's GNB Battery Combined Fabrication/ Test and Maintenance Data for Cell Performance Prediction	14
1.2.4	Theoretical Modeling of Cell Reversals in Long String Lead-Acid Batteries	16
1.3	Neural Networks	16
1.3.1	A Short History of Neural Networks	18
1.3.2	Back Propagation Neural Networks	21
1.3.2.1	Phases of Developing a Back Propagation Neural Network ...	21
1.3.2.2	Structure of a Back Propagation Neural Network	22

1.3.2.3 Training: Supervised Training of a Back Propagation Neural Network	27
1.3.2.4 Recall: Evaluation of a Trained Neural Network with the Training Set	32
1.3.2.5 Test: Evaluation of A Trained Neural Network With a Test Set	32
1.3.3 Other Neural Networks	33

Chapter 2 - Experimental

2.1 Instrumentation and Software	34
2.2 Database	34
2.2.1 Raw Database	34
2.2.2 Feature Database	35
2.2.2.1 Data Preprocessing	35
2.2.2.2 Definition of Class Boundaries	36
2.2.2.3 Data Sampling	37
2.2.2.4 Training and Test Sets	38
2.3 Feature Selection	38
2.4 Neural Network Supervised Training	39
2.4.1 Measuring Neural Network Performance	39
2.4.1.1 Classification Accuracy	40
2.4.1.2 Confusion Matrix	41
2.4.1.3 Risk Weighted Classification Accuracy	41
2.4.2 Neural Network Design Parameters	42
2.4.2.1 Neural Network Architecture	43
2.4.2.2 Neural Network Input Scaling	46
2.4.2.3 Transfer Function	48
2.4.2.4 Learning Rule	48
2.4.2.5 Number of Training Cycles	49
2.4.2.6 Epoch Size	49
2.4.2.7 Error Surface Starting Position	50

Chapter 3 - Data Preprocessing

3.1 Raw Database	52
------------------	----

3.2	Feature Database	53
3.2.1	Features	53
3.2.2	Class Encoding in the Feature Database	53
3.3	Data Sampling: Selection of Cells for the Training and Test Sets	55
3.4	Training and Test Sets	58

Chapter 4 - Results and Discussion

4.1	Objective of the Investigation	60
4.2	Feature Selection	61
4.3	Neural Network Optimization	67
4.3.1	Three Class Optimization	69
4.3.2	Two Class Optimization	70
4.3.2.1	Class 1 versus Classes 2 and 3	70
4.3.2.2	Class 2 versus Classes 1 and 3	72
4.3.3	Single Class Optimization Within a Three Class Neural Network Classifier	72
4.4	Interpreting the Neural Network	77
4.5	Conclusions	81
	References	84
	Appendix A - Neural Network Parameters	92
	Appendix B - Cells Selected for the Training Set	101
	Appendix C - Cells Selected for the Test Set	103

List of Tables

1) Time indexing of raw database using “months until capacity test”	53
2) Neural network input codes for capacity test event features	54
3) Number of cells within the defined class boundaries for a capacity test event	55
4) Number of cells selected from the capacity test events for producing the training and test sets	58
5) Origin and number of cells forming the combined and randomized training set	59
6) Origin and number of cells forming the combined and randomized test set	59
7) Confusion matrices (percent classified), overall percent classification accuracies and total risk factors for 3-class neural network test and training sets	71
8) Confusion matrices (percent classified), overall percent classification accuracies and total risk factors for 2-class neural network test and training sets: class 1 versus classes 2 and 3	73
9) Confusion matrices (percent classified), overall percent classification accuracies and total risk factors for 2-class neural network test and training sets: class 2 versus classes 1 and 3	74
10) Confusion matrices (percent classified), overall percent classification accuracies and total risk factors for Case 1 neural network test and training sets	76
11) Confusion matrices (percent classified), overall percent classification accuracies and total risk factors for Case 2 neural network test and training sets	78

List of Figures

1) A representative electrical energy demand curve with a “shaved” peak	2
2) The combined distribution of capacity among the cells at CEMC from 1989, 1990, and 1991	4
3) The age dependance of capacity in a lead-acid battery	5
4) The lead-acid battery charge/discharge cycle	8
5) The perceptron’s limitation with the XOR problem	20
6) The structure of a feed forward back propagation neural network	23
7) The structure of an individual processing element	24
8) Supervised training of a back propagation neural network	28
9) A two dimensional representation of a hyperspace error surface	31
10) A Hinton diagram	40
11) A three class confusion matrix	42
12) A risk factor matrix for a confusion matrix	43
13) Overtraining a neural network with too many hidden nodes	44
14) Decision regions created by two inputs and three hidden nodes	47
15) Overtraining a neural network with too many training cycles	50
16) The optimized neural network architecture	68

Chapter 1 - Introduction

1.1 Electrical Energy Peak Shaving with Lead-Acid Batteries

1.1.1 Peak Shaving: What Is It and Why Do It?

A representative electrical energy demand curve has peaks and valleys. Typically, electrical energy demand is low during the night, increases steadily during the morning, peaks in the late afternoon, and then declines during the evening. However, superimposed on this curve are sharp peaks of short duration but of very high demand (Figure 1). These short-duration high-demand peaks occur predictably at certain times each day. They result from such things as an industrial process kicking in at the same time each day or people returning from work to begin cooking dinner. As a result, the electrical energy supplier must charge the user for building and maintaining the facilities capable of exceeding these peak demands even though they last for a relatively short time (one to three hours) [1]. Although a peak may only last for one hour the electrical energy supplier will charge the user an entire month at the rate attained by the highest peak in that month [2]. To considerably reduce their electrical energy expenses there is great incentive by electrical energy users who have short-duration high-power peaks to “shave” them.

The idea to shave peaks is not new. A related concept known as “load leveling” was developed shortly after dynamos were invented and installed in central electric power plants in the 1880’s. In load leveling, batteries are charged at night during low power demand and discharged throughout the entire day, adding to the normal dynamo output [3]. In modern

Figure 1: A representative electrical energy demand curve. The sharp peak is “shaved” using electrical energy stored in a lead-acid battery recharged during low demand periods.

times, the use of large lead-acid batteries is under study as one possible solution for peak shaving [4]. The Electric Power Research Institute (EPRI), Palo Alto, California, in conjunction with the Department of Energy (DOE), have created a facility to research such batteries called the Battery Energy Storage Test (BEST) facility. It is located in Newark, New Jersey and is operated by the Public Service Electric and Gas Company.

1.1.2 The EPRI Battery: Purpose and History

This study concerns itself with a lead-acid battery manufactured by GNB, Inc. (Kaukaee, Illinois) in June, 1983. The 340-cell battery, fabricated according to EPRI specifications, was capable of delivering 500 kW for a one hour discharge (1040 Ah cell capacity limit) or 1.2 MWh for a five hour discharge (2080 Ah cell capacity limit) [5]. The cells of this battery were produced and tested in five batches: four batches of 80 cells each and a fifth batch of 20 cells. Each cell was numbered with the batches labeled “circuits” 1 through 5. Detailed records of fabrication materials and measurements were made for each cell. The battery was then installed at the BEST facility in 54 six-cell modules (comprising 324 of the 340 manufactured cells). The performance of over 200 charge/discharge cycles was observed after completion of the initial acceptance tests in December 1983. In July 1987, the battery was installed at Crescent Electric Membership Corporation (CEMC), Statesville, North Carolina (an area electric power distributor) for peak shaving. It has since operated at a maximum discharge of 500 kW for one hour and a minimum discharge of 200 kW for three hours.

Quarterly maintenance data have been recorded since the battery was installed at CEMC. Statistically representative sets of cells were chosen for capacity tests conducted in March 1989 (109 cells) and April 1990 (121 cells). Then, in September 1991, capacity tests were performed on all 324 cells.

A plot of the number of cells with a given capacity versus the cell capacity value shows a Gaussian-like distribution. Figures 2 shows the distribution of cell capacities from capacity

Figure 2: The distribution of capacity among the cells at CEMC from the combined 1989, 1990, and 1991 capacity tests.

tests done on EPRI's GNB battery in 1989, 1990, and 1991. It is desirable to have all of the cells of a battery to be of the same capacity rating to prevent low capacity cells from going into reversal, the state in which the electrodes reverse polarity during deep discharge [6].

Hence, EPRI's GNB battery was studied at all stages of its life to determine if various features from its initial fabrication and routine maintenance could predict which cells would be of low, medium, and high capacity [5, 7-16].

The capacities of lead-acid batteries are dependent upon their age [17]. Figure 3 illustrates a typical plot of percent of rated capacity available versus percent of life used for a lead-acid battery. Early in the life of a lead-acid battery the capacity rating rises above the nominal [18] and stays at this level, only gradually declining as the battery ages. At a certain stage in the life of the battery the capacity begins to sharply drop until a rating of 80% is

Figure 3: The capacity of a lead-acid battery depends upon its age. Capacity decreases rapidly towards the end of a battery's life. When a capacity rating of 80% is reached the battery is considered due for replacement..

reached. At that point the battery is due for replacement [19]. Individual cells within a battery will reach this point of sharp decline earlier than others. Since EPRI's GNB battery has not yet reached this end stage, predicting time of cell death is beyond the reach of this study. However, the results of this study and its predecessors [5, 7-16] will be of value to any future investigations delving into that area.

Through the study of the life of this battery it is hoped that ultimately the physical and chemical parameters responsible for the capacity of lead-acid cells can be better understood. With this knowledge lead-acid batteries can be uniquely constructed to meet specific electrical energy needs.

1.1.3 The Chemistry of Lead-Acid Batteries

Lead-acid cells, developed by Gaston Planté in 1859, were the first practical secondary (rechargeable) cells. Since, they have become the workhorse of secondary batteries as evidenced by them being the most widely used secondary battery in the world [20]. This is due to their readily available low cost raw materials (lead and sulphur), long life, ruggedness, and convenient reversible electrochemical reaction [20, 21].

The lead-acid cell reaction consists of converting chemical energy to electrical energy during discharge and electrical energy to chemical energy during charge. The lead-acid cell consists of one electrode of lead and another of lead dioxide both immersed in a sulfuric acid electrolyte. Oxidation/reduction at the electrodes drives the battery. The cell can be depicted as:

At the positive electrode (cathode), the half cell reaction is:

At the negative electrode (anode), the half cell reaction is:

The full cell reaction is

Discharge is the thermodynamically favored direction for the reactions. At the positive

electrode, the lead dioxide is reduced by the sulfuric acid to produce lead sulfate on the electrode surface and increase the water concentration in the electrolyte; two electron equivalents are required for this reaction. At the negative electrode, the lead is oxidized to lead sulfate by the sulfuric acid; two electron equivalents are produced in this reaction. Connecting the two electrodes via an electrical conductor allows the reaction to proceed: the two electron equivalents required at the positive electrode are supplied by those produced at the negative electrode.

This reaction produces 2 V regardless of the size of the cell [22]. However, as the discharge proceeds, the voltage decreases until a cutoff voltage (typically 1.7 V) is reached. The actual cutoff value varies depending on the design of the cell. For cells connected in series the voltage is additive, with the capacity of the entire battery being the same as that of an individual cell. If the cells are connected in parallel the voltage does not increase but the capacity of the battery increases. For EPRI's GNB battery all of the cells were connected in series, which increased the voltage but not the capacity of the battery [22].

To recharge the cell, electrons are forced in the opposite direction of their discharge flow. This is done by applying an opposite voltage higher than the discharge voltage [23]. The reactions in the cell are reversible so that the original charged state is recreated. However, the principle of the second law of thermodynamics requires that more energy be put into the battery for recharging than is gotten out during a subsequent discharge. Figure 4 illustrates the lead-acid battery charge/discharge cycle.

Figure 4: The lead-acid battery charge/discharge cycle. The longer the cell stays in the discharged state the lower its future capacity will be.

Since a major objective of this study was to determine the features that could predict cell capacities and lifetimes, it was imperative to understand the variables that cause cell failure. The possible causes of cell failure are many: the shedding of the active material from the electrodes; loss of electrolyte through evaporation or conversion of water to O_2 and H_2 (the latter may occur during recharging or from reversal); sulphatation, the process in which $PbSO_4$ on the electrodes is recrystallized into a dense, coarse grained form and inhibits charge

acceptance (this more easily occurs the longer a cell is left discharged [24]); internal electrode shorting; reversal; and the loss of ability of the cell to convert chemical energy to electrical energy and vice versa [25].

1.2 Previous Studies of Predicting and Modeling Cell Performance

The idea of using manufacturer's pre-test results for predicting cell lifetime was first investigated for nickel-cadmium cells using statistical analysis, cluster analysis, and multivariate pattern recognition techniques [26]. Later, this type of inquiry was extended to lead-acid cells [7]. Encouraging results from these studies laid the foundation for extensive investigations into EPRI's GNB battery using these same techniques as applied to fabrication and routine maintenance data [8-15].

Using fabrication and routine maintenance data for predicting cell performance was considered attractive by the first researchers for several reasons [15, 26]. Using the fabrication data could allow one to group cells for specific functions at the outset [26]. In addition, if the initial manufacturer's test data were not available, then routine maintenance data could be used in its place [15]. Routine maintenance data could also be used rather than periodically conducting expensive capacity tests [14].

1.2.1 First Exploration: Predicting Nickel-Cadmium Cell Performance Using Initial Characteristics

Prediction of extended performance and time of failure for nickel-cadmium cells using pattern recognition techniques on manufacturer's test data were investigated by Byers and Perone [26]. K-nearest neighbor analysis and a modified fixed neighborhood classification rule

(a nonparametric hierarchical clustering algorithm) were used. The cells (220 total) were grouped in packs of five to eight. Each pack was cycled and artificially aged under different conditions. Good and bad cells were arbitrarily identified by apparent breaks in the lifetime distribution curve. The result was that the k-nearest neighbor technique could give at most 75% classification accuracy when all of the cells were looked at together with any particular set of features. However, analysis of individual groups of cells that had similar composition gave an overall classification accuracy of 90.7%. Individually examined, the good cells had a classification accuracy of 93.3% and the bad cells had a classification accuracy of 84.6%. This dependence on similar material composition was further demonstrated when clustering was found to occur by manufacturer's lot number. The authors concluded that detailed knowledge of cell construction would probably be required to fully predict cell lifetime. Nonetheless, the results of this study were quite encouraging and gave impetus for further investigations in this area.

1.2.2 Extending the Concept to Predicting Lead-Acid Cell Performance Using Initial Characteristics

An attempt to apply the same type of analysis that had been done on the nickel-cadmium cells to a group of 108 lead-acid golf-cart cells was performed by Perone and Spindler [7, 9]. In their study, linear discriminant and k-nearest neighbor analysis was used. As in the nickel-cadmium cell study, good and bad cells were arbitrarily divided by apparent breaks in the cell failure distribution. The arbitrary class boundaries gave coarse class distinctions. However, after pattern classification was done, the class boundaries were

adjusted to effect a fine tuning to maximize classification accuracy. Preliminary examination and initial acceptance test data produced 10 features that were found to be useful. These features fell into such categories as initial specific gravity, initial water volume added, initial capacity trends, and transformed/combined variables. Two different depth of discharge sets were examined (50% and 75%). The linear discriminant analysis produced the highest average accuracy for all data sub-sets. For short lived batteries in the 50% depth of discharge data sub-set, a classification accuracy of nearly 100% was achieved. The k-nearest neighbor analysis produced the overall best results of 85.5% classification accuracy when the 50% and 75% discharge sets were grouped together.

1.2.3 Investigations Into Predicting Cell Performance for EPRI's GNB Lead-Acid Battery

The Perone and Spindler [7,9] study was important not only because it extended the cell lifetime prediction analysis into lead-acid batteries, but also because it was the foundation for the fabrication and test plan for EPRI's GNB battery [9]. The encouraging outcome of Perone's and Spindler's study opened the way for pattern recognition techniques to be applied to EPRI's GNB battery. Data were collected and analyzed at every stage in the battery's life [7-16]. Three separate studies endeavored to determine whether cell performance could be predicted using different components of that data. The first study explored the use of the fabrication/test data in the same manner as was done by Perone and Spindler [12]. The second study investigated the use of routine maintenance data [13]. And the third study examined the use of fabrication/test and maintenance data together [16].

1.2.3.1 Investigation in the Use of EPRI's GNB Battery Fabrication/Test Data for Cell Performance Prediction

The fabrication/test data of EPRI's GNB battery were examined for features that could predict cell performance. The findings were presented by Petesch in a Master of Science report [12] and published by Perone et al. [14]. The study consisted of two parts. The first part was to determine whether accurate classifications could be made with two classes (low and high performance cells). Accuracy was determined using leave-one-out k-nearest neighbor analysis. An overall accuracy of 92% was achieved on the training set using several different feature sets. Non-linear mapping was used to determine which feature sets, though providing accurate classifications, also produced the best separations in hyper-space. Those features that gave good results with both k-nearest neighbor analysis and non-linear mapping were considered to contain predictive information. A prediction set was not analyzed.

In the second part of the study the cells were classified into three classes (low, medium, and high performance cells). The assumption was made that class divisions could be made based on the data having a Gaussian distribution and classes were established with normalized mean and standard deviation units (the mean was set at zero and standard deviation was set at one). Cells with capacity within one standard deviation of the average were defined as a medium performance class; cells with capacity greater than the average plus one standard deviation were defined as a high performance class; and cells with capacity lower than the average minus one standard deviation were defined as a low performance class. Because of this oversimplified class definition guideline some erroneous classifications were encountered.

As a result, non-linear mapping was used to re-classify some cells. For the re-classified training set the overall training accuracy achieved with various combinations of features ranged from 78% to 86%. For the individual classes the highest accuracy was 92% for the low, 81% for the medium, and 85% for the high performance cells. As in the previous battery classification studies, the maximum classifications were achieved with different feature sets for each class. For the prediction set only a maximum accuracy of ~48% was achieved.

Petesich and Perone concluded that, through multivariate analysis, one could extract out information related to cell performance from the initial fabrication data that proved important for potentially predicting cell performance - even seven years after manufacture. Petesich further concluded that, for some cells, other factors besides the fabrication parameters were responsible for poor cell performance.

1.2.3.2 Investigation in the Use of EPRI's GNB Battery Routine Maintenance Data for Cell Performance Prediction

The routine maintenance data were examined for features that could predict cell performance. The findings were published by Chen in a Master of Science report [13] and by Perone et al. in two journal publications [14, 15]. Like the fabrication/test study done by Petesich, Chen's study was also done with two classification schemes: classification into two classes (low and high) and three classes (low, medium, and high). Class divisions were defined as in Petesich's study, by normalized standard deviations. Chen's study demonstrated the predictive power of the maintenance data using data indexing. Several indexes were tried, but a "battery activity" index proved the most useful. This index was based on the volume of

water required to be added to each cell each quarter for one year prior to a capacity test. The amount of water required was determined to reflect the total activity of the cell in relation to the charge/discharge cycle. For the two-class training set, several features gave 100% overall accuracy. For the prediction set, several sets of three to five features gave around 80% overall accuracy. For the three class training sets, the four best features gave 76% - 88% overall accuracy with the high performance cells providing nearly 100% accuracy. However, no meaningful results were realized with the prediction set.

1.2.3.3 Investigation in the Use of EPRI's GNB Battery Combined Fabrication/Test and Maintenance Data for Cell Performance Prediction

By the time the previous studies were being concluded it was understood from the classification accuracies achieved with prediction sets that everything was not being taken into account. Though it was apparent that fabrication/test data and routine maintenance data contained information useful in predicting cell performance, it was also apparent that, individually, they were inadequate to make practical predictions when using the k-nearest neighbor and the non-linear mapping techniques. Classifications into three classes particularly seemed to be a problem. A study was, therefore, performed to explore the simultaneous use of the fabrication/test data set and the routine maintenance data. The results were presented by Li in a Master of Science report [16].

Three classes were looked at in Li's study (low, medium, and high performance cells). Class boundaries were defined, just as they were in Petesch's and Chen's studies, by normalized standard deviations. K-nearest neighbor was used for developing classification

clusters, and non-linear mapping was used for fine tuning the classifications. The leave-one-out k-nearest neighbor technique was used for determining classification accuracy. Two sets of training and prediction sets were created from the maintenance data as follows,

<u>Training Set</u>	<u>Prediction Set</u>
prior to March 1989 capacity test	prior to April 1990 capacity test
prior to April 1990 capacity test	prior to September 1991 capacity test

The combined fabrication/maintenance data yielded an overall classification accuracy of 60%. However, the major breakthrough came with the reworking of the three class problem. Three separate classifications were done on each individual class with the other two classes bundled as its logical complement. That is, the three class problem was redefined as a series of three two-class problems: low versus medium/high; medium versus low/high; and high versus low/medium. This adaptation allowed higher individual classification accuracies to be achieved than was possible before on prediction sets. Maximum classification accuracies were 85% for the low performance cells, 65% for the medium performance cells, and 92% for the high performance cells. However, these maximum accuracies were not achieved when using the same feature sets. That is, the classification accuracies of the other two classes associated with these maxima were not necessarily high. This is reflected in the maximum overall classification accuracy of 73.8%. Additionally, the same features did not yield the best results in the various training/prediction combinations.

1.2.4 Theoretical Modeling of Cell Reversals in Long String Lead-Acid Batteries

A theoretical study for predicting cell reversals in long battery strings was conducted by Perone and Symons [6]. A computer simulated a battery with parameters affecting cell reversals: discharge profile, string length, capacity, capacity variance, and cutoff voltage. This study was conducted in two phases. Phase one involved exaggerating the frequency of cell reversals with a sharp decay discharge profile to provide statistically significant data. Phase two constituted running the simulation under a more realistic discharge profile for lead-acid batteries. Their study concluded that there were two parameters likely to cause cell reversals. One was having large variances in cell capacities and the other was having a bimodal distribution of cell capacities. String length and average cell capacities were found not to be significant variables. In the light of their conclusions, EPRI's GNB battery was assessed for the probability of cell reversals. It was deemed unlikely that the battery would have serious cell reversal problems.

1.3 Neural Networks

A neural network is a mathematical modeling procedure originally thought of as mimicking the operation of neurons in the human brain [27]. Like the biological neurons, the mathematical neurons process data in parallel, are highly interconnected, and are activated or not based on their relative inputs and outputs to and from one another. Most importantly, they both "learn" by exposure to examples. Mathematically, a neural network allows a mapping to occur between two sets for purposes of classifying, predicting, or other specialized processing (such as signal analysis). It falls in the same category as other multivariate techniques such as

linear discriminant, k-nearest neighbor, machine learning, and statistical least squares techniques [28]. However, neural networks have more capabilities than any of the other techniques. They are nonlinear [29], provide more functional forms [29], and are nonparametric [27]. Since they are nonparametric, assumptions about the data fitting a particular density function are not made [28].

A neural network gains its pattern recognition aptitude by encoding patterns into the activation levels of a system of parallel distributed information processors [31, 32]. This is one of the major advantages of using a neural network. That is, rather than devising complicated programs, one merely presents the neural network with plentiful and representative examples, and it learns from them [31]. Also, in circumstances where theoretical, analytical, or numeric solutions are inadequate (e.g., the relationships between features are unknown), a neural network may be able to associate many obscurely interrelated variables into a usable multi-dimensional mapping [33, 34]. From the input data the neural network will extract its own model [34].

A major disadvantage that neural networks have, which many multivariate statistical techniques do not, is interpretation. In least squares analysis, for example, the slope, intercept, and sign of the correlation coefficient may all have interpretative significance. However, for a neural network, the interpretative significance of the interconnections between nodes and the associated matrix of weights is quite abstract and difficult to explain [35].

1.3.1 A Short History of Neural Networks

Developing computers became of great interest during World War II with the urgent need for such things as code breaking, trajectory determinations, and the diverse and complex calculations necessary for the Manhattan Project. In 1943, McCulloch and Pitts published a paper in which the theoretical foundations for neural networks were presented [27]. This paper suggested that any mathematical or logistic function could be modeled by a parallel array of processors. However, nothing practical was done with the theory at that time.

In 1949, a neuron learning rule was devised by Hebb of McGill University [36]. In 1951, Minsky of the Massachusetts Institute of Technology built the first neurocomputer [37]. His computer was not applied to any task but just demonstrated the neural network principle. Then in 1957, Rosenblatt of Cornell Aeronautical Laboratories built the first neurocomputer applied to an actual pattern recognition task. His design, called the Mark I Perceptron, was trained to recognize visual patterns of letters using his perceptron algorithm [38]. Rosenblatt believed that his perceptrons could significantly outperform digital computers [39]. A major blow to neural networks came in 1969 with the publication of *Perceptrons* by Minsky and Papert [40]. Their book has since become a historical landmark in neural computing. Therein, the deficiencies of perceptrons, as understood and implemented by Rosenblatt, were demonstrated. Many predicate functions (two state output variable statements dependant upon their independent variables), such as the exclusive-or (XOR) function, could not be represented by perceptrons. This was because the perceptron can only create linearly separable decision regions; the XOR function consists of two separate regions that belong to

the same class (see Figure 5). This finding essentially discouraged research into neural networks throughout the 1970's and sent research money elsewhere. Minsky said in a 1981 interview that *Perceptrons* was “overkill” because of his annoyance at Rosenblatt's arrogant claims [41].

Things changed for the better in 1983 when Hopfield of the California Institute of Technology proposed that a nonlinear relationship between the total input received and the output produced by the neurons could overcome some of the limitations of the perceptrons [33]. Hopfield, a physicist, approached neural networks from a different perspective than the computer scientists and mathematicians that had been the predominant researchers. He interjected an energy minimization concept from physics into neural networks and made them theoretically useful [42]. This catalyzed a renewed interest in neural networks. The nonlinear relationship is the foundation of all current theoretical neural network development [43].

In 1986, the back propagation learning rule, which removed still more of the perceptron's limitations, was fully developed by Rumelhart of the University of California at San Diego and others [44]. Though back propagation was first suggested in 1974 by Paul Werbos, who completed his Ph.D. dissertation in 1980 on this topic [45], it was not until Rumelhart that back propagation was made into a practical technique.

Neural networks have since been applied to such diverse fields as predicting the stock market [46], determining credit fraud [46], and medical diagnoses [47]. In chemistry it has been applied to classifying spectra (MS, NMR, IR, near-IR, optical reflectance, UV, and X-ray fluorescence), predicting secondary and tertiary protein structures, process control, and

Figure 5: No single line can draw a boundary to separate class A from B. Rosenblatt's perceptron could not have noncontiguous decision regions as is depicted here with the shaded and unshaded regions.

chromatographic peak classification [33, 48]. Other topics include predicting a compound's solubility in supercritical carbon dioxide [49], drug discovery [50], and chemical bond reactivities [51]. A 1993 study showed that nearly 90% of publications of neural network applications to chemistry used back propagation [52].

Currently a great deal of theoretical research endeavors to extend the power of neural networks. One area of interest probes the use of fuzzy sets in neural networks [53, 54]. Another field couples neural networks with expert systems to let expert systems acquire new knowledge rules [55].

1.3.2 Back Propagation Neural Networks

1.3.2.1 Phases of Developing a Back Propagation Neural Network

Neural network development entails three distinct phases: learning, recall, and testing. During the learning phase, the neural network is trained by being presented with a series of examples in which the input features have known classifications. The network adjusts itself continuously to minimize the differences between the neural network's output and the known classifications. When an individual case has been submitted to the neural network, this is termed a "training cycle." When the network adjusts itself to minimize error, this is called an "epoch." An epoch can occur after every training cycle or after several training cycles to get a net error adjustment of several cases [56]. The number of training cycles necessary to fully train a network may vary from a few hundred to many hundreds-of-thousands [57, 58].

After the network has been tentatively trained, the recall phase can be entered. During the recall phase the same examples with which the network was trained are again presented. However, the network does not adjust itself, but simply generates an output, which the user compares with the desired outcomes. In this way, one can tell how well the network has learned the patterns on which it has been trained.

After the training and recall phases have concluded, the testing phase can commence. In the testing phase the network is presented with patterns it has not encountered before, though they are of the same origin as the training set, to determine how well the network can interpolate patterns it has not seen before [59]. Once a satisfactory neural network has been developed it can be considered for deployment on unknown patterns.

1.3.2.2 Structure of a Back Propagation Neural Network

A neural network consists of a number of distinct layers each with various numbers of mathematical neurons. The layers are also called “slabs” [60] though “layer” seems to be the more accepted nomenclature. The neurons are interchangeably known as processing elements, nodes, or units. At least one author reserves the term “units” for only the input nodes to emphasize their functional difference from all other nodes in a network [61]. The overall structure of a neural network is illustrated in Figure 6 and an individual processing element is portrayed in Figure 7.

First, in neural network architecture, there is an input layer. The number of processing elements in this layer corresponds to the number of inputs. Each processing element in this layer receives only one input from outside of the network. The input may be preprocessed by the computer before it reaches this layer to ensure that all of the inputs are normalized to fit within a certain range such as $[0,1]$ or $[-1,+1]$ so that the system does not become “saturated” with large numbers [62]. Each node in the input layer fans out its input, without modification, to each processing element in the next layer [63]. Each transfer of output from one neuron to the input of another neuron is called a connection [64]. The next layer is called a hidden layer. There may be one or several hidden layers. Each processing element in the first hidden layer receives input from each input layer node. Each input in each processing element is multiplied by a separate weight factor. These products are then summed. The product of a scaling factor and a bias factor are then added. This operation, called the internal activation of the processing element, is presented in Equation (1),

Figure 6: The structure of a feed forward back propagation neural network consisting of three layers (the input layer is not counted when counting layers in a neural network).

Figure 7: The structure of an individual processing element. The inputs (x) are multiplied by weight factors (w), summed, transformed with a transfer function, and then distributed to other processing elements.

$$\text{internal activation} = \sum(x_i \times w_{ij}) + (\theta_j \times B) \quad (1)$$

where x_i is the i^{th} input, and w_{ij} is the corresponding i^{th} weight factor for the j^{th} processing element, θ_j is the scaling factor, which adjusts the output of the j^{th} processing element to fall within a useful range, and B is the bias (always equal to one) and acts as a “ground” input to the entire network [65, 66]. The bias is analogous to an intercept in the equation of a line ($y=mx+b$). Without the bias, all decision regions would be restricted in such a way that would make even simple decision regions impossible [67]. It is analogous to trying to separate two groups of points from one another on a Cartesian plane with the requirement that the separating line always be forced through the origin.

The internal activation is then fed through a transfer function (called a squashing function by some) that effects a nonlinear transformation. The transfer function can be anything that meets two criteria: the output must be confined to an interval such as $[0,1]$ or $[-1,1]$ and vary monotonically at the extremes of the input ranges (i.e., the output never decreases or increases as the absolute value of the independent variable increases) [43, 68]. Two such functions commonly used in neural networks are the sigmoid function, Equation (2), for the interval $[0,1]$ and the hyperbolic tangent function, Equation (3), for the interval $[-1,1]$,

$$\text{Sigmoid} = \frac{1}{1 + e^{-a}} \quad (2)$$

$$\text{TanH} = \frac{e^a - e^{-a}}{e^a + e^{-a}} \quad (3)$$

where the variable, a , is the internal activation. These functions are used because their derivatives are very easily represented in terms of the original function. This property makes them useful in many learning algorithms [69].

Alternative transfer functions can be used in special circumstances. For example, when a frequency component exists in the data a sine function can be used. The sine function provides a “generalized Fourier analysis” [70]. Also threshold logic functions have been used when a single input is used for multiple classifications [33]. However, threshold logic functions may contain singularities which makes them theoretically less useful [33, 71].

The internal structure of some processing elements may be more complex. All nodes have local memories to retain weight factors. However, some architectures may have more memories to contain other variables. One such variable is called a “conscience” and keeps track of how many times a neuron has won in “competitive learning” so that it can be taken out of competition for awhile to give other neurons a chance to learn [72].

The output from each node’s transfer function is then sent to each node in the next hidden layer - or to the output layer if there are no more hidden layers. At that point the same mechanism reoccurs: at each node the internal activations are determined, taken through a transfer function, and then output. For classification problems the number of output layer nodes usually correspond to the number of classes that the user is attempting to classify using

the input features, though this is not always the case.

The structure described here is a “fully connected feed forward” design. It is fully connected in that each node of a layer is connected to all of the nodes of the next layer. It is feed forward such that all of the connections go from one layer to the next without any connections feeding back into the same layer or previous layers. Several neural networks have outputs that go into the same or previous layers, and may have some nodes that skip output to one layer to feed into another layer or connect to several different layers and not just the adjacent ones. Additionally, some nodes may not receive input from every node in the previous layer [59, 73].

1.3.2.3 Training: Supervised Training of a Back Propagation Neural Network

A network learns by adjusting the weights in each processing element throughout the network. Initially, the weights are set to small random numbers such that the sum of all of the weights throughout the network equals one [74]. Through presentation of a training set of data the weights are modified. The training set consists of k data vector pairs $(\mathbf{X}_k, \mathbf{Y}_k)$ where $\mathbf{X}_k$ is a set of input features and $\mathbf{Y}_k$ is the desired output which is associated with the $\mathbf{X}_k$. The neural network produces a set of results $\mathbf{Y}_k'$. During the learning phase, the output layer's results ($\mathbf{Y}_k'$) are compared to the expected output ($\mathbf{Y}_k$) of the training set. This process is depicted in Figure 8. This comparison results in an adjustment of the weight factors (including the θ scaling factor) to minimize the difference between $\mathbf{Y}_k$ and $\mathbf{Y}_k'$. The revision of the weight factors can occur after each training cycle or with epochs of varying lengths [75]. The cases are presented in a random order to attain a more generalized training [76, 77].

Figure 8: Supervised training of a back propagation neural network. Weights are modified to minimize the differences between the network's determination of classes (Y') and their actual classes (Y) based on the input of features (X).

Error minimization can be accomplished by using the “delta rule.” First, an estimation of the derivative of the error surface generated by the differences between Y_k and Y_k' is done. These derivatives are then used to modify the weights. For the output layer’s processing elements the error derivative is calculated by,

$$(\text{error derivative})_j = \text{Out}_j \times (1 - \text{Out}_j) \times (\text{Corr}_j - \text{Out}_j) \quad (4)$$

where Out_j is the output of node j and Corr_j is the correct answer that node j should have. This is a straightforward calculation based on the knowledge of the actual error [78]. For a hidden layer’s processing elements the error derivative is estimated because, unlike the output nodes, their optimal outputs are unknown. Therefore, an assumption that error is diffused over all of the weights is made [79]. The error derivative for the hidden layers is then calculated,

$$(\text{error derivative})_j = \text{Out}_j \times (1 - \text{Out}_j) \times (\text{Corr}_j - \text{Out}_j) \times [\sum (\epsilon_k \times w_{jk})] \quad (5)$$

where ϵ_k is the error derivative of the k node connected to node j in the next higher layer [80].

Once the error derivatives have been determined a delta factor can be calculated to adjust the weights,

$$\Delta w_{ij} = \lambda \times (\text{error derivative})_j \text{Out}_i \quad (6)$$

and for the scaling factor θ ,

$$\Delta\theta_j = \lambda \times (\text{error derivative})_j \quad (7)$$

where λ is the learning rate coefficient. The learning rate coefficient allows divergent sections of the error surface to be traversed at any one time [81]. This is necessary due to the nature of back propagation error surfaces, which typically have considerable areas of flat or near zero slope in several dimensions at once and a theoretically large number of both local and global minima (see Figure 9) [82]. Large sections may then be traversed by making the learning coefficient larger; this makes oscillations more likely but learning is faster and the neural network is less likely to get trapped in a local minimum [76]. In one particular neural network problem a single local minimum was verified as not being global with a Cray-2 supercomputer working for 12 hours on the problem [83]. Smaller sections of the error surface may be traversed by making the learning coefficient smaller; this makes learning slower but prevents oscillations in solutions across the error surface. Further, the learning coefficient can be varied as training progresses to engage the desired learning rates when they are most useful [84]. It has also been shown that if the learning rate coefficients in the hidden layers are larger than those in the output layer then feature differentiation can be formed early in training and allow more complex decisions at the output level [85].

Additionally, a momentum term, μ , is added to further prevent oscillations in the search for a global error minima. The weights are revised by,

Figure 9: A two dimensional representation of a hyperspace error surface. For some problems an infinite number of local and global minima may exist. Though not depicted here, vast regions may be relatively flat.

$$\Delta w_{ij}^{new} = \lambda \times (\text{error derivative})_j \times \text{Out}_i + \mu \times \Delta w_{ij}^{old} \quad (8)$$

and the scaling factor θ is revised by,

$$\Delta \theta_{ij}^{new} = \lambda \times (\text{error derivative})_j + \mu \times \Delta \theta_{ij}^{old} \quad (9)$$

The μ term reinforces previous delta factors and gives “momentum” to the error surface descent trends [81]. As a general rule, the sum of the learning rate coefficient and the momentum term should be approximately equal to one [86].

1.3.2.4 Recall: Evaluation of a Trained Neural Network with the Training Set

In the second phase of developing a neural network, the training set is taken through the network but without the weights being adjusted. The output values are compared to the correct values of the training data. Poor results mean either that the network has not been properly trained, or that there is a problem with the data. If it is the former, then modifications of any of the following parameters may be necessary: number of nodes in each hidden layer; number of hidden layers; learning and momentum coefficients; transfer functions; or number of epochs. Other parameters may also need adjustment. Retraining of the network must then be done. If the problem is the latter, then the data may need some additional preprocessing; different input features may need to be found; the network may be trying to learn several things at once which interfere with one another [87]. On the other hand, obtaining good results with the training set still does not necessarily mean that the network was properly trained. The network must be evaluated with a test set.

1.3.2.5 Test: Evaluation of A Trained Neural Network With a Test Set

With sufficient classification accuracy achieved on recall of the training set, an evaluation of the test set can ensue. Here the network is presented with patterns of the same origin as the training set which were not encountered during training. This procedure determines how well the network can interpolate to patterns it has not seen before. Obtaining good results at this point indicates that the neural network training was headed in the right direction, and slight modifications in the successful neural network could perhaps lead to better results with retraining. Finally, after the network's performance has met the desired criteria of

success defined by the user, it is deployed with real world data where the outcomes are unknown.

1.3.3 Other Neural Networks

There are several neural network designs other than back propagation. Some have avoided the local minima problem that is the bane of back propagation. Others have developed learning algorithms that are incredibly fast [88]. Some neural networks have been designed to excel in one area. Back propagation has been popular because it is a good general purpose neural network.. Even back propagation has been modified with different learning rules to increase its power. One learning rule is the “extended delta bar delta” rule. This modification of the delta learning rule presented earlier automatically adjusts the learning rates and momentum factors throughout the learning process [89].

Some of the common neural networks are briefly described here. A Hopfield network is used for recognizing a pattern from “partial or corrupted” data [90]. The Adaptive Bidirectional Associative Memory network has a similar purpose as the Hopfield but is implemented differently [91]. Self Organizing Map Networks do unsupervised learning to map data of a higher dimension to a lower dimension. In this way nonlinear mapping analysis can be done [92, 93]. Spatiotemporal networks were developed to handle time dependant input and output data [94, 95]. Fuzzy ARTMAP incorporates fuzzy sets into a neural network. Logicon Projection forms decision regions as hyper-spheres [96].

Chapter 2 - Experimental

2.1 Instrumentation and Software

An IBM/PC compatible 486-DX2/66MHz computer with 8 MByte RAM, 512 KByte SRAM cache, and 550 KByte hard disk drive were used for development of the neural networks, database management, and creating diagrams and illustrations.

Excel® (ver. 5.0; Microsoft Corp., Redmond, WA) was used to manage the databases. Management of the databases included importing files previously created using SYMPHONY™ (Lotus Corp, Cambridge, MA), reorganizing and editing the data, preprocessing of the data using statistical functions, and exporting the data in a format compatible with the neural network software. The neural networks were developed using NeuralWorks Professional II/PLUS (ver. 5.0; NeuralWare, Inc., Pittsburgh, PA). Diagrams and illustrations were created using TurboCAD® for Windows™ (ver. 2.0; IMSI, San Rafael, CA). All software was executed under Windows™ (ver. 3.11; Microsoft Corp.) with MS-DOS® (ver. 6.2; Microsoft Corp.).

2.2 Database

2.2.1 Raw Database

The raw database contained all of the unprocessed maintenance data. This included quarterly maintenance task data (float voltages, specific gravities, water additions, and electrolyte levels) for each of the 324 cells from August, 1987 through September, 1991 and the results of all of the capacity tests done in March, 1989, April, 1990, and September, 1991.

Capacity test data consisted of the results of capacity tests for 109 cells from March, 1989, 121 cells from April, 1990, and 323 cells from September 1991.

2.2.2 Feature Database

A feature database contains features which may prove useful in classifying and is derived from the raw database. These features may include transformations and combinations of the elements found in the raw database. For example, prior to a capacity test several quarters of float voltage readings were taken for each cell. The combination of several float voltages, transformed through linear regression, form a slope and a correlation coefficient which may demonstrate a trend. The slope and correlation coefficient would hence be elements in the feature database.

The feature database also contained class assignments associated with each feature pattern. During the training phase the class assignment data fields are used for supervised training. During the recall and test phases the class assignments are used to gauge the performance of a neural network by comparing the patterns' actual classes to the neural network's class assignment of them. Hence only those cells associated with a capacity test event were retained for the feature database.

2.2.2.1 Data Preprocessing

Inherent in the raw data are major variations in magnitude and scale. This included variations from feature to feature. In order that each feature might be used in conjunction with others, these incidental magnitude and scale properties had to be removed. To eliminate these differences, the features can be transformed via normalization according to Equation (10),

$$(XN_j)_i = (X_j)_i / (X_{\text{mean}})_j \quad (10)$$

where for the j^{th} feature of the i^{th} cell, $(XN_j)_i$ is the normalized value of the raw data $(X_j)_i$. $(X_{\text{mean}})_j$ is the mean value of the j^{th} feature for all cells of a particular maintenance event. This form of normalization eliminates the differences in magnitudes between features yet retains the relative differences in ranges within a feature.

Data obtained for a particular feature from different times or from different origins may contain variations which make combining and comparing them problematic. To eliminate the magnitudes of the standard deviations between features, each of the features can be autoscaled using,

$$(XS_j)_i = [(XN_j)_i - (XN_{\text{mean}})_j] / (XNSD)_j \quad (11)$$

where for the j^{th} feature of the i^{th} cell, $(XS_j)_i$ is the autoscaled value of the normalized data $(XN_j)_i$ generated in Equation (10). The normalized mean value, $(XN_{\text{mean}})_j$, is from Equation (10), and $(XNSD)_j$ is the sample standard deviation of the normalized j^{th} feature for all cells of a particular maintenance event.

2.2.2.2 Definition of Class Boundaries

Cell capacity, expressed as a percentage of the nominal value (2080 Ah), was the figure of merit used to distinguish how well a cell performs. The assumption was made that class divisions could be made based on their having a Gaussian distribution. Classes were

established based on their normalized mean and standard deviation units (the mean was at zero and standard deviation set at one). Cells with capacity within one standard deviation of the average were defined as a medium performance class; cells with capacity greater than the average plus one standard deviation were defined as a high performance class; and cells with capacity lower than the average minus one standard deviation were defined as a low performance class. A class assignment must be associated with each pattern in the feature database for supervised learning (see section 1.3.2.3) and thus must be available during the recall and testing phases so that neural network performance can be assessed (see sections 1.3.2.4 and 1.3.2.5).

2.2.2.3 Data Sampling

Unlike classical statistical methods in which the number of samples necessary to obtain statistically significant results can be determined, in neural networks this number can only be estimated [97]. Neural networks typically need hundreds to thousands of examples in the training set to justify their use [57, 58]. In many instances the number of examples required is dependent upon the type of data. In behavioral models, such as predicting stock market trends, the number of data pairs must be fairly large. However, for physical models, such as chemical processes, the data sets can be smaller [98]. Even for behavioral models with small data sets neural networks can be justified as long as the training data set is relatively free of noisy data or idiosyncratic examples. The fewer the number of cases in the training set the less eccentricity is acceptable [99]. So it is typically essential to use as many patterns as possible for a neural network study to be successful.

During training, if the number of representatives for one class is overwhelmingly larger than any other class then the algorithm classifies everything as the class with the most representatives. It does this because, in this situation, error is greatly reduced by classifying everything as the one with the most representatives.

2.2.2.4 Training and Test Sets

The choice of a training set is probably the most important aspects for success of a neural network project. The training set needs to be as large as possible so that an adequate number of patterns can be represented and learned [33]. In this way the data is more uniform and less susceptible to outlier patterns disrupting the learning process. Generally, all of the data is combined and split with 75-80% of the combined data forming the training set and the remaining 20-25% forming the test set [100].

2.3 Feature Selection

For neural networks highly correlated input features are not troublesome [101]. However, as with statistical methods, deletion of variables with insignificant consequence on the output improves the effectiveness of the modeling [102]. Hence it is advantageous to reduce the number of inputs to optimize neural network performance.

A Hinton diagram, named after Geoffrey Hinton who co-introduced the back propagation algorithm [44], allows one to ascertain which inputs have relatively small contributions to the neural network's solution. A Hinton diagram arranges all of the nodes in the network along both the x and y axis. At the intersection of the nodes is a graphical representation of the magnitude of the weight between the output from the node indicated on

the x-coordinate and the input to the node on the y-coordinate. The magnitudes of the weights are indicated by the size and color of a box at the x,y-intersection. The colors are defined by the user so that various ranges can be represented. The larger the box of a particular color the larger the magnitude of the weight within the defined range. Positive and negative values are indicated by solid or outline boxes. If the weights associated with a particular input node are all small then that input, relative to the other inputs, has little impact on the solution obtained by the network [103]. Figure 10 illustrates a Hinton diagram.

Feature selection can be done by training a neural network to a satisfactory level with many features input, examining the Hinton diagram, and then eliminating those features associated with inputs that add nothing to the solution. The network is then re-trained with the remaining features. The elimination process is repeated until the performance of the neural network begins to deteriorate. In this way the strongest features are retained and the performance of the network can be optimized.

2.4 Neural Network Supervised Training

2.4.1 Measuring Neural Network Performance

In order to judge the performance of a neural network one must be able to quantify its performance such that it can be compared to other neural networks and other classification techniques. Calculating classification accuracies, constructing confusion matrices, and determining risk factors are tools that provide different types of information about the classification performance of a classifier. Taken together they form a good picture of a classifier's capabilities.

Figure 10: A Hinton diagram. Along the bottom of the diagram are listed the nodes which send outputs. The left column list the nodes receiving the outputs. At the intersection of the row and column is a graphical representation of the magnitude of the connecting weights. In this illustration input 2 (I2) is a candidate for elimination.

2.4.1.1 Classification Accuracy

A classification accuracy quantifies how well a classifier correctly identifies the actual class. It can be calculated either as an overall accuracy or as a class specific accuracy,

$$\text{Overall Classification Accuracy} = (C/C_o) \times 100\% \quad (12)$$

$$\text{Class Specific Classification Accuracy} = (C_n/C_m) \times 100\% \quad (13)$$

where C is the number of correct classifications in the whole set C_i of classes, and C_n is the number of correctly classified cases of class C_m .

2.4.1.2 Confusion Matrix

A more powerful tool for evaluating the performance of a classifier is a confusion matrix. A confusion matrix provides much more information than the classification accuracies of Equations (12) and (13) because it not only conveys the percentages of correct classifications but also the types and percentages of misclassifications. A confusion matrix is a table consisting of column headers indicating the true class assignments and row labels indicating the neural network class assignments. At the intersection of a row and column is the percentage of cases of a particular class that the neural network assigned to the class corresponding to that row. In the diagonal running from top left to bottom right are the class specific classification accuracies of Equation (13). The other positions on the table indicate false positives. For example, Figure 11 depicts a confusion matrix in which 86% of true class 1 cases were correctly classified, but 7% of the class 2 cases and 22 % of the class 3 cases were misclassified as class 1. Ideally then, if a neural network classified all of the cases correctly, there would be 100's running in a diagonal from top left to bottom right of the table with zeros everywhere else.

2.4.1.3 Risk Weighted Classification Accuracy

Because some misclassifications are considered more significant than others a quantitative “risk” factor can be assigned to each level of significance of correct and incorrect classification. Correct classifications are given positive values and misclassifications are given

Neural Network Identified Class	True Class		
	Class 1	Class 2	Class 3
Class 1	86	7	22
Class 2	4	90	3
Class 3	10	3	75

Figure 11: A three class confusion matrix. The values signify the percentages of cases of the class indicated by the column that were classified by the neural network as belonging to the class indicated by the row.

negative values. The larger the absolute value of a risk factor the more significant the classification associated with that risk factor is to the overall objective [104]. An example of a matrix of risk factors appears in Figure 12.

Equation (14) shows how a total risk weighted measure of classification accuracy is achieved,

$$\text{Total Risk} = \sum_{i=1}^n \sum_{j=1}^n (P_{ij} R_{ij}) \quad (14)$$

where P_{ij} is the percentage of cases of class i classified by the neural network as class j and R_{ij} is its corresponding risk factor.

2.4.2 Neural Network Design Parameters

Though theoretically there may exist a neural network that can solve a particular problem, theory does not tell how to find that neural network [105]. One must empirically discover it. However, it is not discovered by a random search, but rather by systematically

Neural Network Identified Class	True Class		
	Class 1	Class 2	Class 3
Class 1	1	-1	-0.5
Class 2	-0.5	+1	-0.25
Class 3	0	0	0

Figure 12: A risk factor matrix for a confusion matrix. The values indicate a weighting for each classification or misclassification of a corresponding confusion matrix (see Figure 11)

exploring key components of neural network design. Of primary importance is the overall architecture: how many hidden nodes, how many hidden layers, and what connections between the layers. Other parameters, though not having as dramatic an effect as the overall architecture, nevertheless play an important supporting role in finding a satisfying solution to a problem. Some of them may play bigger roles than others for a specific problem. These other parameters include choices in input scaling ranges, transfer functions, learning rules, number of training cycles, epoch sizes, and error surface starting positions.

2.4.2.1 Neural Network Architecture

The architecture of a neural network, including number of hidden nodes, number of hidden layers, and the connections between the nodes in the various layers, is of primary importance to the performance of a network. There is an optimum configuration for a particular type of problem.

The most fundamental consideration of any neural network is the number of hidden nodes. Too few hidden nodes will prevent the neural network from training well and prevent a

Figure 13: Over determination of the training set can occur as the number of hidden nodes increases. A result of over determination is an increase in error for the test set.

solution [34]. But too many hidden nodes can cause an “over-training” problem that does not become apparent until the testing phase. That is, the neural network will have fit the training patterns so well during the training phase that it is unable to infer anything outside of the training set and performs poorly during the testing phase. This relationship is illustrated in Figure 13 [106]. An over-trained neural network becomes nothing more than a look up table for the training set [107]. This effect is analogous to fitting n data points to a function with m independent variables when $(n-1) \leq m$. The more nodes that a network has the easier it can fit the patterns. Cutting back on the number of nodes creates a constriction in the learning

process and makes the neural network more general [108]. A general rule of thumb for the number nodes is that, on average, there should be ten samples per weight [106]. Another formula cited [109] for determining the number of nodes in the hidden layer is,

$$\# \text{ Hidden nodes} = \frac{(\# \text{ Training cases})}{5 \times [(\# \text{ Output nodes}) + (\# \text{ Input nodes})]} \quad (15)$$

However, these formulas falter in many practical instances and the actual size of the network must be determined by trial and error [110, 111].

The number of nodes and the number of cases in the training set are interrelated as both Equation (15) and the ten samples per weight rule demonstrate. Thus, many of the conditions detailed in section 2.2.2.3 concerning the number of cases in the training data set parallel the requirements for the number of nodes for a satisfactory solution. In many cases it depends upon the type of data being used. In behavioral models the number of nodes must adhere to Equation (15) or the ten samples per weight rule fairly closely. However, for physical models the number of hidden nodes can be increased [98]. Even for behavioral models the number of hidden nodes can be larger than that suggested by Equation (15) and the ten samples per weight rule as long as the training data set is relatively free of noisy data or idiosyncratic examples [99].

Another consideration in building a neural network is the number of hidden layers. A neural network with a single hidden layer, given adequate processing elements, can

theoretically represent any mathematical or logical function [112, 113]. In real applications though, particularly for difficult problems, more hidden layers may be necessary [114]. This is because a neural network can construct complex decision regions: a hyper-plane is generated by each node in a first hidden layer; convex regions are formed from hyper-planes by each node in a second hidden layer; concave regions are created from convex regions by each node in a third hidden layer [115]. A concave figure is one in which a line segment can be drawn between any two points in a geometric set in which no points on that line will be outside of that set. Convexity allows for points on that line segment to be outside of the set [116]. Figure 14 depicts a set of decision regions created by an input layer and one hidden layer.

Determining what connections should exist between the various nodes and layers is difficult to determine. Sometimes connections exist only between adjacent layers. Sometimes all prior layers feed into all subsequent layers. The connection scheme which works best can sometimes be arrived at by using a Hinton diagram to eliminate nonessential connections.

2.4.2.2 Neural Network Input Scaling

Beyond any preprocessing the user does to the data, the neural network does a linear mapping of the input data within user specified ranges. Ranges such as [0, 1] or [-1, 1] are typical. This mapping allows the network to work with numbers that are within the ranges compatible with the summation and transfer functions of the processing elements. The mapping is done for a particular input field by mapping an input value, f , to a new mapped input value I ,

Figure 14: Decision regions created by two inputs (x) and three hidden units (h).

$$I = \frac{(R - r) \times f + (M \times r - m \times R)}{(M - m)} \quad (16)$$

where the source input minimum and maximum are r and R and the destination input minimum and maximum are m and M [117].

When the inputs are scaled between zero and one the average is 0.5. This type of scaling enhances the effects of average input behavior. A scaling between -1 and +1 provides

an average of zero. This type of scaling enhances the effects of deviant input behavior [118].

2.4.2.3 Transfer Function

There are a multitude of transfer functions available for the hidden and output nodes. Hidden node transfer functions of the sigmoid [Equation (2)] and the hyperbolic tangent [Equation (3)] are the two most common transfer functions. Although the sigmoid function seems somewhat more popular, its output (i.e., $[0,1]$) could lead to a bias to learning outputs approaching one more than those closer to zero. The hyperbolic tangent function gives a more even learning distribution to low and high values [68]. It has been reported that adding $-\frac{1}{2}$ onto the sigmoid function will restrict the activation level range from $-\frac{1}{2}$ to $+\frac{1}{2}$, and double the learning speed [119].

Among the transfer functions available for the output nodes in NeuralWorks Professional II/PLUS there is a linear transfer function called “softmax output.” This transfer function was designed to be used with classification problems that are categorical in which the output vectors all sum up to one. It was designed to work with the one-in-n code which will be discussed in section 3.2.2 [120].

2.4.2.4 Learning Rule

There are a myriad of learning rules. The most common learning rule used in combination with back propagation is the delta rule which was discussed in section 1.5.3.3. With the delta rule the momentum and learning rate coefficients must be set by the user. This gives the user significant control over the learning process, though it can be difficult to determine what the settings should be during different stages of the learning process. A more

sophisticated adaptation of the delta rule is the extended-delta-bar-delta rule which automatically modifies the momentum and learning rate coefficients as training proceeds [121]. The difference between the delta rule and the extended-delta-bar-delta rule is roughly analogous to the difference between a manual and an automatic transmission in an automobile.

Another difference between these two learning rules is the point in training when error is back-propagated. The delta rule updates the weights after each training cycle. The extended-delta-bar-delta rule updates the weights after a specified number of training cycles (an epoch) hence making the error back propagated cumulative over a number of training cycles.

2.4.2.5 Number of Training Cycles

In addition to excessive number of hidden nodes, over-training can also occur from too large a number of training cycles [122]. In this situation, the neural network weight space converges on the training set to such a degree that the neural network cannot interpolate any other patterns. Figure 15 illustrates this effect. The NeuralWorks Professional II/PLUS “save best” function can be used to avoid over-training. The save best function can be set up to save the neural network to disk when the highest classification accuracy for the test set is achieved during training. The save best evaluation is conducted automatically at specified training cycle increments set by the user.

2.4.2.6 Epoch Size

For the extended-delta-bar-delta rule various epoch sizes can be tried since a cumulative error is back propagated. For neural networks training data appear to resonate at

Figure 15: Overtraining of a neural network may occur from too many training cycles. Past a certain number of training cycles the training set is over determined and generalization to the test set cannot occur.

particular epochs. Training with the wrong epoch can inhibit convergence to an optimum solution. For example, training with too small of an epoch may cause oscillations in the weights. Training with too large of an epoch may cause subtle trends to be lost [123].

2.4.2.7 Error Surface Starting Position

Since the error surface of a particular problem may have an extremely large number of local minima [82] it is desirable to approach the error surface from different positions to perhaps avoid some local minima “traps.” By changing the initial weight conditions and the random order in which patterns are presented to the neural network for training one has essentially embarked on the error surface from a different launch point. In NeuralWorks

Professional II/PLUS the “random seed” setting can be used for this purpose. The random seed is an integer, set by the user, which initializes a random number generating function. This function controls the randomization of the initial weights and the randomized order in which patterns in the training set are presented to the neural network during training. This permits the initial conditions to be reproduced by setting the same random seed or varying the initial state by changing the setting.

Chapter 3 - Data Preprocessing

3.1 Raw Database

The data for creating the raw database were obtained from files originally created using SYMPHONY™. These original files were converted from SYMPHONY™ format to Excel® 5.0 format where they were edited and organized so that they could be used to generate the feature database. Cell #240 was eliminated from the database because maintenance information for this cell was incomplete after November, 1987. One set of maintenance events (February, 1990) was eliminated due to missing water addition data.

The raw database was reorganized based on a time index relative to the capacity test dates. Maintenance events were established as the month when the maintenance tasks occurred. Four comparably time-spaced maintenance events were identified for each capacity test date. Time-until-capacity-test was the index for each maintenance event and were designated “t1, t2, t3, and t4” for the maintenance events closest through furthest from the capacity test date. Table 1 shows the time index used for maintenance events associated with each capacity test event. Each maintenance event had associated with it the maintenance tasks of float voltages, specific gravities, water additions, and electrolyte levels. These features were normalized using Equation 10.

Table 1: Time Indexing of Raw Database Using “Months Until Capacity Test.”

Capacity Test Event	Maintenance Event (t1)	Maintenance Event (t2)	Maintenance Event (t3)	Maintenance Event (t4)
May 1989	February 1989	November 1988	August 1988	February 1988
April 1990	May 1990	November 1989	August 1989	May 1989
September 1991	September 1991	March 1991	November 1990	August 1990

3.2 Feature Database

3.2.1 Features

Additional features that had proved useful for classifying the cells in previous studies of this battery [13-16], along with a few others that were hoped would prove effective, were generated from the normalized data represented in Table 1. These included combinations and relational transformations of the time indexed features associated with each capacity test event from Table 1. Ten features were generated for each of the maintenance tasks to produce a total of 40 additional features. Thus, a total of 56 features were established for each cell for each capacity test event. Table 2 defines these features and indicates their neural network input code number. Each of the 56 features in Table 2 was autoscaled separately using the Excel® “STANDARDIZE” function.

3.2.2 Class Encoding in the Feature Database

Based on the class definitions for the battery cells in section 2.2.2 the capacity tests produced the distribution shown in Table 3. Each of the cells associated with a capacity test event in the feature database was assigned a class in the feature database.

Table 2: Neural Network Input Codes for Capacity Test Event Features

Feature Description	Float Voltage	Specific Gravity	Water Addition	Electrolyte Level
t4 (see Table 1)	2	16	30	44
t3 (see Table 1)	3	17	31	45
t2 (see Table 1)	4	18	32	46
t1 (see Table 1)	5	19	33	47
mean of (t1,t2,t3,t4)	6	20	34	48
t1×t2×t3×t4	7	21	35	49
slope of (t1,t2,t3,t4)	8	22	36	50
correlation coefficient of (t1,t2,t3,t4)	9	23	37	51
t1-t2	10	24	38	52
t1-t3	11	25	39	53
t1-t4	12	26	40	54
t2-t3	13	27	41	55
t2-t4	14	28	42	56
t3-t4	15	29	43	57

A “one-of-n” code was used to designate the classes for each set of patterns in the feature database. That is, each class was represented by a separate data field. If a set of features belonged to a particular class the field associated with that class would have a “1” and the other class data fields would have “0.” Hence for classes 1, 2, and 3 the coding would be [1 0 0], [0 1 0], and [0 0 1] respectively. This type of coding is typical of classification

problems involving more than two classes and makes finding solutions for the neural network more efficient [124, 125]. For a two class problem only one data field would be necessary since two classes can be represented by a “0” or “1” in one data field.

Table 3: Number of Cells Within the Defined Class Boundaries for a Capacity Test Event

Capacity Test Event	Number of Class 1 Cells	Number of Class 2 Cells	Number of Class 3 Cells
March 1989	17	15	77
April 1990	17	24	80
September 1991	37	50	236

3.3 Data Sampling: Selection of Cells for the Training and Test Sets

Selection of the cells for the training and test sets had to address several concerns. One concern was that neural networks typically need hundreds to thousands of examples in the training data set to justify their use [57, 58]. The database contained 109 patterns from 1989, 121 patterns from 1990, and 323 patterns from 1991 (see Table 3). Compounding the first concern was the number of representatives of each class. The overwhelming representation of class 3 cells would bias the learning process in favor of class 3 patterns and inhibit the class 1 and class 2 patterns from being learned. Thus, many of the class 3 patterns had to be eliminated. It appeared that there would be an inadequate number of patterns for one year to be used as a training set and another as a test set as was done in earlier studies [13-16].

Another concern was that training with the maintenance/capacity events of one year, and then using the following year's maintenance/capacity events as a test set, would not provide a strong enough foundation to extrapolate to general and long term conditions [126].

The solution arrived at was to combine the 1989, 1990, and 1991 data, randomize the data, and then select one portion for a training set and another portion for a testing set. Combining the 1989, 1990, and 1991 data generated a large enough training set to make using a neural network acceptable. Randomizing the 1989, 1990, and 1991 data addresses the long term generality concern.

Although it was essential to use as many patterns as possible for this study, several patterns had to be eliminated. Since there were so few class 1 and class 2 cells, and their classifications were of primary importance, none of these cells were eliminated from the database. However, the number of class 3 cells was so much larger than the numbers of either the class 1 or class 2 cells that the number of class 3 cells had to be reduced in order to let each class have approximately equal representation during training. It was determined that if any class 3 cell had a class change to a lower capacity class throughout the study, then that class 3 cell would be included in the training set. For example, if a cell was class 3 in 1989 but changed to class 2 in either 1990 or 1991, then that cell would be included as a representative of class 3 for the 1989 capacity test event. However, if a class 2 cell changed to class 3 or a class 3 cell changed to class 1 during a later capacity test event then that cell would be eliminated as a class 3 representative since this information was of less importance to the goals of the study. That is, it was better to eliminate information on class 3 cells which had increased

in capacity in deference to class 3 cells which had decreased or stayed constant; in a practical deployment of a neural network it would be better to correctly identify those cells which are likely to deteriorate or stay constant in capacity than to identify cells which may later improve in performance.

For the cells which changed classes between the 1989, 1990, and 1991 capacity test events twelve class 3 cells changed to class 2, nine class 1 cells changed to class 3, and one class 1 changed to class 2. Ten cells changed to a higher capacity class (i.e., class 3 changing to class 1 or class 2 changing to class 1 or class 3). In 1990, capacity tests were conducted on some cells which were not tested in 1989. For those cells, two changed from class 1 to class 3, and two changed to a higher capacity class.

Some cells remained in the same class throughout all of the 1989, 1990, and 1991 capacity test events: sixty class 3 cells, nine class 2 cells, and seven class 1 cells. For those cells tested in 1990 but not in 1989 five class 3, two class 2 cells, and one class 1 cell remained in the same class from 1990 to the 1991 capacity test.

The capacity value for each capacity test event was sorted in ascending order. The class 3 cells which were to be included due to class changes to lower capacity classes were selected. Those class 3 cells which were to be excluded due to class changes to higher capacity classes were eliminated. The remaining class 3 cells (those which did not change classes over the study period) were then chosen so as to as broadly represent the distribution of class 3 cells as possible. These cells were chosen until the total number of class 3 cells fell into the approximate ranges of the number of class 1 and class 2 cells for a given capacity test event

The result of this process is presented in Table 4.

3.4 Training and Test Sets

With the cells selected, the division of the database into a training set and a test set was performed. The three capacity test events were combined on one spreadsheet. A column was created and filled with the Excel® “RAND” function such that each battery cell had associated with it a random number between zero and one. The database was then sorted in ascending and descending order approximately ten times using the random number column as the sort

Table 4: Number of Cells Selected from the Capacity Test Events for Producing the Training and Test Sets

Capacity Test Event	Number of Class 1 Cells	Number of Class 2 Cells	Number of Class 3 Cells
March 1989	17	15	18
April 1990	17	24	21
September 1991	37	50	45

index. After each sort the RAND function would produce a new set of random numbers so that each successive sort was different than the others. This effectively randomized the capacity test events between 1989, 1990, and 1991. The random number column was deleted and training and test sets were selected. For the training set the first 75% of the combined and randomized data was chosen. The test set comprised the remaining 25% of the data. Table 5 shows the number of cells from each capacity test event forming the combined and randomized

training set. Table 6 shows the same type of information for the test set.

The data represented by Tables 5 and 6 were then saved using Excel® in “tab delimited text” format to be used by NeuralWorks Professional II/PLUS as training and test sets.

Table 5: Origin and Number of Cells Forming the Combined and Randomized Training Set

Capacity Test Event	Number of Class 1 Cells	Number of Class 2 Cells	Number of Class 3 Cells
March 1989	13	7	15
April 1990	12	20	15
September 1991	26	39	35
Combined/Randomized Training Set	51	66	65

Table 6: Origin and Number of Cells Forming the Combined and Randomized Test Set

Capacity Test Event	Number of Class 1 Cells	Number of Class 2 Cells	Number of Class 3 Cells
March 1989	4	8	3
April 1990	5	4	6
September 1991	11	11	10
Combined/Randomized Test Set	20	23	19

Chapter 4 - Results and Discussion

4.1 Objective of the Investigation

Li had shown that using maintenance and fabrication features together provided better classification accuracies than when either set of features were used separately [16]. However, it required a complicated use of features. Perone and Chen had shown that using a “battery activity” index was better than using a time index for organizing the maintenance events into features [13-15]. However, using a battery activity index was not as simple and straight forward as using a time index. The approach used in this study was to see if the power of a neural network could, using easily acquired maintenance events and a simple time indexing, produce classification accuracies superior to what was achieved in any of the previous studies [12-16]. The goal was to achieve classifications such that greater than 90% of the poor performing cells could be removed from the battery and greater than 90% of the high performing cells could be selected with a minimum of the remaining cells being misclassified as high performing. The features and neural network parameters that help achieve this goal might prove both instructive as to the physical/chemical properties of low and high performing cells and provide the ability to select cells for creating a consistently high performance battery with a low possibility of cell reversals.

4.2 Feature Selection

Fundamental to the study was determining if using easily acquired maintenance events, based on a simple time indexing system, would provide features with the ability to meet the desired classification objectives. If this was achievable then a straightforward approach could be used to identify high performing cells and eliminate poor performing cells from a battery. The features which impart this ability could give insight into the physical/chemical properties of lead/acid batteries which are fundamental to their performance.

Preliminary training trials, done with all 56 features input with five hidden nodes and three output nodes, yielded an initial set of conditions for systematic feature selection. Training seemed best when the inputs were scaled in the $[-1,+1]$ range, the hyperbolic tangent transfer function was used with the hidden nodes, the softmax output transfer function was used with the output layer nodes, and the extended-delta-bar-delta rule was used as the learning rule with an epoch of 20. All other settings were the defaults generated by the NeuralWorks Professional II/PLUS back propagation Instanet command. A printout of a neural network containing these default settings is in Appendix A.

In the first round of the feature selection procedure all 56 of the features were input into a set of ten neural networks, each with a single hidden layer containing from one to ten hidden nodes. To gauge each neural network's performance, the test set's overall classification accuracy, as well as the balance of the individual classification accuracies, were monitored. The training set's classification accuracies were useful only as confirmation that the test set's results were not fortuitous. The highest classification accuracies of classes 1 and 2

were of primary importance with as few misclassifications between them as possible. As a result of this investigation, the neural network with one hidden node proved to be adequate; having more hidden nodes did not improve the overall classification accuracy. The percent classification accuracies for the one hidden node neural network were,

	Overall	Class 1	Class 2	Class 3
Test Set	66.1	60.0	78.3	57.9
Training Set	66.5	74.5	81.8	44.6.

In the test set 5% of the class 1 cells were misclassified as class 2 and 4.4% of the class 2 cells were misclassified as class 1. For the training set no class 1 cells were misclassified as class 2, but 4.6% of the class 2 cells were misclassified as class 1. Fifteen features, indicated by the Hinton diagram as insignificant, were eliminated as inputs for the next phase of feature selection.

In the second stage of the feature selection process the same set of neural networks with a single hidden layer, each having one to ten hidden nodes, were trained using the reduced set of features. Again the neural network with one hidden node provided classification accuracies as high as those with more hidden nodes. The percent classification accuracies for the one hidden node neural network were,

	Overall	Class 1	Class 2	Class 3
Test Set	72.6	75.0	78.3	63.2
Training Set	70.9	100	78.3	43.1.

For the test set there were no misclassifications of class 1 cells as 2, but 13.0% of the class 2 cells were misclassified as class 1. For the training set no class 1 cells were misclassified as class 2, but 9.1% of the class 2 cells were misclassified as class 1. Four more input features were identified as insignificant with the Hinton diagram and deactivated as inputs for the next round of training for feature selection.

At various points in the feature selection process, the stronger of the features which had previously been eliminated were reintroduced to see their effect. This cycle of feature reduction was repeated several times until a point was reached when deterioration of overall classification accuracy for the test set began to occur when additional features were eliminated. In all phases of the feature selection process the neural network that proved best had either one, two, or three hidden nodes. The final neural network of this feature selection procedure contained 13 input nodes and one hidden node. The percent classification accuracies were,

	Overall	Class 1	Class 2	Class 3
Test Set	71.0	70.0	82.6	57.9
Training Set	65.4	82.4	80.3	36.9.

For the test set there were no misclassifications between classes 1 and 2. For the training set no class 1 cells were misclassified as class 2, but 4.6% of the class 2 cells were misclassified as class 1.

The neural network input codes for the features (see Table 2) which appeared most significant were 2, 5, 6, 7, 10, 14, 17, 20, 21, 30, 41, 42, and 43. Of these, input codes 2, 5, 6,

7, 10, 14, and 17 stood out as more important than input codes 20, 21, 30, 41, 42, and 43. An examination of Table 2 reveals that of the eight most important of the selected features, seven derive from the float voltages (input codes 2, 5, 6, 7, 10, and 14) and the other is the specific gravity feature t3 (input code 17). The least important selected features all derive from the water added maintenance task. None of the selected features were derived from the electrolyte level maintenance task. The reasons for the relative importance of the features will be discussed in a later section (section 4.4).

To verify that this procedure had retained the most significant features and eliminated less significant features, various experiments were conducted. One experiment trained various neural networks using only those features that had been excluded during the feature selection process (the non-selected feature set). The percent classification accuracies with one hidden node were,

	Overall	Class 1	Class 2	Class 3
Test Set	56.5	70.0	47.8	52.6
Training Set	66.5	78.4	65.2	58.5.

For the test set 5% of class 1 cells were misclassified as class 2, and 13.0% of the class 2 cells were misclassified as class 1. For the training set no class 1 cells were misclassified as class 2 but 7.6% of the class 2 cells were misclassified as class 1.

Using the non-selected set of features the number of feature inputs were reduced to 26 using the same procedure as before with percent classification accuracies of,

	Overall	Class 1	Class 2	Class 3
Test Set	64.5	65.0	56.5	73.7
Training Set	73.6	80.4	62.1	80.0.

For the test set 10% of the class 1 cells were misclassified as class 2, and 17.4% of the class 2 cells were misclassified as class 1. For the training set no class 1 cells were misclassified as class 2, but 3.0% of the class 2 cells were misclassified as class 1. When more inputs were deactivated the test set misclassifications between classes 1 and 2 increased and the overall classification accuracy began to diminish.

Fifteen randomly selected features from the non-selected set were used as inputs to neural networks with one or two hidden nodes. The neural network input codes were 4, 9, 12, 19, 23, 26, 33, 34, 38, 39, 40, 47, 50, 51, 54. The neural network with two hidden nodes produced the best overall percent classification accuracy with results of,

	Overall	Class 1	Class 2	Class 3
Test Set	56.5	55.0	52.2	63.2
Training Set	56.6	60.8	60.6	49.2.

The test set had 20% of the class 1 cells misclassified as class 2, and 17.4% of the class 2 cells misclassified as class 1. For the training set 5.9% of the class 1 cells were misclassified as class 2, and 7.6% of the class 2 cells were misclassified as class 1.

During the non-selected feature set experiments one group of features had shown themselves to be consistently more important than the others. These were the features derived

from the float voltages. Thus, features derived from the float voltages were the strongest in both the selected feature set and the non-selected feature set. Since these features appeared to have essential classifying information they were evaluated by themselves. Using only the float voltage features as inputs (neural network input codes 2 - 15), various neural networks were tried with one to ten hidden nodes. Four hidden nodes produced the best overall classification accuracy with percent classification accuracies of,

	Overall	Class 1	Class 2	Class 3
Test Set	64.5	70.0	65.2	57.9
Training Set	58.8	74.5	68.2	36.9.

The test set had 20% of the class 1 cells misclassified as class 2, and 8.7% of the class 2 cells misclassified as class 1. For the training set 19.6% of the class 1 cells were misclassified as class 2, and 12.1% of the class 2 cells were misclassified as class 1.

What these series of experiments demonstrated was that through the use of the Hinton diagram the strongest features were successfully selected during the feature selection procedure. Though the features that belonged to the non-selected set contained information related to cell performance, it took many more of those features to even come close to the performance of the neural networks trained on the selected feature set. The inferiority of the non-selected feature set was also demonstrated by the lower overall classification accuracies and higher proportion of misclassifications between classes 1 and 2. Further it shows that, though the float voltage derived features were the most useful features, by themselves they do

not have the information to maximize the classification accuracies and minimize misclassifications.

4.3 Neural Network Optimization

Once the number of selected features had been established as being generally significant to all three classes, the other aspects of neural network design could be explored to obtain an optimized neural network. This included investigating the number of hidden layers, the number of hidden nodes within multiple hidden layers and their interconnections, variations in epoch length, and input scaling ranges.

The first consideration to neural network design is the overall architecture. This facet of design has the largest impact on overall neural network performance. Thus, a systematic study was conducted. The number of hidden layers was varied from one to three, with the number of nodes in each varying from one to four. Connections between layers were tried with only adjacent layers connected and with prior layers feeding into some or all subsequent layers. The outcome of this investigation yielded an architecture, depicted in Figure 16, that produced promising results. The inputs were the selected features described in section 3.4.1. The bias was input to all nodes in all layers. There were three hidden layers. The first hidden layer consisted of three nodes with inputs from all of the input layer nodes except nodes 20 and 21 (see Table 2). The second hidden layer consisted of three hidden nodes with inputs from the first hidden layer and all nodes in the input layer. The third hidden layer consisted of one node with inputs from all nodes in the input layer, and the first and second hidden layers. Each hidden layer node used the hyperbolic tangent transfer function. The output layer consisted of

Figure 16: The architecture determined to produce the best performing neural networks for this study. The input layer consists of 13 nodes of which ten feed into the first hidden layer. All of the input nodes feed into the second and third hidden layers. The first hidden layer, consisting of three nodes, feeds into the second and third hidden layers. The second hidden layer, consisting of three nodes, feeds only into the third hidden layer. The third hidden layer, consisting of one hidden node, feeds only into the three-node output layer.

three nodes with the only input to each being from the third hidden layer node (and the bias).

The output layer nodes used the softmax output transfer function in conjunction with the one-in-n code. The explanation of why this architecture seems to work well will be discussed in section 4.4.

Having an adequate architecture in hand, the neural network performance was fine tuned by systematically investigating the epoch size and input scaling ranges. Epochs were varied from 5 to 175 training cycles. Input ranges had the form of $[-1, +1]$ or $[0, 1]$ in which ranges within these forms included such variations as $[-0.8, +0.8]$ and $[0.2, 0.8]$. Variations of

these parameters produced neural networks that were more specialized in their abilities and are discussed in detail in section 4.4.

4.3.1 Three Class Optimization

Since the problem was defined in terms of three classes, the most straightforward approach was to create a three-class optimized neural network. However, since not all classifications and misclassifications have the same significance a total risk factor was used as the criterion to judge the neural networks. The risk factor matrix depicted in Figure 12 was used with Equation (14) to calculate the total risk factor. Table 7 presents the confusion matrices and total risk factors for several neural networks that produced high total risk factors for 3-class optimization.

Inspection of the confusion matrices in Table 7 reveals that the manner in which misclassifications occurred was not by chance, but that the neural networks were actually finding decision regions based on the input patterns. Classes 1 and 2 were separated from one another more than either were from class 3 as indicated by the percentages of misclassifications. That is, most misclassifications usually occurred between class 3 and the other classes as one would expect if actual decision regions were being formed. Depending upon the user's tolerance for various misclassifications the evaluation presented in Table 7 would change somewhat.

Examining the risk factors and overall percent classification accuracies in Table 7 reveals that the training set typically had lower values than the test set. This is due to the NeuralWorks Professional II/PLUS save best function which uses the performance of the test

set and not the training set as the criterion for saving the neural network to disk. At a certain number of training cycles over-training begins to occur: the test set's results become poorer as the training set's results begin to improve (see Figure 15). In general, to have confidence in the overall performance of a neural network on unknown patterns, it is necessary for the training set have comparable or better results than the test set. Neural network 3 in Table 7 meets these criteria the best.

4.3.2 Two Class Optimization

An approach that was explored as possibly an easier problem for a neural network was to solve for two classes rather than three. This consisted of breaking the overall classification problem into two. For one problem, class 1 cells were to be separated from a class consisting of both class 2 and class 3 cells. The other problem was to separate class 2 cells from a class consisting of both class 1 and class 3 cells.

The same architecture and inputs as used in the three class optimization (section 3.5.1) were used in these experiments. However, the output layer only consisted of one node, since this is all that is required in a two class problem.

4.3.2.1 Class 1 versus Classes 2 and 3

The results from this study are presented in Table 8. The higher the classification accuracy for class 1 the higher the number of misclassifications from classes 2 and 3. This was the same trend that the three class study had. The difference here is that information is lost regarding what cells are being misclassified as class 1. In the three class evaluation (section 3.5.1) relatively high amounts of misclassifications by the other two classes as class 1 may be

Table 7: Confusion Matrices (Percent Classified), Overall Percent Classification Accuracies and Total Risk Factors for 3-Class Neural Network Test and Training Sets

NN ID ^a	PC ^b	Test Set Confusion Matrix (%)					Training Set Confusion Matrix (%)				
		True Class			OCA ^c	Total Risk	True Class			OCA ^c	Total Risk
		Class 1	Class 2	Class 3			Class 1	Class 2	Class 3		
1	1	75.0	0.0	10.5	79.0	155.8	82.4	3.0	21.5	73.6	146.6
	2	0.0	91.3	21.1			0.0	83.3	21.5		
	3	25.0	8.7	68.4			17.6	13.7	57.0		
2	1	75.0	0.0	21.1	77.4	154.9	88.2	3.0	27.7	72.0	149.7
	2	0.0	95.7	21.1			0.0	84.9	26.2		
	3	25.0	4.3	57.8			11.8	12.1	46.1		
3	1	80.0	0.0	21.1	77.4	155.5	94.1	6.1	27.7	74.7	153.2
	2	0.0	91.3	21.1			0.0	84.8	23.1		
	3	20.0	8.7	57.8			5.9	9.1	49.2		
4	1	80.0	4.3	21.1	79.0	155.5	88.2	7.6	35.4	69.2	140.1
	2	0.0	95.7	21.1			0.0	83.3	24.6		
	3	20.0	0.0	57.8			11.8	9.1	40.0		
5	1	85.0	4.4	26.3	77.4	153.5	96.1	7.6	46.2	67.6	144.9
	2	0.0	91.3	21.1			0.0	86.4	27.7		
	3	15.0	4.3	52.6			3.9	6.0	26.1		
6	1	90.0	4.4	21.1	77.4	152.4	90.2	9.1	36.9	68.1	136.8
	2	0.0	82.6	21.1			0.0	80.3	24.6		
	3	10.0	13.0	57.8			9.8	10.6	38.5		
7	1	100	8.7	31.6	80.6	158.6	100	9.1	49.2	68.1	143.8
	2	0	87.0	15.8			0	83.3	23.1		
	3	0	4.4	52.6			0	7.6	27.7		

Neural network parameters: 1) 12,300 training cycles, epoch=65, [-1, +1] input scaling; 2) 14,900 training cycles, epoch=65, [-0.95, +0.95] input scaling; 3) 15,100 training cycles, epoch=51, [-1, +1] input scaling; 4) 9,300 training cycles, epoch=50, [-1.2, +1.2] input scaling; 5) 9,300 training cycles, epoch=33, [-0.9, +0.9] input scaling; 6) 12,300 training cycles, epoch=25, [-1.1, +1.1] input scaling; 7) 9,300 training cycles, epoch=25, [-0.9, +0.9] input scaling.

^a NN ID - Neural network identification.

^b PC - Neural network predicted class.

^c OCA - overall classification accuracy.

acceptable if, for example, the amount of class 2 cells identified as class 1 is very small. With the results of Table 8 the origin of the misclassifications are unknown making the results less informative than those from the three class study.

4.3.2.2 Class 2 versus Classes 1 and 3

The results of this evaluation were more promising than the class 1 versus classes 2 and 3 study. Table 9 presents the results of the two best neural networks. For neural network number 14 potentially 92.3% of the class 2 cells could be identified and removed from a battery with 13.0% of non-class 2 cells being removed. The training set's results corresponded fairly well to the test set's results, indicating that the neural network would probably apply in a general manner. However, these results are still not as good as those achieved in the three class study. This is because roughly twice as many cells belong to the combined class of classes 1 and 3 as belong to class 2. The 13% of non-class 2 cells of neural network number 14 is, in actual number of cells, larger than those indicated by the 15.8% in neural network number 7. Thus the percentages between Tables 7 and 9 cannot be directly compared.

4.3.3 Single Class Optimization Within a Three Class Neural Network Classifier

Another strategy was explored to maximize the classification accuracies. The approach taken was to create neural networks which specialized in classifying one class. The neural network would concentrate on achieving the highest classification accuracy for one class at a time with the fewest number of false positives from the other two classes. The classification accuracy of the other two classes relative to one another would be

Table 8: Confusion Matrices (Percent Classified) and Overall Percent Classification Accuracies for 2-Class Neural Network Test and Training Sets: Class 1 Versus Classes 2 and 3

NN ID ^a	PC ^b	Test Set			Training Set		
		Confusion Matrix (%)			Confusion Matrix (%)		
		True Class		OCA ^c	True Class		OCA ^c
		1	2 & 3		1	2 & 3	
8	1	76.2	5.0	88.7	67.2	3.9	86.9
	2 & 3	23.8	95.0		32.8	96.1	
9	1	78.6	10.0	87.1	68.7	7.8	85.7
	2 & 3	21.4	90.0		31.3	92.2	
10	1	85.7	20.0	82.3	74.8	9.8	85.7
	2 & 3	14.3	80.0		25.2	90.2	
11	1	95.2	30.0	77.4	82.4	11.8	86.8
	2 & 3	4.8	70.0		17.6	88.2	
12	1	100	40.0	72.6	89.6	25.5	79.2
	2 & 3	0	60.0		10.7	74.5	

Neural network parameters: 8) 17,500 training cycles, epoch=50, [-1, +1] input scaling; 9) 20,500 training cycles, epoch=50, [-0.85, +0.85] input scaling; 10) 28,800 training cycles, epoch=60, [-0.85, +0.85] input scaling; 11) 24,100 training cycles, epoch=50, [-0.95, +0.95] input scaling; 12) 11,000 training cycles, epoch=50, [-1, +1] input scaling.

^a NN ID - Neural network identification.

^b PC - Neural network predicted class.

^c OCA - overall classification accuracy.

Table 9: Confusion Matrices (Percent Classified) and Overall Percent Classification Accuracies for 2-Class Neural Network Test and Training Sets: Class 2 Versus Classes 1 and 3

		Test Set			Training Set		
		Confusion Matrix (%)			Confusion Matrix (%)		
NN ID ^a	PC ^b	True Class		OCA ^c	True Class		OCA ^c
		<u>2</u>	<u>1 & 3</u>		<u>2</u>	<u>1 & 3</u>	
13	2	79.5	8.7	87.1	76.7	12.1	84.1
	1 & 3	20.5	91.3		23.3	87.8	
14	2	92.3	13.0	88.7	90.5	18.2	85.2
	1 & 3	7.7	87.0		9.5	81.8	

Neural network parameters: 13) 18,000 training cycles, epoch=15, [-1, +1] input scaling;
14) 20,565 training cycles, epoch=15, [-0.8, +0.8] input scaling.

^a NN ID - Neural network identification.

^b PC - Neural network predicted class.

^c OCA - overall classification accuracy.

inconsequential. However, since the goal of this investigation was to eliminate low capacity cells (class 2) and accurately separate out high capacity cells (class 1), class 3 was not of importance except as a source of false positives for class 1. The technique was used with some success by Li in her classification study for this battery using maintenance events in conjunction with the fabrication data [16].

The classification problem was hence broken up into two cases. The Case 1 objectives were to maximize class 1 classification accuracy but minimize the number of class 2 and class 3 cells identified as class 1. Misclassification of class 2 as class 1 was considered more undesirable than misclassification of class 3 as class 1. The Case 2 objectives were to maximize class 2 classification accuracy but minimize the number of class 2 cells identified as class 1, and classes 1 and 3 cells identified as class 2.

For this type of classification investigation the total risk factor [Equation (14)] simplifies to the sum of three products: for Case 1 only the risk factor matrix row which the neural network identifies as class 1 need be considered; for Case 2 one only need be concerned with the risk factor matrix row which the neural network identifies as class 2. Hence specialist neural networks can be created.

Table 10 presents a graduation of neural networks for Case 1 depending upon ones tolerance for misclassifications of class 2 and 3 cells. That is, a neural network with the ability to identify a larger number of the high performance cells would require a tolerance for a larger portion of lower performing cells to be included. Examination of the risk factors for the training and test sets shows that neural network 17 is consistently high for both. Hence, neural

Table 10: Confusion Matrices (Percent Classified), Overall Percent Classification Accuracies, and Total Risk Factors for Case 1 Neural Network Test and Training Sets

Test Set							Training Set				
NN ID ^a	PC ^b	Confusion Matrix (%)				Total Risk	Confusion Matrix (%)				Total Risk
		True Class			OCA ^c		True Class				
		Class 1	Class 2	Class 3			Class 1	Class 2	Class 3	OCA ^c	
15	1	70.0	0.0	0.0	75.8	70.0	66.7	1.5	10.8	73.1	59.8
	2	5.0	82.6	26.3			0.0	83.3	23.1		
	3	20.0	17.4	73.7			33.3	12.1	64.6		
16	1	75.0	0.0	5.3	74.2	72.4	78.4	4.5	13.8	73.6	67.0
	2	5.0	78.3	21.1			0.0	84.8	27.7		
	3	20.0	21.7	68.4			21.6	10.6	58.5		
17	1	80.0	4.4	5.3	67.7	73.0	94.1	4.6	15.4	72.0	81.8
	2	10.0	69.6	42.1			0.0	74.2	32.3		
	3	10.0	26.1	52.6			5.9	21.2	52.3		
18	1	90.0	4.4	21.1	77.4	75.1	90.2	9.1	36.9	68.1	62.7
	2	0.0	82.6	21.1			0.0	80.3	24.6		
	3	10.0	13.0	57.9			9.8	10.6	38.5		
19	1	95.0	4.4	31.6	72.6	74.8	98.0	4.6	49.2	68.1	68.8
	2	0.0	73.9	21.1			0.0	89.4	27.7		
	3	5.0	21.7	47.4			2.0	6.1	23.1		
20 ^d	1	100	8.7	31.6	80.6	75.5	100	9.1	49.2	68.1	66.3
	2	0	87.0	15.8			0	83.3	23.1		
	3	0	4.4	52.6			0	7.6	27.7		

Neural network parameters: 15) 15,600 training cycles, epoch=40, [-1, +1] input scaling; 16) 15,600 training cycles, epoch=35, [-1, +1] input scaling; 17) 17,800 training cycles, epoch=20, [-1, +1] input scaling; 18) 12,300 training cycles, epoch=25, [-1.1, +1.1] input scaling; 19) 8,600 training cycles, epoch=20, [-1, +1] input scaling, input nodes 41 and 43 not fed into first hidden layer; 20) 9,300 training cycles, epoch=25, [-0.9, +0.9] input scaling.

^a NN ID - Neural network identification.

^b PC - Neural network predicted class.

^c OCA - overall classification accuracy.

^d This neural network is identical to neural network 7 in Table 7.

network 17, among all of those in Table 10, could probably be deployed in the real world with the most overall confidence. However, if a high tolerance for class 3 cells was acceptable (to such a degree that you may have more class 3 cells than your neural network indicates) then neural networks 19 and 20 could be deployed with confidence since the classification accuracies for all but class 3 correspond well in both the training and test sets.

Table 11 presents the best neural networks specializing in Case 2. Based on the how well the risk factors correspond in both the training and test sets, neural network 21 seems the best choice to be deployed with the most confidence in the real world. Although neural networks 22 and 23 both have higher test set classification accuracies for class 2, the corresponding training set class 2 classification accuracies do not match well. However, neural networks 22 and 23 show great potential for becoming optimum solutions for the Case 2 problem.

The potential net effect of applying both specialist neural networks to the problem would be to identify, with a certain degree of assurance, a consistent set of high performing cells. This could be achieved by removing those cells identified as class 2 (by the class 2 specialist) from that set of cells identified as class 1 (by the class 1 specialist).

4.4 Interpreting the Neural Network

The weights and connections of the neural network that produced the best results are not readily interpreted. The tools or theories that allow for an interpretation are not yet available - and perhaps may never be. However, the architectures of the neural networks that produced the best results can be rationalized. The best performing neural networks, from the

Table 11: Confusion Matrices (Percent Classified), Overall Percent Classification Accuracies, and Total Risk Factors for Case 2 Neural Network Test and Training Sets

		Test Set					Training Set				
NN ID ^a	PC ^b	Confusion Matrix (%)				Total Risk	Confusion Matrix (%)				Total Risk
		True Class			OCA ^c		True Class			OCA ^c	
		Class 1	Class 2	Class 3			Class 1	Class 2	Class 3		
21	1	60.0	0.0	10.5	74.2	83.1	76.5	1.5	18.5	73.1	80.3
	2	0.0	87.0	15.8			0.0	86.4	24.6		
	3	40.0	13.0	73.7			23.5	12.1	56.9		
22	1	70.0	0.0	15.8	75.8	86.0	80.4	3.0	20.0	74.2	79.4
	2	0.0	91.3	21.1			0.0	84.8	21.5		
	3	30.0	8.7	63.2			19.6	12.1	58.5		
23	1	75.0	0.0	21.1	77.4	90.4	88.2	3.0	27.7	72.0	78.4
	2	0.0	95.7	21.1			0.0	84.9	26.2		
	3	25.0	4.4	57.9			11.8	12.1	46.2		

Neural network parameters: 21) 9,400 training cycles, epoch=70, [-1, +1] input scaling; 22) 15,720 training cycles, epoch=125, [-1, +1] input scaling; 23) 14,900 training cycles, epoch=20, [-0.95, +0.95] input scaling.

^a NN ID - Neural network identification.

^b PC - Neural network predicted class.

^c OCA - overall classification accuracy.

feature selection procedure to the optimization phase, all had one common architectural element: the output layer always had only one hidden node feeding into it. The reason for this probably lies in Equation (15) which balances the total number of nodes in a neural network with the number of training cases. The number of training cases was fixed at 182 and the number of output nodes was fixed at three. During the feature selection phase the number of input nodes ranged from 13 to 56. Plugging in these values into Equation (15) yields less than one hidden node when using more than 33 input nodes to two hidden nodes when using 13 input nodes. Since the best neural networks from the feature selection procedure had either one or two hidden nodes the architecture corresponded well with the general guidelines established by Equation (15).

How do the optimized neural networks with three hidden layers (containing a total of seven hidden nodes) fit into Equation (15)? Since the third hidden layer receives inputs from all prior layers (the input layer, and hidden layers one and two), the optimized neural networks can be viewed as the same one hidden node neural network from the feature selection procedure, but with an additional six inputs from the first and second hidden layers. The first two hidden layers in Figure 16 appear to extract important information from the input features and then recombine them with the input features in the third hidden layer to enhance their effect above what was achieved when only one hidden node was used.

The reason various features proved more useful than others is believed to lie in the quality of the data from each maintenance task. The float voltages were obtained by well trained technicians under supervision. The specific gravity readings were obtained with a

rough and standardized technique using a hydrometer. The water added task was not recorded as accurately as the float voltages or the specific gravities. Chen found that features derived from the water added task may potentially be most significant because it reflected the total activity of the cell [13]. Thus, despite the lower quality of the water added records, its cell performance information content is demonstrated by its being the generator of several features in the selected feature set. The electrolyte level reading task was not undertaken with as much attention by the technicians taking the readings as the other tasks and thus may not provide as fine an indicator as needed by a study of this type.

Overall best performance tended to occur with epochs between 15 to 65 training cycles. However, in the specialist neural networks in which a single class was optimized within a three class classifier, the best class 1 classifiers had short epochs and the best class 2 classifiers had long epochs (compare Table 10 with Table 11). This indicates that subtle trends in the features may be key to correctly classifying the high performing cells. On the other hand, overall trends in the features seem to contain information related to classifying poor performing cells (see section 2.4.2.6).

In all cases, while holding all other variables constant, the input scaling in the $[-1,+1]$ form provided better performing neural networks than when the data was scaled in the $[0,1]$ range. Typically the $[0,1]$ scaled inputs had more class 2 cells misclassified as class 1 and consistently produced lower overall classification accuracies. The implication of this is that the deviant behavior in the features are more significant for determining cell performance than is the average behavior (see section 2.4.2.2).

4.5 Conclusions

The results achieved in this study using time indexed maintenance events are a major advancement over what had been accomplished in the previous studies of the GNB battery. The maximum overall classification accuracy achieved in this study was 80.6% (see neural network 7 in Table 7 or neural network 20 in Table 10). The maximum classification accuracies achieved were 100% for class 1 (neural network 7 or 20) and 95.7% for class 2 (neural networks 2, 4, and 23). Neural networks optimized for class 3 were not investigated though 73.7% was achieved by two neural networks presented (15 and 21).

The results of this study represent the first stage in the development of a practical way to identify the high and low performing cells in a battery in a simple manner. A set of perhaps 70-95% of the highest performing cells of a battery could potentially be selected with the assurance that probably only 4.5% of the class 2 cells would be included. Depending upon the tolerance for average performing cells nearly 100% of the high performing cells could be selected with about 9% of the poor performing cells included. The number of poor performing cells could probably be reduced even further with the application of a neural network specializing in identifying poor performing cells. The implications of this are manifold. For long strings of cells, such as the GNB battery evaluated in this study, the potential of cell reversals could be reduced considerably. For demanding applications, the highest performing cells could be selected. The economic benefits of being able to use routine maintenance events to predict cell performance is very attractive. Conducting capacity tests on a large battery, such as the GNB battery, takes an “army” of technicians or an extensive automated data

collection system, either of which is expensive. Also the battery must be taken out of service to conduct the capacity tests.

For future investigations neural networks other than back propagation should be investigated. Several other types of neural networks were briefly examined with the data for this study, but not enough information was gained to determine if any had greater potential than back propagation. The NeuralWorks Professional II/PLUS software has the ability to weight error such that specified classifications and misclassifications are given different significance. Weighted error should be investigated to determine if specialization of a neural network for a particular class could be achieved. Another area that should be examined more thoroughly is the production of more useful features. Features derived from float voltages in particular should be of primary interest. For the water added and electrolyte level tasks it is strongly recommended that higher quality data collection be implemented. Improvement here could lead to more powerful features [16]. And finally, the success of this investigation warrants extending this type of study to other battery systems such as sodium-sulphur, zinc, bromine, and others.

The results indicate the feasibility of the routine application of neural networks for performance prediction as part of a maintenance strategy for long-string energy storage systems. An ideal testing platform would be in a modern battery energy storage facility with sophisticated and automated data collection capability where individual cell discharge profiles could be obtained periodically. Thus, with a neural network's training periodically updated

using the automated data collection system, prediction of cell performance could be based on the maintenance history of each cell.

References

- 1 C. A. Vincent, F. Bonino, M. Lazzari, and B. Scrosati, *Modern Batteries: An Introduction to Electrochemical Power Sources*, (London: Edward Arnold Ltd, 1984), p. 12.
- 2 Sam Perone, Professor of Chemistry, San Jose State University, personal communication (November, 1994).
- 3 Hans Bode, *Lead-Acid Batteries*, trans. R. J. Brodd and Karl V. Kordesch, (New York: John Wiley and Sons, 1977), p. 3.
- 4 Vincent et al., pp. 12-15.
- 5 R. N. McClellan and S. L. Deshpande, *500 kWh Lead/Acid Battery for Peak Shaving Energy Storage, Testing and Evaluation*, EPRI-EM3707, (October 1984).
- 6 Sam P. Perone and Phil Symons, "Modeling the Occurrences of Cell Reversals in Long Strings of Lead/Acid Cells," *Journal of Power Sources*, **41**, 277-290 (1993).
- 7 Sam P. Perone and W. C. Spindler, "Battery Lifetime Prediction by Pattern Recognition Application to Lead-Acid Battery Life-Cycling Test Data," *Journal of Power Sources*, **13**, 23-38 (1984).
- 8 S. P. Perone and W. C. Spindler, *Power Sources 10, Proc. 14th Int. Power Sources Symp.*, L. J. Pearce (ed.), Brighton, UK, (September, 1984).
- 9 Sam P. Perone and W. C. Spindler, "Predicting Life of Electric Vehicle and Load-Leveling Lead Acid Batteries from Initial Acceptance Test Data by Use of Pattern Recognition Analysis," *Power Sources*, **10**, 507-523 (1985).
- 10 S. P. Perone and W. C. Spindler, *2nd Int. Conf. Batteries for Utility Energy Storage, Newport Beach, CA, July 24-28, 1989, Special Technical Publication*, (Palo Alto: Energy Power Research Institute, July, 1989).
- 11 S. P. Perone, W. C. Spindler, and S. Deshpande, *Proc 3rd Int. Conf. Batteries for Utility Energy Storage*, Kobe, Japan, March 18-22, 1991.
- 12 R. Petesh, Master Thesis, San Jose State University, August, 1992.

- 13 P. H. Chen, Master Thesis, San Jose State University, August, 1992.
- 14 Sam P. Perone, Robert Petesch, Pei-Hwei Chen, William C. Spindler, and Sanjay L. Deshpande, "Multivariate Analysis of Data from Fabrication, Testing and Operation of a Large Lead/Acid Peak-Shaving Battery," *Journal of Power Sources*, **37**, 379-402 (1992).
- 15 Sam P. Perone, Pei-Hwei Chen, and William C. Spindler "Prediction of Individual Cell Performance in a Large Lead/Acid Peak-Shaving Battery," *Journal of Power Sources*, **45**, 73-92 (1993).
- 16 X. Li, Master Thesis, San Jose State University, October, 1994.
- 17 R. A. Perez, *The Complete Battery Book*, (Blue Ridge Summit, PA: Tab Books, Inc., 1985), p. 25.
- 18 Vincent et al., p. 98.
- 19 Perez, p. 25.
- 20 Bode, p. ix.
- 21 Vincent et al., p.95.
- 22 Perez, p. 33.
- 23 Perez, p. 24.
- 24 Vincent et al., p. 106.
- 25 Bode, pp. 334-349.
- 26 W. Arthur Byers and S. P. Perone, "Computerized Pattern Recognition Applied to Ni-Cd Cell Lifetime Prediction," *Journal of the Electrochemical Society*, **126**, 720-725 (1979).
- 27 W. S. McCulloch and W. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biology*, **5**, 115-133, 1943.

- 28 Sholom M. Weiss and Casimir A. Kulikowski, *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*, (San Mateo, California: Morgan Kaufmann Publishers, Inc., 1991), pp. 1-16.
- 29 LiMin Fu, *Neural Networks in Computer Intellegence*, (New York: McGraw-Hill, 1994), p. 176.
- 30 Weiss and Kulikowski, p. 82.
- 31 Robert Hecht-Nielsen, *Neurocomputing*, (Reading, Massachusetts: Addison-Westley Publishing Company, 1991), p. 2.
- 32 Fu, p. 10.
- 33 J. Zupan and J. Gasteiger, "Neural Networks: A New Method for Solving Chemical Problems or Just a Passing Phase," *Anal. Chim. Acta*, **248**, 1-30 (1990).
- 34 Peter A. Jansson, "Neural Networks: An Overview," *Anal. Chem.*, **63**, 357A-362A (1991).
- 35 Weiss and Kulikowski, p. 168.
- 36 D. Hebb, *The Organization of Behavior*, (New York: John Wiley, 1949).
- 37 Hecht-Nielsen, p. 15.
- 38 Hecht-Nielsen, pp. 5-10.
- 39 R. Colin Johnson and Chappel Brown, *Cognizers: Neural Networks and Machines that Think*, (NewYork: John Wiley and Sons, 1988), p. 154.
- 40 M. Minsky and S. Papert, *Perceptrons: An Introduction to Computaional Geometry*, (Cambridge, Massachussetts: MIT Press, 1969).
- 41 Johnson and Brown, p. 154.
- 42 Fu, p. 4.
- 43 Jure Zupan and Johann Gasteiger, *Neural Networks for Chemists: An Introduction*, (New York: VCH Publishers, 1993), p. 27.

- 44 D. E. Rumelhart, G. E. Hinton, and R. J. Reynolds, "Learning Internal Representation by Error Propagation," *Explorations in the Microstructures of Cognition*, 1, (Cambridge, Massachusetts: MIT Press, 1986).
- 45 Johnson and Brown, p. 164.
- 46 *Neural Computing: A Technology Handbook for Professional II/PLUS and NeuralWorks Explorer*, (Pittsburgh, Pennsylvania: NeuralWare, Inc, Technical Publications Group, 1993), pp. 9-12.
- 47 Weiss and Kulikowski, pp. 153-165.
- 48 R. Rowe, V. J. Mulley, J. C. Hughes, I. T. Nabney, and R. M. Debenham, "Neural Networks for Chromatographic Peak Classification - A Preliminary Study," *LC-GC*, 12, 690-698 (1994).
- 49 P. Battersby, J. R. Dean, W. R. Tomlinson, S. M. Hitchen, and P. Myers, "Predicting Solubility in Supercritical Fluid Extraction Using a Neural Network," *Analyst*, 119, 925-928 (1994).
- 50 Fu, pp. 411-421.
- 51 Zupan, *Neural Networks for Chemists*, pp. 183-196, 211-220
- 52 Zupan, *Neural Networks for Chemists*, p 119.
- 53 Fu., pp. 298-301.
- 54 Daniel McNeill and Paul Freiberger, *Fuzzy Logic*, (New York: Simon and Schuster, 1993).
- 55 Fu, pp. 295-298.
- 56 *NeuralWorks Reference Guide: NeuralWorks Professional II/PLUS and NeuralWorks Explorer*, (Pittsburgh, Pennsylvania: NeuralWare, Inc, Technical Publications Group 1994), pp. 93, 104.
- 57 *Using NeuralWorks: NeuralWorks Professional II/PLUS and NeuralWorks Explorer*, (Pittsburgh, Pennsylvania: NeuralWare, Inc, Technical Publications Group, 1993), p. 144.

- 58 Zupan, *Neural Networks for Chemists*, p. 143.
- 59 Casimir C. Casey Klimasauskas, *Non-Linear Feed Forward Neural Networks*, Revision 4.3, (Pittsburgh, Pennsylvania: NeuralWare, Inc, Technical Publications Group, 1994), pp. 9-10.
- 60 Hecht-Nielsen, p. 26.
- 61 Zupan, *Neural Networks for Chemists*, pp. 42-43.
- 62 *Using NeuralWorks*, p. 64.
- 63 Zupan, *Neural Networks for Chemists*, p. 42.
- 64 Hecht-Nielsen, p. 22.
- 65 Hecht-Nielsen, p. 86.
- 66 Zupan, *Neural Networks for Chemists*, pp. 30-31.
- 67 Zupan, *Neural Networks for Chemists*, p. 29.
- 68 *Using NeuralWorks*, p. 64.
- 69 Zupan, *Neural Networks for Chemists*, p. 127.
- 70 *Neural Computing*, p. 72.
- 71 Zupan, *Neural Networks for Chemists*, p. 23.
- 72 Hecht-Nielsen, pp. 38, 69.
- 73 Zupan, *Neural Networks for Chemists*, p. 45.
- 74 Zupan, *Neural Networks for Chemists*, p. 41.
- 75 Weiss and Kulikowski, p. 99.
- 76 Weiss and Kulikowski, p. 100.
- 77 *Using NeuralWorks*, p. 40.

- 78 Zupan, *Neural Networks for Chemists*, p. 129.
- 79 Zupan, *Neural Networks for Chemists*, pp. 129-131.
- 80 Weiss and Kulikowski, p. 98.
- 81 Weiss and Kulikowski, p. 101.
- 82 Hecht-Nielsen, p. 130.
- 83 Hecht-Nielsen, p. 131.
- 84 *Using NeuralWorks*, pp. 136-137.
- 85 *Using NeuralWorks*, p. 94.
- 86 Zupan, *Neural Networks for Chemists*, p. 141.
- 87 Matthew Zeidenberg, *Neural Networks in Artificial Intelligence*, (New York: Ellix Horwood, Ltd., 1990), p. 116.
- 88 Fu, pp. 57, 151.
- 89 *Neural Computing*, p. 126.
- 90 Zupan, *Neural Networks for Chemists*, p. 53.
- 91 Zupan, *Neural Networks for Chemists*, p. 67.
- 92 *NeuralWorks Reference Guide*, p. 293.
- 93 Zupan, *Neural Networks for Chemists*, pp. 79-80.
- 94 Hecht-Nielsen, pp. 164-191.
- 95 *NeuralWorks Reference Guide*, pp. 307-312.
- 96 *Neural Computing*, pp. 157-180, 209-226, 277-292.
- 97 Zupan, *Neural Networks for Chemists*, p. 137.

- 98 *Using NeuralWorks*, pp. 19-20.
- 99 Dean Miklavic, NeuralWare, Inc., personal communication (January 1995).
- 100 Casimir C. Casey Klimasauskas, *Neural Network Development*, Revision 4.4, (Pittsburgh, Pennsylvania: NeuralWare, Inc, Technical Publications Group, 1994), p. 243.
- 101 Klimasauskas, *Neural Network Development*, p. 42.
- 102 J. Moody and J. Utans, *Principle Architecture Selection for Neural Networks, Neural Information Processing Systems 4*, (San Mateo, CA: Morgan Kaufman, 1992), pp. 683-690.
- 103 Klimasauskas, *Neural Network Development*, pp. 257,260-261.
- 104 Weiss and Kulikowski, pp. 18-23.
- 105 Fu, p. 177.
- 106 Weiss and Kulikowski, p. 104.
- 107 Weiss and Kulikowski, p. 102.
- 108 Fu, pp. 81-82, 89.
- 109 *Using NeuralWorks*, p. 20.
- 110 Zupan, *Neural Networks for Chemists*, p. 137-146.
- 111 Fu, pp. 33, 89.
- 112 Weiss and Kulikowski, p. 112.
- 113 K. Hornick, M. Stinhcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, **2:5**, 359-366 (1989).
- 114 Klimasauskas, *Neural Network Development*, pp. 272-274.
- 115 Fu, p. 31.

- 116 Minsky and Papert, pp. 6, 103.
- 117 *NeuralWorks Reference Guide*, pp. 227-228.
- 118 Klimasauskas, *Neural Network Development*, p. 123.
- 119 Fu, p. 88.
- 120 *NeuralWorks Reference Guide*, p. 62.
- 121 *NeuralWorks Reference Guide*, pp. 126, 132-134
- 122 Zupan, *Neural Networks for Chemists*, p. 142.
- 123 *Using NeuralWorks*, p. 152.
- 124 *NeuralWorks Reference Guide*, p. 23.
- 125 Klimasauskas, *Neural Network Development*, pp. 66-68.
- 126 Sam Perone, Professor of Chemistry, San Jose State University, personal communication (March, 1995).

Appendix A - Neural Network Parameters

The following is a printout of the NeuralWorks Professional II/PLUS parameters for neural network 7 from Table 7 (neural network 20 in Table 10 is equivalent):

Title: Combined and Randomized 1989, 1990, and 1991 Data

Display Mode: Hinton

Type: Hetero-Associative

Display Style: default

Control Strategy: backprop

L/R Schedule: edbd

9300 Learn

0 Recall

0 Layer

25 Aux 1

0 Aux 2

0 Aux 3

Network Global L/R Schedule: edbd

Learn Count	5000	0	0	0	0
Temperature	0.0000	0.0000	0.0000	0.0000	0.0000
Alpha Incr (S)	0.0950	0.0000	0.0000	0.0000	0.0000
Alpha Incr (E)	0.0000	0.0000	0.0000	0.0000	0.0000
Alpha Decrease	0.1000	0.0000	0.0000	0.0000	0.0000
Alpha Bound	2.0000	2.0000	2.0000	2.0000	2.0000
Convex Factor	0.7000	0.7000	0.7000	0.7000	0.7000
Mu Incr (Scl)	0.0100	0.0100	0.0100	0.0100	0.0100
Mu Incr (Exp)	0.0000	0.0000	0.0000	0.0000	0.0000
Mu Decrease	0.0100	0.0100	0.0100	0.0100	0.0100
Recovery Tol.	1.5000	0.0000	0.0000	0.0000	0.0000
Recall Count	1	0	0	0	0
Temperature	0.0000	0.0000	0.0000	0.0000	0.0000
Firing Rate	100	100	100	100	100
Input Clamp	0.0000	0.0000	0.0000	0.0000	0.0000
Mod Factor	1.0000	1.0000	1.0000	1.0000	1.0000
Gain	1.0000	0.0000	0.0000	0.0000	0.0000
Coefficient 5	0.0000	0.0000	0.0000	0.0000	0.0000
Coefficient 6	0.0000	0.0000	0.0000	0.0000	0.0000
Coefficient 7	0.0000	0.0000	0.0000	0.0000	0.0000
Coefficient 6	0.0000	0.0000	0.0000	0.0000	0.0000
Coefficient 7	0.0000	0.0000	0.0000	0.0000	0.0000

IO Parameters

Learn Data: File Rand. (rndtrn1) Binary Load

Recall Data: File Seq. (rndtst1) Binary Load

Result File: Desired Output, Output

```

UserIO Program: userio
  I/P Ranges:  -0.9000,  0.9000
  O/P Ranges:   0.0000,  1.0000
  I/P Start Col: 1      MinMax Table: rndtrn1
  O/P Start Col: 57     Number of Entries: 59
MinMax Table <rndtrn1>:
Col:    1      2      3      4      5      6
Min:   -2.3417 -2.6973 -2.4522 -4.2286 -2.9510 -2.9009
Max:    3.292  3.537  2.709  4.307  3.414  3.442
Col:    7      8      9     10     11     12
Min:   -2.2365 -2.1660 -2.2022 -2.4534 -2.9576 -1.9132
Max:   13.58   1.789  2.341  5.017  4.299  2.8
Col:   13     14     15     16     17     18
Min:  -1.8291 -2.7461 -3.1590 -3.1523 -3.4315 -3.8740
Max:   1.975  3.023  2.429  3.846  3.303  2.31
Col:   19     20     21     22     23     24
Min:  -4.0588 -4.0004 -3.2714 -1.6854 -2.2738 -3.5894
Max:   3.135  3.164  4.217  1.764  3.189  3.583
Col:   25     26     27     28     29     30
Min:  -3.4013 -3.7607 -3.0139 -3.4311 -3.4968 -3.5937
Max:   2.954  3.427  2.816  4.204  8.305  3.993
Col:   31     32     33     34     35     36
Min:  -3.3042 -14.3273 -8.8486 -8.8019 -2.1425 -2.0396
Max:   3.488  5.736  6.473  7.883  4.377  1.702
Col:   37     38     39     40     41     42
Min: -10.1425 -11.2002 -13.9614 -2.8623 -3.3340 -3.1466
Max:   4.263  3.687  2.985  2.764  3.413  3.752
Col:   43     44     45     46     47     48
Min:  -5.1031 -4.5894 -2.9114 -4.0613 -3.4980 -2.4599
Max:   2.23   3.404  3.039  3.365  3.675  5.177
Col:   49     50     51     52     53     54
Min:  -2.2622 -1.5282 -4.8714 -4.4728 -4.1440 -2.8103
Max:   6.348  1.591  2.532  2.581  2.439  5.188
Col:   55     56     57     58     59
Min:  -2.6894 -5.2192 0.0000 0.0000 0.0000
Max:   4.92   5.423  1      1      1
Layer: 1
  PEs: 1      Wgt Fields: 2      Sum: Sum
Spacing: 5     F' offset: 0.00    Transfer: Linear
Shape: Square      Output: Direct
Scale: 1.00   Low Limit: -9999.00 Error Func: standard

```

Offset: 0.00 High Limit: 9999.00 Learn: --None--
 Init Low: -0.100 Init High: 0.100 L/R Schedule: (Network)
 Winner 1: None Winner 2: None
 PE: Bias
 1.000 Err Factor 0.000 Desired
 0.000 Sum 1.000 Transfer 1.000 Output
 0 Weights 0.000 Error 0.000 Current Error
 Layer: In
 PEs: 13 Wgt Fields: 1 Sum: Sum
 Spacing: 5 F' offset: 0.00 Transfer: Linear
 Shape: Square Output: Direct
 Scale: 1.00 Low Limit: -9999.00 Error Func: standard
 Offset: 0.00 High Limit: 9999.00 Learn: --None--
 Init Low: -0.100 Init High: 0.100 L/R Schedule: (Network)
 Winner 1: None Winner 2: None
 PE: 2
 1.000 Err Factor -0.317 Desired
 -0.317 Sum -0.317 Transfer -0.317 Output
 0 Weights 0.000 Error 0.000 Current Error
 PE: 5
 1.000 Err Factor -0.631 Desired
 -0.631 Sum -0.631 Transfer -0.631 Output
 0 Weights 0.000 Error 0.000 Current Error
 PE: 6
 1.000 Err Factor -0.287 Desired
 -0.287 Sum -0.287 Transfer -0.287 Output
 0 Weights 0.000 Error 0.000 Current Error
 PE: 7
 1.000 Err Factor -0.300 Desired
 -0.300 Sum -0.300 Transfer -0.300 Output
 0 Weights 0.000 Error 0.000 Current Error
 PE: 10
 1.000 Err Factor -0.535 Desired
 -0.535 Sum -0.535 Transfer -0.535 Output
 0 Weights 0.000 Error 0.000 Current Error
 PE: 14
 1.000 Err Factor 0.399 Desired
 0.399 Sum 0.399 Transfer 0.399 Output
 0 Weights 0.000 Error 0.000 Current Error

PE: 17

1.000 Err Factor	-0.838 Desired	
-0.838 Sum	-0.838 Transfer	-0.838 Output
0 Weights	0.000 Error	0.000 Current Error

PE: 20

1.000 Err Factor	-0.900 Desired	
-0.900 Sum	-0.900 Transfer	-0.900 Output
0 Weights	0.000 Error	0.000 Current Error

PE: 21

1.000 Err Factor	-0.900 Desired	
-0.900 Sum	-0.900 Transfer	-0.900 Output
0 Weights	0.000 Error	0.000 Current Error

PE: 30

1.000 Err Factor	-0.348 Desired	
-0.348 Sum	-0.348 Transfer	-0.348 Output
0 Weights	0.000 Error	0.000 Current Error

PE: 41

1.000 Err Factor	-0.784 Desired	
-0.784 Sum	-0.784 Transfer	-0.784 Output
0 Weights	0.000 Error	0.000 Current Error

PE: 42

1.000 Err Factor	-0.056 Desired	
-0.056 Sum	-0.056 Transfer	-0.056 Output
0 Weights	0.000 Error	0.000 Current Error

PE: 43

1.000 Err Factor	0.900 Desired	
0.900 Sum	0.900 Transfer	0.900 Output
0 Weights	0.000 Error	0.000 Current Error

Layer: Hidden1

PEs: 3	Wgt Fields: 7	Sum: Sum
Spacing: 5	F ¹ offset: 0.10	Transfer: TanH
Shape: Square		Output: Direct
Scale: 1.00	Low Limit: -9999.00	Error Func: standard
Offset: 0.00	High Limit: 9999.00	Learn: Ext DBD
Init Low: -0.134	Init High: 0.134	L/R Schedule: (Network)
Winner 1: None		Winner 2: None

PE: 58

1.000 Err Factor	0.000 Desired	
-0.593 Sum	-0.532 Transfer	-0.532 Output
12 Weights	0.000 Error	0.811 Current Error

Input PE	Input Value	Weight	Type	Delta	Weight				
Bias	+1.0000	+0.1333	V-r	+0.0506	+1.3451	+0.8013	+0.0080	-0.0559	+6.1654
2	-0.3169	+0.1117	V-r	-0.0144	+0.9919	+0.7812	-0.0030	+0.1730	-1.4501
5	-0.6313	-0.1525	V-r	-0.0044	+0.8923	+0.6273	-0.0022	-0.1222	-0.3837
6	-0.2873	-0.1912	V-r	-0.0045	+1.1166	+0.6810	-0.0017	-0.1714	-1.5260
7	-0.2997	-0.1902	V-r	-0.0061	+1.3973	+0.7240	-0.0018	-0.1554	-1.6028
10	-0.5350	-0.2483	V-r	-0.0042	+0.4227	+0.5376	-0.0033	-0.1284	+0.6021
14	+0.3992	-0.6417	V-r	+0.0051	+0.5238	+0.6357	+0.0026	-0.5646	-1.1995
17	-0.8379	+0.4122	V-r	+0.0020	+0.7733	+0.7808	+0.0007	+0.1294	-0.3282
30	-0.3481	-0.0565	V-r	-0.0217	+0.8649	+0.7690	-0.0068	+0.0524	-2.0028
41	-0.7845	+0.0820	V-r	-0.0032	+0.3677	+0.6425	-0.0021	+0.1806	+1.7426
42	-0.0562	-0.7926	V-r	-0.0013	+0.8457	+0.7713	-0.0002	-0.3004	+0.8199
43	+0.9000	-0.4780	V-r	+0.0047	+0.6102	+0.6548	+0.0017	-0.4672	-1.2575

PE: 59

1.000 Err Factor	0.000 Desired	
0.696 Sum	0.602 Transfer	0.602 Output
12 Weights	0.000 Error	-1.004 Current Error

Input PE	Input Value	Weight	Type	Delta	Weight				
Bias	+1.0000	+0.1209	V-r	-0.0447	+0.5162	+0.4394	-0.0045	+0.0274	-10.7166
2	-0.3169	-0.2984	V-r	+0.0056	+0.5270	+0.4998	+0.0001	-0.2975	+1.1598
5	-0.6313	-2.4360	V-r	-0.0104	+0.8373	+0.7792	-0.0040	-1.8678	-0.3308
6	-0.2873	+0.6890	V-r	+0.0062	+0.6032	+0.5938	+0.0047	+0.5534	+0.8658
7	-0.2997	+0.6681	V-r	+0.0045	+0.3456	+0.5343	+0.0047	+0.5503	+0.9798
10	-0.5350	+0.5493	V-r	-0.0118	+0.4766	+0.5774	-0.0099	+0.6057	-0.3498
14	+0.3992	-0.2000	V-r	+0.0172	+0.5658	+0.6057	+0.0114	-0.2695	+0.6649
17	-0.8379	+0.3824	V-r	+0.0148	+0.4524	+0.5208	+0.0121	+0.4066	+0.7388
30	-0.3481	-0.3824	V-r	+0.0112	+0.8421	+0.5060	-0.0035	-0.2809	+3.6538
41	-0.7845	+0.7952	V-r	+0.0041	+0.4612	+0.6336	+0.0030	+0.6233	-1.2060
42	-0.0562	-0.4430	V-r	+0.0159	+0.6221	+0.6503	+0.0086	-0.1501	-0.0884
43	+0.9000	+0.5571	V-r	+0.0193	+0.7362	+0.6288	+0.0084	+0.4138	+1.6304

PE: 60

1.000 Err Factor	0.000 Desired	
-1.052 Sum	-0.782 Transfer	-0.782 Output
12 Weights	0.000 Error	1.521 Current Error

Input PE	Input Value	Weight	Type	Delta	Weight				
Bias	+1.0000	-0.0998	V-r	+0.0014	+0.6602	+0.5460	-0.0032	-0.1008	+12.9938
2	-0.3169	+0.2548	V-r	-0.0015	+0.5391	+0.6110	-0.0011	+0.2070	-2.4516
5	-0.6313	+0.4352	V-r	-0.0030	+0.5898	+0.6244	-0.0016	+0.3567	-0.6416
6	-0.2873	-0.0942	V-r	-0.0061	+0.8456	+0.5906	-0.0034	-0.0982	-2.6852
7	-0.2997	-0.2610	V-r	-0.0056	+0.7109	+0.5747	-0.0033	-0.2695	-2.8426
10	-0.5350	-0.7115	V-r	+0.0041	+0.7129	+0.6999	+0.0017	-0.3368	+1.2410

14	+0.3992	-0.1718	V-r	-0.0028	+0.3498	+0.5444	-0.0021	-0.2098	-2.4695
17	-0.8379	-0.3375	V-r	-0.0099	+0.6184	+0.6000	-0.0053	-0.2725	-0.8954
30	-0.3481	+0.2903	V-r	-0.0016	+0.6955	+0.6181	+0.0003	+0.2432	-4.3834
41	-0.7845	+0.8276	V-r	-0.0158	+1.1724	+0.7584	-0.0038	+0.3663	+2.8814
42	-0.0562	+0.0410	V-r	-0.0201	+1.1035	+0.7610	-0.0043	+0.0755	+1.2892
43	+0.9000	-0.6042	V-r	+0.0010	+0.4731	+0.6462	+0.0000	-0.5119	-2.2688

Layer: Hidden2

PEs: 3	Wgt Fields: 7	Sum: Sum
Spacing: 5	F' offset: 0.10	Transfer: TanH
Shape: Square		Output: Direct
Scale: 1.00	Low Limit: -9999.00	Error Func: standard
Offset: 0.00	High Limit: 9999.00	Learn: Ext DBD
Init Low: -0.577	Init High: 0.577	L/R Schedule: (Network)
Winner 1: None		Winner 2: None

PE: 61

1.000 Err Factor	0.000 Desired	
-1.080 Sum	-0.793 Transfer	-0.793 Output
17 Weights	0.000 Error	-1.836 Current Error

Input PE Input Value Weight Type Delta Weight

Bias	+1.0000	+0.7260	V-r	-0.0235	+0.4899	+0.4946	+0.0023	+0.5846	-7.2543
2	-0.3169	-0.0965	V-r	+0.0021	+0.3207	+0.4880	-0.0025	-0.0956	+0.1163
5	-0.6313	-1.5058	V-r	-0.0066	+0.6030	+0.7064	-0.0039	-1.2281	-0.5463
6	-0.2873	-0.2919	V-r	+0.0027	+0.3208	+0.5645	+0.0020	-0.2859	-0.1849
7	-0.2997	-0.0517	V-r	+0.0033	+0.3778	+0.5727	+0.0019	-0.0450	-0.1165
10	-0.5350	+0.4317	V-r	-0.0158	+0.5959	+0.6432	-0.0088	+0.4013	-0.0576
14	+0.3992	-0.2797	V-r	+0.0183	+0.6727	+0.5797	+0.0097	-0.1716	+0.0195
17	-0.8379	+0.7453	V-r	+0.0115	+0.4742	+0.5933	+0.0088	+0.6349	+0.5449
20	-0.9000	+0.3875	V-r	+0.0159	+0.9386	+0.7267	+0.0061	+0.3363	-1.3175
21	-0.9000	+0.0921	V-r	+0.0160	+0.9407	+0.7383	+0.0061	+0.0576	-1.2452
30	-0.3481	-0.1898	V-r	+0.0004	+0.6371	+0.5179	-0.0067	-0.1057	+2.5201
41	-0.7845	+1.0689	V-r	-0.0005	+0.7281	+0.7287	+0.0017	+0.9713	-0.1538
42	-0.0562	-0.8353	V-r	+0.0148	+0.6976	+0.6042	+0.0067	-0.6155	+0.4288
43	+0.9000	-0.4437	V-r	+0.0205	+0.8668	+0.6728	+0.0071	-0.4938	+0.9100
58	-0.5320	+0.7026	V-r	-0.0350	+0.9922	+0.6765	-0.0123	+0.6584	-1.4212
59	+0.6021	+0.3420	V-r	+0.0083	+0.5174	+0.6360	+0.0059	+0.3927	-0.2472
60	-0.7824	+0.2493	V-r	+0.0012	+0.8700	+0.6328	-0.0015	+0.3501	+0.2338

PE: 62

1.000 Err Factor	0.000 Desired	
1.402 Sum	0.886 Transfer	0.886 Output
17 Weights	0.000 Error	-1.912 Current Error

Input PE	Input Value	Weight	Type	Delta	Weight				
Bias	+1.0000	-0.0408	V-r	-0.0060	+0.8269	+0.5548	+0.0047	-0.1716	-17.6979
2	-0.3169	-0.6073	V-r	+0.0023	+0.6248	+0.5751	-0.0002	-0.4569	+2.7340
5	-0.6313	-1.2147	V-r	+0.0005	+0.4028	+0.6398	-0.0001	-1.0224	+0.4555
6	-0.2873	+0.2760	V-r	+0.0060	+0.6097	+0.6188	+0.0041	+0.1541	+2.8738
7	-0.2997	+0.5866	V-r	+0.0067	+0.6752	+0.6119	+0.0039	+0.5065	+3.0786
10	-0.5350	+0.9955	V-r	-0.0127	+0.7193	+0.6106	-0.0058	+0.8486	-1.3597
14	+0.3992	+0.2971	V-r	+0.0122	+0.7141	+0.5858	+0.0062	+0.3012	+2.7603
17	-0.8379	+0.0205	V-r	+0.0212	+0.7652	+0.6324	+0.0089	-0.0867	+1.1974
20	-0.9000	+0.9969	V-r	+0.0183	+0.8616	+0.6052	+0.0084	+0.7882	-2.1318
21	-0.9000	+0.4691	V-r	+0.0190	+0.8776	+0.6160	+0.0084	+0.2721	-1.9344
30	-0.3481	-0.9309	V-r	+0.0039	+0.4808	+0.4886	-0.0040	-0.8204	+5.9444
41	-0.7845	-0.8760	V-r	+0.0083	+0.4988	+0.5996	+0.0037	-0.6273	-3.4663
42	-0.0562	-0.4423	V-r	+0.0182	+0.7384	+0.6792	+0.0076	-0.3693	-1.2940
43	+0.9000	+0.6252	V-r	+0.0065	+0.7385	+0.6443	+0.0042	+0.3849	+3.0876
58	-0.5320	-0.2419	V-r	-0.0238	+1.0730	+0.6141	-0.0085	-0.1833	-4.8813
59	+0.6021	+0.0579	V-r	+0.0038	+0.4448	+0.6268	+0.0027	+0.1343	-3.2159
60	-0.7824	-0.9292	V-r	+0.0081	+0.9776	+0.7199	+0.0023	-0.6088	-1.3646

PE: 63

1.000 Err Factor	0.000 Desired	
-1.158 Sum	-0.820 Transfer	-0.820 Output
17 Weights	0.000 Error	-0.389 Current Error

Input PE	Input Value	Weight	Type	Delta	Weight				
Bias	+1.0000	-0.3774	V-r	-0.0068	+0.5852	+0.4926	+0.0004	-0.4268	-4.1655
2	-0.3169	-0.3501	V-r	-0.0004	+0.7016	+0.5740	-0.0005	-0.3559	+0.4344
5	-0.6313	-0.2911	V-r	-0.0012	+0.8527	+0.7165	-0.0006	-0.2145	-0.0643
6	-0.2873	-0.3190	V-r	+0.0021	+0.5593	+0.7224	+0.0010	-0.4440	+0.3916
7	-0.2997	+0.5526	V-r	+0.0011	+0.4220	+0.6578	+0.0010	+0.4492	+0.4361
10	-0.5350	+0.3313	V-r	-0.0045	+0.6967	+0.6283	-0.0023	+0.3602	-0.1992
14	+0.3992	+0.6496	V-r	+0.0076	+0.9272	+0.6582	+0.0028	+0.5915	+0.4353
17	-0.8379	+0.2160	V-r	+0.0037	+0.4835	+0.5079	+0.0030	+0.2323	+0.2478
20	-0.9000	+0.2140	V-r	+0.0059	+0.8677	+0.6887	+0.0024	+0.2185	-0.6140
21	-0.9000	-0.0748	V-r	+0.0055	+0.8447	+0.6619	+0.0024	-0.0683	-0.5693
30	-0.3481	-0.5760	V-r	+0.0019	+0.6694	+0.4769	-0.0013	-0.5362	+1.3910
41	-0.7845	+0.5139	V-r	+0.0007	+0.3401	+0.5514	+0.0005	+0.4602	-0.6170
42	-0.0562	-0.4581	V-r	+0.0042	+0.6201	+0.5795	+0.0025	-0.4732	-0.1030
43	+0.9000	+0.2185	V-r	+0.0051	+0.6935	+0.5854	+0.0026	+0.1842	+0.7033
58	-0.5320	+0.4345	V-r	-0.0129	+1.1281	+0.6914	-0.0040	+0.5032	-1.0786
59	+0.6021	-0.2839	V-r	+0.0033	+0.8130	+0.7697	+0.0012	-0.2629	-0.5314
60	-0.7824	+0.5046	V-r	+0.0010	+0.7255	+0.5652	-0.0004	+0.4899	-0.2250

Layer: Hidden3

PEs: 1 Wgt Fields: 7 Sum: Sum
 Spacing: 5 F' offset: 0.10 Transfer: TanH
 Shape: Square Output: Direct
 Scale: 1.00 Low Limit: -9999.00 Error Func: standard
 Offset: 0.00 High Limit: 9999.00 Learn: Ext DBD
 Init Low: -0.577 Init High: 0.577 L/R Schedule: (Network)
 Winner 1: None Winner 2: None
 PE: 64

1.000 Err Factor 0.000 Desired
 0.889 Sum 0.711 Transfer 0.711 Output
 20 Weights 0.000 Error 2.927 Current Error

Input PE Input Value Weight Type Delta Weight

Bias	+1.0000	+0.4052	V-r	+0.0341	+0.6315	+0.5159	+0.0044	+0.5146	+17.2399
2	-0.3169	+0.1918	V-r	-0.0031	+0.6087	+0.5457	+0.0003	+0.1418	-1.8265
5	-0.6313	+1.4108	V-r	+0.0046	+0.6628	+0.6902	+0.0026	+1.1881	+0.1831
6	-0.2873	-0.7977	V-r	-0.0043	+0.3848	+0.6460	-0.0039	-0.6120	-1.7181
7	-0.2997	-0.6720	V-r	-0.0042	+0.3907	+0.6413	-0.0039	-0.5075	-1.9029
10	-0.5350	-0.2296	V-r	+0.0128	+0.6228	+0.6180	+0.0076	-0.1351	+0.8629
14	+0.3992	-0.4085	V-r	-0.0193	+0.7281	+0.5949	-0.0096	-0.2939	-1.9022
17	-0.8379	-0.5027	V-r	-0.0134	+0.4661	+0.5022	-0.0108	-0.5205	-1.0616
20	-0.9000	-0.4237	V-r	-0.0165	+0.8149	+0.6385	-0.0078	-0.4426	+2.5310
21	-0.9000	-0.6127	V-r	-0.0179	+0.8418	+0.6565	-0.0078	-0.6418	+2.3457
30	-0.3481	+0.5061	V-r	-0.0119	+0.6855	+0.4710	+0.0022	+0.4462	-5.7866
41	-0.7845	-1.2348	V-r	-0.0025	+0.5054	+0.6554	-0.0023	-0.7745	+2.5006
42	-0.0562	+0.5344	V-r	-0.0116	+0.5184	+0.5788	-0.0083	+0.5405	+0.4009
43	+0.9000	-0.6564	V-r	-0.0252	+0.8270	+0.6704	-0.0088	-0.4732	-2.8588
58	-0.5320	+0.5276	V-r	+0.0422	+1.1041	+0.6562	+0.0136	+0.3649	+4.5277
59	+0.6021	-0.4620	V-r	-0.0094	+0.7006	+0.6883	-0.0046	-0.4855	+2.3064
60	-0.7824	-0.0384	V-r	-0.0055	+0.6676	+0.5954	+0.0003	+0.0429	+0.8840
61	-0.7933	-1.0653	V-r	+0.0137	+0.2465	+0.5083	+0.0140	-0.9294	+13.5074
62	+0.8858	-1.1093	V-r	+0.0037	+0.6911	+0.6009	-0.0034	-0.9100	+0.7000
63	-0.8204	-0.2259	V-r	-0.0065	+0.5859	+0.6059	-0.0007	-0.2169	-1.6283

Layer: Out

PEs: 3 Wgt Fields: 7 Sum: Sum
 Spacing: 5 F' offset: 0.10 Transfer: Linear
 Shape: Square Output: SoftMax
 Scale: 1.00 Low Limit: -9999.00 Error Func: SoftMax
 Offset: 0.00 High Limit: 9999.00 Learn: Ext DBD
 Init Low: -0.707 Init High: 0.707 L/R Schedule: (Network)
 Winner 1: None Winner 2: None

PE: 65

1.000 Err Factor 0.000 Desired
-5.220 Sum -5.220 Transfer 0.001 Output
2 Weights -0.001 Error -0.001 Current Error

Input PE Input Value Weight Type Delta Weight

Bias +1.0000 -2.3552 V-r +0.0172 +0.5647 +0.5435 +0.0463 -1.4994 -4.7245
64 +0.7109 -4.0298 V-r -0.0206 +0.5585 +0.5488 -0.0390 -3.5916 +2.9379

PE: 66

1.000 Err Factor 1.000 Desired
2.169 Sum 2.169 Transfer 0.865 Output
2 Weights 0.135 Error 0.135 Current Error

Input PE Input Value Weight Type Delta Weight

Bias +1.0000 +0.2446 V-r +0.1017 +0.3414 +0.4506 +0.0769 +0.0922 +1.3737
64 +0.7109 +2.7075 V-r +0.0300 +0.6973 +0.5876 +0.0160 +2.3515 +0.4328

PE: 67

1.000 Err Factor 0.000 Desired
0.306 Sum 0.306 Transfer 0.134 Output
2 Weights -0.134 Error -0.134 Current Error

Input PE Input Value Weight Type Delta Weight

Bias +1.0000 -0.0776 V-r -0.2146 +0.8171 +0.5117 -0.1232 +0.3835 +3.3509
64 +0.7109 +0.5403 V-r -0.0039 +0.4941 +0.5872 +0.0230 -0.1820 -3.3707

Appendix B - Cells Selected for the Training Set

1989 - Cell Number (Class):

17 (3), 23 (2), 29, (2), 40, (3), 41 (3), 59 (2), 60 (3), 65 (2), 78 (3), 90 (2), 91 (3), 102 (3), 109 (2), 139 (3), 185 (3), 197 (2), 221 (3), 230 (1), 231 (3), 232 (3), 234 (1), 239 (1), 243 (1), 245 (1), 246 (1), 263 (1), 265 (1), 273 (1), 276 (1), 278 (3), 286 (1), 297 (1), 298 (3), 303 (1), 307 (3).

1990 - Cell Number (Class):

9 (3), 13 (3), 17 (3), 20 (2), 21 (2), 22 (2), 25 (2), 28 (2), 41 (3), 59 (2), 60 (2), 62 (2), 63 (2), 65 (2), 102 (3), 105 (2), 109 (2), 128 (3), 130 (3), 140 (3), 185 (2), 192 (2), 197 (2), 199 (2), 219 (3), 221 (2), 228 (2), 231 (3), 232 (3), 233 (3), 235 (3), 236 (1), 239 (1), 243 (1), 245 (1), 246 (1), 249 (2), 251 (1), 252 (1), 262 (1), 265 (1), 276 (1), 286 (3), 297 (1), 298 (3), 299 (2), 303 (1).

1991 - Cell Number (Class):

9 (3), 12 (3), 13 (3), 14 (3), 15 (3), 17 (2), 21 (2), 22 (2), 24 (3), 26 (2), 27 (2), 28 (2), 29 (2), 30 (2), 36 (2), 39 (2), 40 (2), 43 (1), 56 (1), 59 (2), 62 (2), 67 (2), 70 (3), 74 (2), 87 (3), 91 (2), 99 (3), 102 (2), 105 (2), 106 (3), 110 (2), 111 (2), 121 (1), 130 (3), 131 (3), 136 (3), 137 (3), 139 (3), 145 (3), 152 (1), 183 (2), 185 (2), 186 (2), 188 (1), 192 (2), 197 (2), 198 (2), 199 (2), 200 (2), 205 (2), 206 (2), 208 (2), 212 (3), 214 (2), 219 (3), 222 (3), 223 (3), 230 (1), 231 (2), 233 (3), 236 (3), 239 (1), 241 (3), 243 (1), 244 (3), 245 (3), 246 (3), 247 (3), 248 (3), 249 (2), 254 (1), 261 (1), 262 (3), 263 (3), 264 (1), 265 (1), 266 (1), 270 (1), 273 (1), 276 (1),

1991 (Concluded) - Cell Number (Class):

279 (2), 280 (1), 281 (1), 282 (1), 285 (2), 286 (3), 292 (1), 293 (1), 297 (3), 298 (2), 303 (1),
304 (1), 306 (1), 307 (3), 310 (1), 316 (3), 317 (1), 319 (3), 323 (2), 332 (1).

Appendix C - Cells Selected for the Test Set

1989 - Cell Number (Class):

18 (2), 21 (2), 22 (2), 25 (2), 28 (2), 62 (2), 63 (3), 105 (2), 140 (3), 192 (3), 199 (2), 235 (1),
236 (1), 242 (1), 262 (1).

1990 - Cell Number (Class):

18 (3), 23 (2), 29 (2), 40 (3), 91 (2), 146 (1), 225 (2), 230 (1), 234 (3), 244 (3), 254 (1), 263
(3), 267 (1), 273 (1), 278 (3).

1991 - Cell Number (Class):

32 (2), 41 (3), 45 (1), 60 (2), 63 (2), 78 (3), 86 (3), 101 (3), 109 (2), 112 (2), 114 (2), 120 (2),
135 (3), 138 (3), 179 (1), 210 (1), 217 (3), 221 (2), 224 (1), 225 (2), 232 (3), 234 (3), 235 (2),
242 (1), 268 (1), 271 (1), 274 (1), 278 (2), 289 (1), 291 (1), 294 (3), 324 (1).