

9-15-2011

# Identification and Classification of Player Types in Massive Multiplayer Online Games using Avatar Behavior

Earl M. Bednar

Follow this and additional works at: <https://scholar.afit.edu/etd>

Part of the [Operational Research Commons](#)

---

## Recommended Citation

Bednar, Earl M., "Identification and Classification of Player Types in Massive Multiplayer Online Games using Avatar Behavior" (2011). *Theses and Dissertations*. 1486.  
<https://scholar.afit.edu/etd/1486>

This Dissertation is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact [richard.mansfield@afit.edu](mailto:richard.mansfield@afit.edu).



IDENTIFICATION AND CLASSIFICATION OF PLAYER TYPES IN MASSIVE  
MULTIPLAYER ONLINE GAMES USING AVATAR BEHAVIOR

DISSERTATION

Earl M. Bednar, Major, USAF

AFIT/DS/ENS/11-01

DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY

**AIR FORCE INSTITUTE OF TECHNOLOGY**

---

---

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

AFIT/DS/ENS/11-01

IDENTIFICATION AND CLASSIFICATION OF PLAYER TYPES IN MASSIVE  
MULTIPLAYER ONLINE GAMES USING AVATAR BEHAVIOR

DISSERTATION

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the  
Degree of Doctor of Philosophy in Operations Research

Earl M. Bednar, B.S., M.S.

Major, USAF

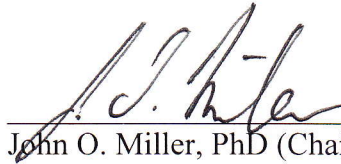
Aug 2011

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

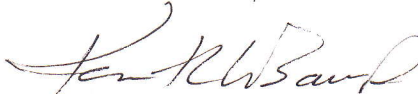
IDENTIFICATION AND CLASSIFICATION OF PLAYER TYPES IN MASSIVE  
MULTIPLAYER ONLINE GAMES USING AVATAR BEHAVIOR

Earl M. Bednar, B.S., M.S.  
Major, USAF


Approved:

  
\_\_\_\_\_  
John O. Miller, PhD (Chairman)

17 Aug 2011  
Date


  
\_\_\_\_\_  
Kenneth W. Bauer, Jr., PhD (Member)

17 AUG 2011  
Date

  
\_\_\_\_\_  
Richard Raines, PhD (Member)

17 Aug 2011  
Date

Accepted:

  
\_\_\_\_\_  
M. U. Thomas, PhD  
Dean, Graduate School of  
Engineering and Management

25 Aug 2011  
Date

**Abstract**

The purpose of our research is to develop an improved methodology for classifying players (identifying deviant players such as terrorists) through multivariate analysis of data from avatar characteristics and behaviors in massive multiplayer online games (MMOGs). To build our classification models, we developed three significant enhancements to the standard Generalized Regression Neural Networks (GRNN) modeling method. The first enhancement is a feature selection technique based on GRNNs, allowing us to tailor our feature set to be best modeled by GRNNs. The second enhancement is a hybrid GRNN which allows each feature to be modeled by a GRNN tailored to its data type. The third enhancement is a spread estimation technique for large data sets that is faster than exhaustive searches, yet more accurate than a standard heuristic. We applied our new techniques to a set of data from the MMOG, *Everquest II*, to identify deviant players ('gold farmers'). The identification of gold farmers is similar to labeling terrorists in that the ratio of gold farmer to standard player is extremely small, and the in-game behaviors for a gold farmer have detectable differences from a standard player. Our results were promising given the difficulty of the classification process, primarily the extremely unbalanced data set with a small number of observations from the class of interest. As a screening tool our method identifies a significantly reduced set of avatars and associated players with a much improved probability of containing a number of players displaying deviant behaviors. With further efforts at improving computing efficiencies to allow inclusion of additional features and observations with our framework, we expect even better results.

# Table of Contents

	Page
Abstract.....	iv
List of Figures.....	vii
List of Tables.....	ix
I. Introduction.....	1
II. Background.....	7
2.1. Literature Review on Player Classification.....	7
2.2. Overview of Artificial Neural Networks for Classification.....	14
2.2.1. Artificial Neural Networks.....	16
2.2.2. Preparing Data for Generalized Regression Neural Networks.....	24
2.2.3. Artificial Neural Network Performance Measures.....	26
2.3. Summary.....	31
III. Feature Selection for Player Classification.....	32
3.1. Review of Select Feature Selection Techniques.....	33
3.2. New Feature Selection Technique.....	35
3.2.1. Technique Overview.....	36
3.2.2. Gradient of Generalized Regression Neural Networks.....	40
3.3. University of Wisconsin Breast Cancer Data Example.....	45
3.3.1. Data description.....	45
3.3.2. Analysis.....	47
3.4. Conclusion.....	50
IV. Hybrid Generalized Regression Neural Network for Classification.....	52
4.1. Background.....	52
4.2. Methodology.....	53
4.3. Bot Traffic Example.....	54
4.3.1. Original TCP Analysis.....	57
4.3.2. Initial TCP Analysis with Artificial Neural Networks.....	61
4.3.3. Second TCP Analysis with Artificial Neural Networks.....	64
4.3.4. Conclusion.....	68
4.4. Summary.....	69
V. Spread Estimation for Classification with Large Data Sets.....	70
5.1. Background.....	71
5.2. Methodology.....	73
5.3. Multiple Data Sets Comparisons.....	73
5.3.1. Data Overview.....	74

5.3.2. Analysis.....	75
5.3.3. Conclusion.....	78
5.4. Summary.....	79
VI. Analysis of ‘Gold Farmers’ in <i>EverQuest II</i> Using Feature Selection and Hybrid Generalized Regression Neural Networks .....	80
6.1. Data Description.....	81
6.2. Issues Relating to Large Data Set Sets .....	83
6.3. Analysis .....	86
6.4. Conclusion.....	977
VII. Summary Research Contributions and Future Research.....	100
7.1. Research Contributions .....	100
7.1.1. Feature Selection Using Generalized Regression Neural Networks.....	100
7.1.2. Hybrid Generalized Regression Neural Network .....	101
7.1.3. Spread Estimation Technique for Large Data Sets .....	102
7.1.4. Develop Framework to Classify Players by Predetermined Categories Using Information Obtained Through In-Game Behaviors.....	103
7.2. Recommendations for Future Work .....	103
7.2.1. Develop Tool to Identify In-Game Player Associations and Movement Patterns .....	104
7.2.2. Feature Selection Improvements Using Generalized Regression Neural Networks .....	104
7.2.3. Processing Improvements for Hybrid Generalized Regression Neural Networks .....	105
7.2.4. Spread Estimation for Large Data Sets .....	105
7.2.5. Further Develop Framework to Classify Players by Predetermined Categories Using Information Obtained Through Observed Behaviors .....	106
Appendix A: List of Acronyms.....	108
Appendix B: Binary Receiver Operating Characteristic Curves for EQ2 Analysis .....	109
Appendix C: Features Reduced vs. F-Measure for EQ2 Analysis.....	116
Bibliography .....	121



## List of Figures

	Page
Figure 1: “On the Internet, nobody knows you’re a dog” (Steiner, 1993).....	2
Figure 2: Flow Chart for Typical Modeling Using Generalized Regression Neural Networks .....	5
Figure 3: Flow Chart for Modeling using Hybrid Generalized Regression Neural Network with Feature Reduction .....	5
Figure 4: Bartle Personality Interest Graph (Bartle, 1996).....	8
Figure 5: Example Feed Forward Neural Network.....	17
Figure 6: Example Radial Basis Function Neural Networks .....	18
Figure 7: Two Category Generalized Regression Neural Network .....	19
Figure 8: Contour Plots of a Parzen Windows Distribution .....	20
Figure 9: Confusion Matrix .....	27
Figure 10: Confusion Matrix Notional Example .....	27
Figure 11: Contour plot of F-Measure related to recall and precision.....	30
Figure 12: Receiver Operating Characteristic Curve Notional Example.....	31
Figure 13: Plot of apparent classification accuracies after stepwise feature reduction ....	34
Figure 14: Flow Chart Depicting Generalized Regression Neural Network Feature Reduction for Numeric and Binary Data Types.....	36
Figure 15: Plot of apparent classification accuracies versus number of features removed for University of Wisconsin Breast Cancer Data.....	49
Figure 16: Flow Chart of the Hybrid Generalized Regression Neural Network .....	54

Figure 17: Lines from a Transmission Control Protocol Trace .....	55
Figure 18: Histogram of client response times shorter than 0.5 seconds (Chen, 2009) ...	58
Figure 19: Evaluation results for the proposed decision schemes with different input size (Chen, 2009).....	59
Figure 20: Evaluation results for the integrated schemes (Chen, 2009).....	60
Figure 21: ROC curve of Hybrid GRNN for Bot detection.....	64
Figure 22: Flow Chart of Faster Spread Finding For Extremely Large Data Sets .....	73
Figure 23: ROC Curves with F-Measure for EQ2-B1 Feature Reduction Sets.....	88
Figure 24: Plot of Retained Features vs. F-Measure for EQ2-B1.....	89
Figure 25: Plot of Retained Features vs. F-Measure for Numeric Data Set .....	90
Figure 26: ROC Curves with F-Measure for Final Results.....	92
Figure 27: ROC Curves for EQ2-B1 feature reduction sets. ....	109
Figure 28: ROC Curves for EQ2-B2 feature reduction sets. ....	110
Figure 29: ROC Curves for EQ2-B3 feature reduction sets. ....	111
Figure 30: ROC Curves for EQ2-B4 feature reduction sets. ....	1144
Figure 31: ROC Curves for EQ2-B5 feature reduction sets. ....	115
Figure 32: Plot of Retained features vs. F-Measure for Numeric Data Set .....	116
Figure 33: Plot of Retained features vs. F-Measure for EQ2-1 Binary Data Set.....	117
Figure 34: Plot of Retained features vs. F-Measure for EQ2-2 Binary Data Set.....	118
Figure 35: Plot of Retained features vs. F-Measure for EQ2-3 Binary Data Set.....	118
Figure 36: Plot of Retained features vs. F-Measure for EQ2-4 Binary Data Set.....	119
Figure 37: Plot of Retained features vs. F-Measure for EQ2-5 Binary Data Set.....	120

## List of Tables

	Page
Table 1: Summary from 28 medical studies (Sargent, 2001) .....	15
Table 2: Alternate Parzen Windows Kernels from (Specht, 1990) .....	21
Table 3: Feature Overview of Breast Cytology Data.....	46
Table 4: Summary of Breast Cytology Feature Selection .....	48
Table 5: Transmission Control Protocol Traces Summary (Chen, 2009).....	56
Table 6: Feature Overview of Initial Transmission Control Protocol Data from <i>Ragnarok Online</i> .....	62
Table 7: Hybrid confusion matrices for Bot detection.....	63
Table 8: Comparative Confusion Matrices for Bot Detection .....	64
Table 9: Feature Overview of Second Transmission Control Protocol Data from <i>Ragnarok Online</i> .....	65
Table 10: Hybrid Confusion Matrices for Bot Detection .....	66
Table 11: Comparative Confusion Matrices for Bot Detection .....	67
Table 12: Times for Creating Confusion Matrices for Bot Detection .....	68
Table 13: EQ2 data sets with the number of features and observations .....	75
Table 14: Spreads by Collection Type and Data Set .....	76
Table 15: Spread Collection Times by Collection Type and Data Set .....	76
Table 16: Spread Collection Times Relative to Exhaustive Search Collection Time by Collection Type and Data Set .....	77
Table 17: Spread collection times relative to normal collection time along with spread values.....	77

Table 18: Feature Overview of the 1% Sample Data Collected From <i>EverQuest II</i> .....	82
Table 19: Binary Feature Reduction EQ2-B1.....	89
Table 20: Summary of Feature Reduction for EQ2 Binary Data.....	90
Table 21: Numeric Feature Reduction.....	91
Table 22: Results from Convex Combination .....	92
Table 23: Model Parameters .....	93
Table 24: Final Model Confusion Matrix .....	93
Table 25: Retained Features.....	95
Table 26: 500/500 Model Resulting Confusion Matrix.....	96
Table 27: 5% Model Resulting Confusion Matrix.....	96
Table 28: Numeric Feature Reduction.....	116
Table 29: Binary Feature Reduction EQ2-B1.....	117
Table 30: Binary Feature Reduction EQ2-B2.....	117
Table 31: Binary Feature Reduction EQ2-B3.....	118
Table 32: Binary Feature Reduction EQ2-B4.....	119
Table 33: Binary Feature Reduction EQ2-B5.....	120

# IDENTIFICATION AND CLASSIFICATION OF PLAYER TYPES IN MASSIVE MULTIPLAYER ONLINE GAMES USING AVATAR BEHAVIOR

## I. Introduction

Playing massive multiplayer online games (MMOGs) is one of the most popular activities in the world today. A MMOG is “a type of computer game that enables hundreds or thousands of players to simultaneously interact in a game world which they are connected to via the Internet (Game Entertainment Europe, 2008).” Millions of people around the world play MMOGs, one of the most successful games, *World of Warcraft (WoW)*, claims over 12M players (Blizzard Entertainment, 2010). Players can shop, talk, stage combat, and explore with people they may have never met in person. To access these game worlds, a player creates a computer generated character called an avatar, a virtual representation of the player. These avatars are developed and controlled by the player. The players then use these avatars to interact with the virtual world and other avatars.

It is commonly believed that using these avatars provides anonymity, as seen in Figure 1, a comic from the New Yorker (Steiner, 1993). This anonymity can be a blessing and a curse. It is a blessing in that it allows people to interact without worry of being judged on personal appearance, race, or handicap. It also affords the freedom to act as you wish knowing you do not need to worry about how you are perceived; you can always delete an avatar and create a new one to start over. It is a curse in that a side effect of this anonymity and the ease of communication afforded by MMOGs is they

have become virtual hot beds of criminal activity which includes trafficking in credit card numbers (The City Paper, 2008), cheating (Laurens, 2007; Yan, 2002), espionage (BBC News, 2009), and grieving (Lin, 2005), which can be seen as a form of terrorism. It is possible, that monitoring avatar grieving behaviors in MMOGs could lend insight into understanding real life terrorists, much like epidemiologists who are attempting to use *WoW* data to study people's behavior during epidemic outbreaks (Balicer, 2007).

Similarly, the deputy director of the Center for Terrorism and Intelligence Studies states that he believes such a game could provide ways to study how terrorist cells form and operate. He believes that the use of MMOG avatars add a realistic dimension to study terrorists' tactical decision-making and may generate more useful information than a standard simulation (Their, 2008). However, before we can study their actions, we need to find the people we wish to study within the MMOG who rely on the anonymity.



**Figure 1: “On the Internet, nobody knows you’re a dog” (Steiner, 1993)**

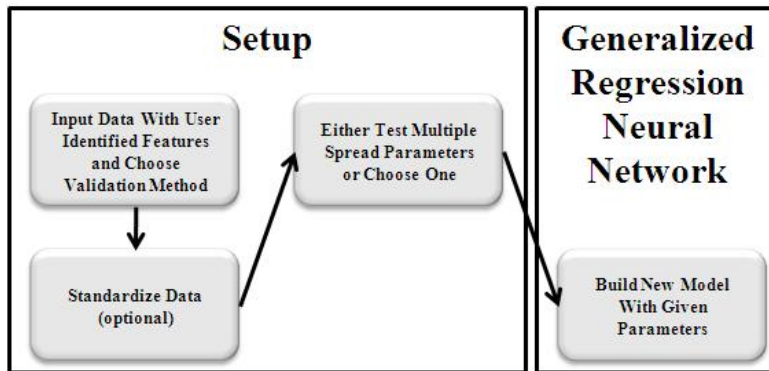
This research focuses on chipping away at this avatar anonymity. We identify behaviors and characteristics from avatars and use them to build models to classify players. We believe that since there is someone controlling the avatar, the avatar will display specific characteristics that can be used to identify the player behind the avatar. This kind of information can be used by game companies to identify what kind of players play their games and could aid them in developing more content for those players or developing additional content to bring in players. This kind of information can also be used to identify deviant behaviors such as ‘gold farming.’ Gold farming refers to the practice of trading virtual in-game resources such as currency, items, and avatars for real-world currency. Gold farming is considered a deviant behavior for three main reasons (Keegan, 2010). First, in game economies are carefully developed by the game developers and gold farmers upset the balance of these economies. Second, gold farmers activities often adversely affect the playing experience of other players. Third, gold farming assigns a real-world value to virtual property bringing with it questions about property rights and taxation along with criminal activities such as money laundering.

MMOG information can also be used to identify criminals and criminal activities, ranging from information exchanges to terrorist activities. The identification of gold farmers is similar to labeling terrorists in that the ratio of gold farmer to standard player is extremely small, and the in-game behaviors for a gold farmer have detectable differences from a standard player. This research can be not only applied to the Department of Defense (DoD) problem of identifying terrorists in online games, but onto other monitored systems such as video surveillance of a public area. Similarities can be drawn between video surveillance and avatar observation within an MMOG to include

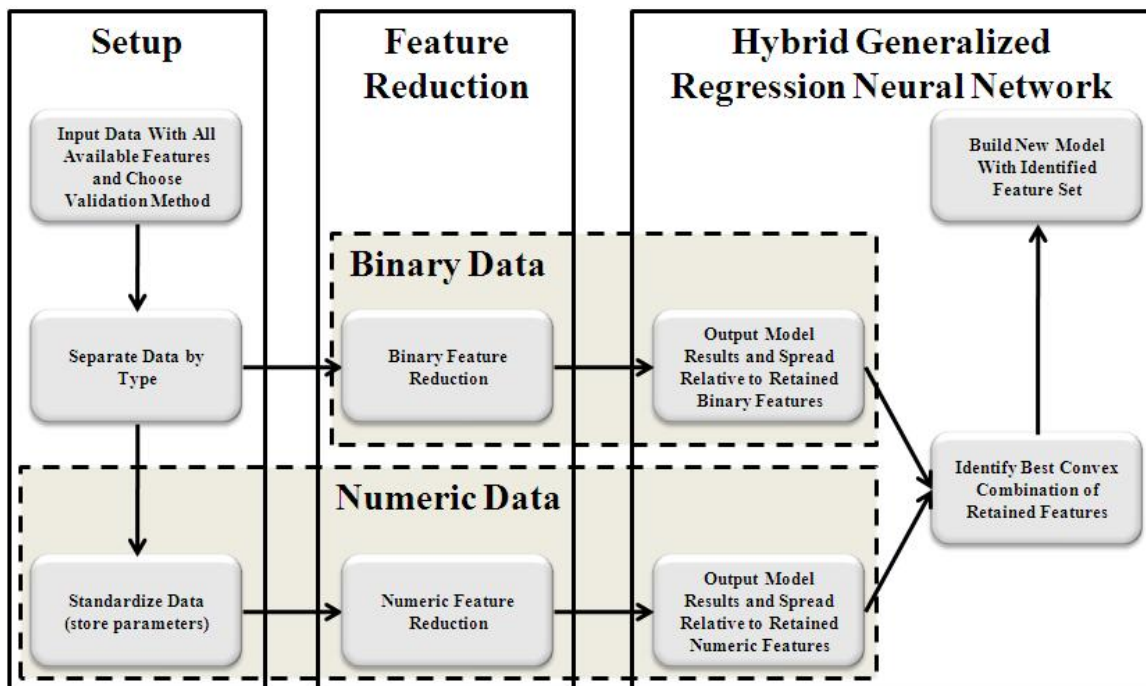
movement path, appearance, race, gender, and player interaction. These observations represent behavioral features which cannot easily be modeled as a linear combination of independent features, leading us to consider this data as non-linear.

Since we believe the MMOG data would be best modeled with non-linear techniques, we employ Artificial Neural Networks (ANNs), based on their proven ability to model non-linear data (Loeffelholz, 2009). Generalized Regression Neural Networks (GRNNs) are the primary tools we use to develop the player classification models. Specifically, our approach begins with all available features in a data set and then builds a GRNN to reduce the number of features used for classification. An enhancement with our approach, handles multiple data types with tailored feature reduction techniques, hence our label of hybrid GRNN. As an example, avatar behavior data consists of numeric, binary, and categorical data. The numeric and binary data are readily imported into mathematical models, but categorical data needs to be converted into a numeric form. Examples of categorical data are gender, hair color, and avatar profession. Converting categorical data with more than two options such as hair color (red, blue, green, and brown) into numeric form is done by taking each feature within the categorical feature and creating a new binary feature. For our hair color case, we would then have red, blue, green and brown as binary features. Converting categorical data with only two options such as gender (male and female) into numeric form can be done by assigning each value to a binary switch. For our gender case, we would have the gender feature with 0 equaling male, and 1 equaling female. Figure 2 is a flow chart showing the typical way to model data using a GRNN while Figure 3 illustrates our hybrid GRNN approach, including a feature reduction step, used in this research.





**Figure 2: Flow Chart for Typical Modeling Using Generalized Regression Neural Networks**



**Figure 3: Flow Chart for Modeling using Hybrid Generalized Regression Neural Network with Feature Reduction**

The first chapter of this document is an introduction to give the reader a brief overview of the research. The second chapter focuses on the research background which covers a literature review and an overview of ANNs. The third chapter gives a detailed

description of our feature selection method based on GRNNs and includes an example using University of Wisconsin Breast Cancer Data (Breast Cancer Wisconsin (Original) Data Set, 1992). The fourth chapter covers our hybrid GRNN developed to handle multiple data types and includes an example using data derived from MMOG packet communication data (Chen, 2009). The fifth chapter discusses issues in parameter selection for the GRNN when applied to large data sets and includes a comparison of different methods and data sizes. The sixth chapter is a full analysis using the method outlined in Figure 3 on data obtained from the MMOG *EverQuest II* with a focus in identifying the criminal activity of gold farming. The seventh and final chapter is a summary of this research and covers the research contributions.

## II. Background

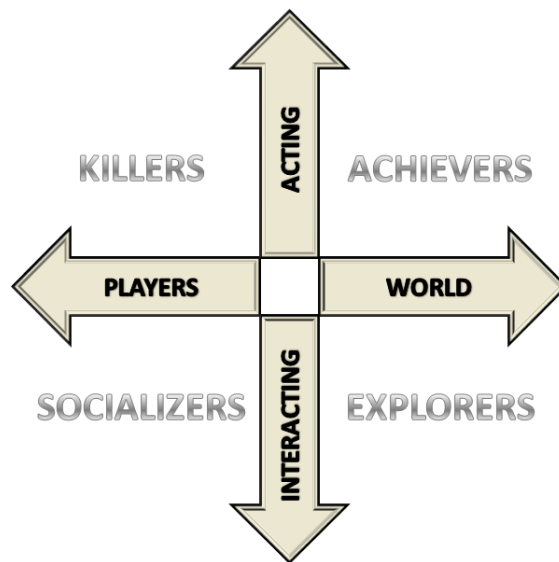
This chapter presents a literature review on player classification and discusses Artificial Neural Network (ANN) background and use for classification. The player classification section covers a variety of approaches used to identify and classify players in many different Massive Multiplayer Online Games (MMOGs). The ANN section covers the Feed Forward Neural Network (FFNN) and the Generalized Regression Neural Network (GRNN) as well as techniques to compare the effectiveness of ANNs.

### 2.1. Literature Review on Player Classification

Classifying a player in an online game is a difficult challenge. The limited combination of avatar appearances and the limited number of actions shroud the individuality of the player. The generic nature of the avatar, combined with the ability to alter ones responses and actions with limited recourse, make classifying the player behind the avatar difficult. Researchers are attempting to identify unique behaviors associated with a player controlling the avatar. Some of the observable information used to identify these unique behaviors are network activity, action sequences, chat, movement, and avatar location within the game. A goal for researchers is to develop methods to classify players using the observable avatar characteristics and behaviors.

Noted as one of the earliest works in player classification, in 1996 Richard Bartle developed four personality types to classify Multi User Domain (MUD) players. As seen in (Bartle, 1996), the four Bartle personality types are *Socializer*, *Explorer*, *Achiever*, and *Killer*. Using the interest graph in Figure 4, we can see that *Socializers* value interacting

with other players. This generally means they would prefer talking to others and getting to know them, rather than competing against them. They tend to use the game worlds as a setting, and other people are reasons to be there. *Explorers* are interested in interacting with the world. They enjoy traveling through the virtual environment and discovering new areas, creatures and adventures. They find value in other players as a way to share their knowledge of the game, but other players are not essential to game play. *Achievers* value acting on the world. This means their goal is to master the game. Knowledge about the game only has value if it leads to successfully completing some part of the game. They also find other players lend to authenticity, but are not necessarily important. *Killers* enjoy acting on other players. They are highly competitive and find that knowledge about the world is not important unless it gives them an advantage over other players. They are usually focused on perfecting skills and techniques that are applied when competing against other players.



**Figure 4: Bartle Personality Interest Graph (Bartle, 1996)**

Although the Bartle personality types were developed for MUDs, they also encompass MMOG players. There is even a test, developed by Erwin Andreason, currently available to be taken to determine a player's personality type (GamerDNA, 2009). The Bartle's Personality Test is used as a fun way for players to classify themselves. Even though it is labeled for entertainment purposes only, the gaming community feels that the results are representative of their individual gaming personalities.

Dr. Ruck Thawonmas, a professor at Ritsemeikan University in Japan, has worked a lot with player classification. Thawonmas et al. researched classification of players' types in order to identify player behaviors and assist game developers in developing content to fulfill player demands. For their research, they use MMOG simulators *Zereal* and *Simac* and MMOGs *The Ice* and *Angels Love*. *Zereal* is "a MMOG simulation platform that provides a (coarse) simulation of active players that can be used to test various approaches for player usage logging (Tveit, 2003)." It was developed at the Norwegian University of Science and Technology. *Zereal* uses Markov models to simulate player actions. *Simac* is a MMOG simulator designed to simulate player types and actions not available in *Zereal*. *The Ice* is an educational game developed at the Intelligent Computer Entertainment Lab at Ritsemeikan University in Japan. *Angel's Love* is a commercial MMOG. It is free to play and initially was released in Taiwan. A Japanese version is also available.

Thawonmas, Ho, and Matsumoto use logs of action sequences and item sequences from *Zereal* to classify a player's Bartle personality type (Thawonmas, 2005a). Action and item sequences are aggregated into the probability of specific actions for each player

type. These probability strings are classified using Adaptive Memory-Based Reasoning (AMBR) which performs majority voting among the  $k$  nearest neighbors and incrementing  $k$  if ties occur. Their results show around a 92% classification rate using three different types of *Zereal* agents.

Thawonmas and Ho use an Action Transition Probability Matrix (ATPM) and Kullback Liebler Entropy (KLE) to classify action logs from *Simac* (Thawonmas, 2007a). KLE is a distance measure used for comparing similarity between probability distributions. Each log is parsed into an ATPM to identify the probability from moving from one action state to another. Then a training set is used to classify a Bartle personality type with the KLE nearest neighbors. Their results show around a 95% classification rate across all four Bartle personality types.

Thawonmas and Hata use symbol sub-sequences and KeyGraph to analyze players' action behaviors (Thawonmas, 2005b). KeyGraph is “an algorithm for extracting keywords representing the asserted main point in a document, without relying on external devices such as natural language processing tools or a document corpus (Ohsawa, 1998).” KeyGraph uses indexing based on information within the documents such as term frequency and location (Ohsawa, 1998). Thawonmas and Hata identify an algorithm that aggregates frequent sub-sequences of consecutive actions. Then, KeyGraph is used to identify the co-occurrence of actions from the reduced sequence. Unknown KeyGraphs are then compared to known KeyGraphs by observation. Using *Zereal*, they are able to achieve a classification rate of 90% using three types of simulated agents.

Thawonmas and Matsumoto use Hidden Markov Models (HMMs) of player action sequences to classify players of MMOGs (Matsumoto, 2004; Thawonmas, 2005c). A set of training action sequences is turned into a training set of HMMs. Then, using the Viterbi algorithm, unknown action sequences are assigned to a training HMM with the highest log probability. Using *Zereal*, they are able to classify specific simulated agent types with a success rate between 80% and 100% depending on the agent type.

Thawonmas, Kurashige, Iizuka, and Kantardzic use Self-Organizing Maps (SOMs) to cluster online-game users based on their player trails (Thawonmas, 2006). They simulate player trails using a 2D map with a 600x600 grid, derived from an online game map. They then use a SOM to cluster users based on their movement patterns. They conclude that a SOM was able to be generated from online-game trails and could successfully cluster users based on these trails. Since this was a proof of concept, no classification performance experiment has been done.

Thawonmas and Iizuka use player action logs from *The ICE* to apply Classical Multidimensional Scaling (CMDS) and KeyGraph to analyze players' action behaviors (Thawonmas, 2008). CMDS is used to cluster the logs and identify players with similar logs. CMDS is a technique for mapping pair-wise relationships to coordinates. Then, KeyGraph is used to identify the co-occurrence of actions. Using the KeyGraph data each cluster is related to a Bartle personality type. This is done by identifying actions that relate to each Bartle type. They conclude they were successful in identifying three clusters from *The ICE* game logs, which matched the Bartle Player types of *Achievers*, *Explorers*, and *Socializers*.

Thawonmas and Iizuka also use Haar wavelets and Dynamic Time Warping (DTW) to classify action logs (Thawonmas, 2007b). The size of the log is reduced by using Haar wavelet transformation. These reduced logs are then compared to a training set using  $k$  nearest neighbor and the DTW distance. DTW distance is used for deriving distance between time series data. Using logs from *The ICE* they separated players into three groups. The groups are each assigned a series of three tasks to be performed in order. The difference between the three groups is the order of the three tasks. They are able to classify players from each of the three groups. They do not give a specific analytical result, but they did supply many confusion matrices. Looking at the confusion matrices, their best classification rate is 83% across the three groups.

Thawonmas, Kurashige, and Chen use an algorithm to detect landmarks and use player transition probability matrices from *The Ice* and *Angel's Love* game logs to cluster players (Thawonmas, 2007c). They develop an algorithm that divides up the game map and associates a square with a high amount of traffic with a landmark. Player trails are then used to develop player transition probability matrices from between the identified landmarks and CMDS is used to cluster players. They claim their “evaluation results confirmed that [their] approach successfully identified player clusters having different movement patterns (Thawonmas, 2007c).”

Leuski and Lavrenko use statistical language modeling and text clustering techniques to explore connections between human activities and the content of textual information regarding those activities (Leuski, 2006). They synchronized chat and game logs from the commercial online game *BladeMistress*. Using hypothesis testing, they try to identify a connection among avatar chat, actions, location, and time. They note that



they are able to classify large monster kills with a 90% true positive rate from message content, and a 60% true positive rate for smaller monster kills.

Chen and Hong use avatar idle time as a biometric identification of a player (Chen, 2007). They use the time between avatar movements in *Angel's Love*. The idle time includes chat and character maintenance. They suggested using a one-sided Wilcoxon test on the Kullback-Leiber divergence between two idle time distributions to identify if both distributions come from the same player. They conclude that a larger history size and longer detection time increases the accuracy of determining between two players to over 90%.

Chen, Jiang, Huang, Chu, Lei, and Chen use internet packet information from the commercial online game *Ragnarok Online* to identify bots vs. human players (Chen, 2009). A bot is a computer program designed to play a game instead of a human. They are generally used to perform repetitive tasks that a player needs to do for a game but doesn't want to spend the time. Chen et al. use a combination of analyzing client response times, burstiness trends, burstiness magnitude and reaction to network conditions. Burstiness is the variability of packet counts sent in successive order. They are able to achieve a 95% correct detection rate with a high false positive rate of about 40%.

Ahmad, Keegan, Srivastava, Williams and Contractor investigate identifying gold farmers using a multistage approach (Ahmad, 2009). A gold farmer in this investigation is a bot created to kill in-game monsters to earn gold to be sold to other players. This kind of activity is generally banned by MMOGs due to the fact it depreciates the in-game economy. As part of The Virtual World Exploratorium Project, they have access to three

years of player data from one of the largest commercial MMOGs, *EverQuest II*. The multistage approach consists of using a deductive logistic multiple regression model to identify specific traits of gold farmers. Then, an inductive evaluation of binary classifiers such as Naive-Bayes,  $k$ -Nearest Neighbors, Bayesian Networks, and Decision Trees is used to correctly identify gold farmers. The results show a true positive rate of around 20% with a false positive rate practically nonexistent.

From this discussion on research in classification of players in MMOGs, three major themes arise. First, the majority of the research that has been done consists of using simulators instead of actual MMOG data. This can skew data since this approach uses mathematical techniques to identify preprogrammed behaviors defined by mathematical equations to represent a specific category. Second, throughout the research, researchers assign specific behaviors/attributes to a category of player based upon assumptions. These assumptions can easily mislead classification efforts since the actual significant features are unknown. Third, none of the approaches examined in the literature use ANNs which have been shown to be effective in classifying nonlinear data.

## **2.2. Overview of Artificial Neural Networks for Classification**

ANNs are widely used in pattern recognition. They have been shown to be ideal for modeling nonlinear systems as seen in Loeffelholz (2009). Two main attributes of ANNs are they have no assumption that the features are linearly related to the output like regression, and they have no assumptions of distribution, giving them the freedom to be adapted to any system. Sargent (2001) conducted research in comparison between ANNs and linear regression. In his article, he examines 29 medical studies that use both ANNs and regression for their analysis. He uses their results to compare the performance

between ANNs and regression. He concludes that from these results, there is no conclusive evidence that ANN or regression is better than the other. Looking at Table 1, we can see that 4/29 studies (14%) favored regression, 13/29 studies (45%) were tied, and 12/29 studies (41%) favored ANNs. With these results, it would seem that using ANNs would be a better option since 86% of the time it would have the same or better results than using regression.

**Table 1: Summary from 28 medical studies (Sargent, 2001)**

First author	Regression	ANN	n	Validation method	Result
Lippmann <sup>14</sup>	LR	BP	80,600	50% split	EQUIV
Ennis <sup>15</sup>	LR	BP	41,021	66% split	EQUIV
Warner <sup>16</sup>	LR	BP	32,092	66% split	REGR
Cooper <sup>17</sup>	LR	BP	14,199	70% split	EQUIV
Burke <sup>18</sup>	LR	BP	8271	63% split	EQUIV
Selker <sup>19</sup>	LR	BP	5773	60% split	EQUIV
Rowland <sup>20</sup>	LR	BP	5626	66% split	EQUIV
Goodman <sup>21</sup>	LR	Not stated	5516	Bootstrap	EQUIV
Duh <sup>22</sup>	LR	BP	1674	50% split	EQUIV
Ravdin <sup>23</sup>	CR	BP	1590	50% split	EQUIV
Ravdin <sup>24</sup>	CR	BP	1373	66% split	EQUIV
Eisenstein <sup>25</sup>	LR	Not stated	1139	55% split	REGR
Lapuerta <sup>26</sup>	LR	BP	1081	52% split	ANN
Virtanen <sup>27</sup>	LR	BP	974	5-fold cv	REGR
Zemikow <sup>28</sup>	LR	Adaptive gradient	890	50% split	ANN
Zemikow <sup>29</sup>	LR	Descent gradient	890	50% split	ANN
Michie <sup>30</sup>	Logistic	BP	768	12-fold cv	REGR
Jefferson <sup>31</sup>	LR	Genetic	620	Leave 1 out	ANN
Ohno-Machado <sup>32</sup>	CR	BP	588	10-fold cv	EQUIV
		3 ANNs (BP, generalized regression, and probabilistic)			
Buchman <sup>33</sup>	LR		491	66% split	ANN
Faraggi <sup>34</sup>	CR	BP	475	50% split	EQUIV
Kattan <sup>35</sup>	CR	BP	424	66% split	ANN
Doig <sup>36</sup>	LR	BP	422	66% split	EQUIV
Dybowski <sup>37</sup>	LR	Genetic	422	60% split	ANN
Lette <sup>38</sup>	LR	BP	360	55% split	ANN
Marchevsky <sup>39</sup>	LR	Genetic	279	80% split	ANN
Rae <sup>40</sup>	LR	BP	274	66% split	ANN
Hammad <sup>41</sup>	LR	BP	251	80% split	ANN
Biagiotti <sup>42</sup>	LR	BP	226	5-fold cv	ANN

LR: logistic regression; BP: backpropagation; CV: cross-validation; EQUIV: methods performed equivalently; REGR: regression outperformed ANN; ANN: ANN outperformed regression; CR: Cox regression.

The remainder of this section is broken into three subsections. The first subsection covers the three common ANNs, FFNNs, Radial Basis Function Neural Networks (RBFNNs), and GRNNs. The second subsection covers how to prepare data

for the GRNN, including standardization and validation techniques. The third subsection covers how to rate the ANN performance. It covers confusion matrices, performance measures, and receiver operating characteristic (ROC) curves.

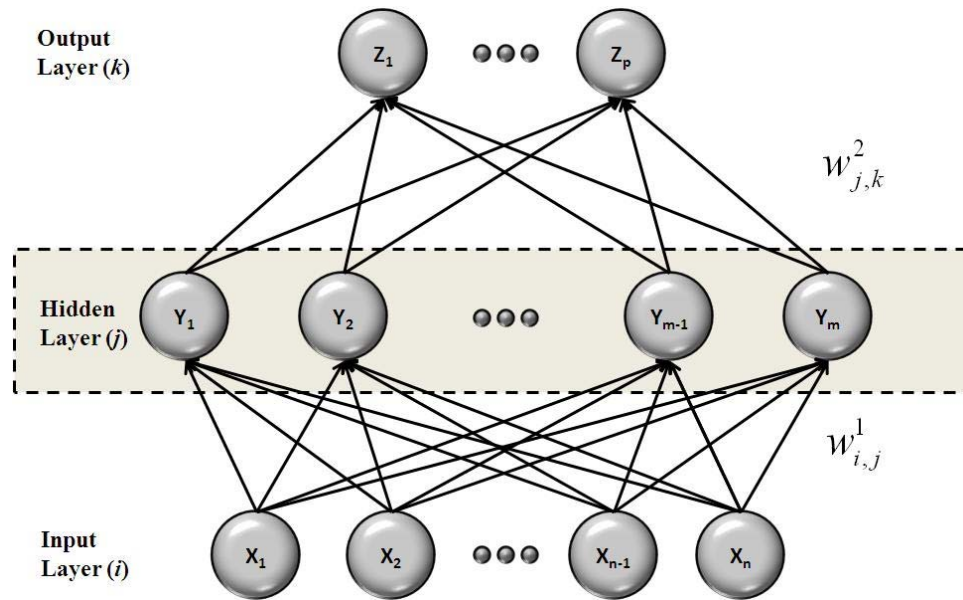
### **2.2.1. Artificial Neural Networks**

ANNs are inspired by biological systems and are modeled as a collection of artificial neurons, as seen in Figure 5, Figure 6, and Figure 7. The collections of artificial neurons make ANNs ideal for modeling nonlinear systems. Neural networks are computationally intensive, but as computers have advanced, they have allowed results to be obtained faster and have enabled the use of massive data sets.

Three common ANNs are FFNNs (Figure 5), RBFNNs (Figure 6), and GRNNs (Figure 7). FFNNs are very common, but require a large amount of time to train and assign weights to the neurons. RBFNNs are similar to FFNNs in that the neurons have weights, but they have fewer weights to train and each neuron is assigned a distribution. GRNNs are similar to RBFNNs in that each neuron is assigned a distribution, but there are no weights to train, making them relatively fast compared to RBFNNs and FFNNs.

FFNNs consist of at least three layers. They are an input layer, a hidden layer and an output layer. The input layer consists of each feature and can come in any numbered format. The hidden layer may consist of one or more interconnected layers. In our example (Figure 5) there is only one hidden layer. The number of hidden layers and neurons within each hidden layer is specified by the user. Each of the arcs connecting the nodes has a weight. The size of the weights depends on the significance of the feature. The weights are iteratively calculated using backpropagation to minimize the squared difference between the current predicted response and the true response. It may take

many cycles to identify good weights and the process may never finish training. Even if the training finishes, the resulting weights may not be optimal (Wasserman, 1989). The output layer can have one or more output nodes, depending on the response.

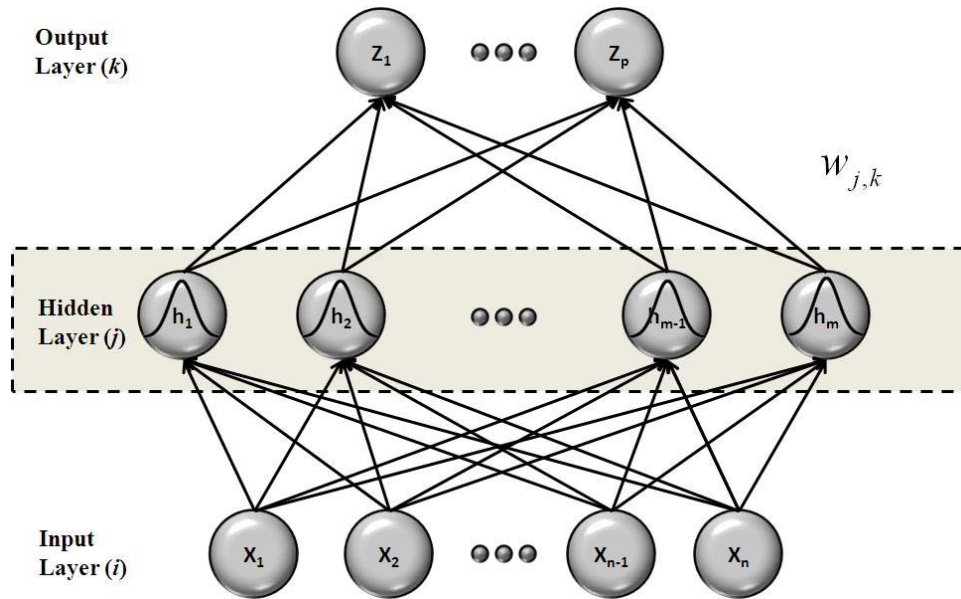


**Figure 5: Example Feed Forward Neural Network**

Even with the training problems, FFNNs are very effective. The main drawback of training a FFNN is the need for a user to monitor the training. This is mainly due to the random element to the training resulting in different models each time the data is trained. The resulting different models can also lead to complications when determining an optimal or effective number of hidden layers and neurons per layer. The probability of not finding optimal weights for each configuration of layers and neurons makes it difficult to compare the different configurations.

RBFNNs are comprised of a series of neurons that are represented by a Gaussian distribution centered on each point from the training set. During the development of the

model, individual weights are trained and assigned to each neuron. Each observation in the training set relates to a neuron. The RBFNN requires an input parameter to determine the spread of the Gaussian distributions. Determination of the input parameter is generally done by trial and error, but there are heuristics that have been developed to help determine this parameter.

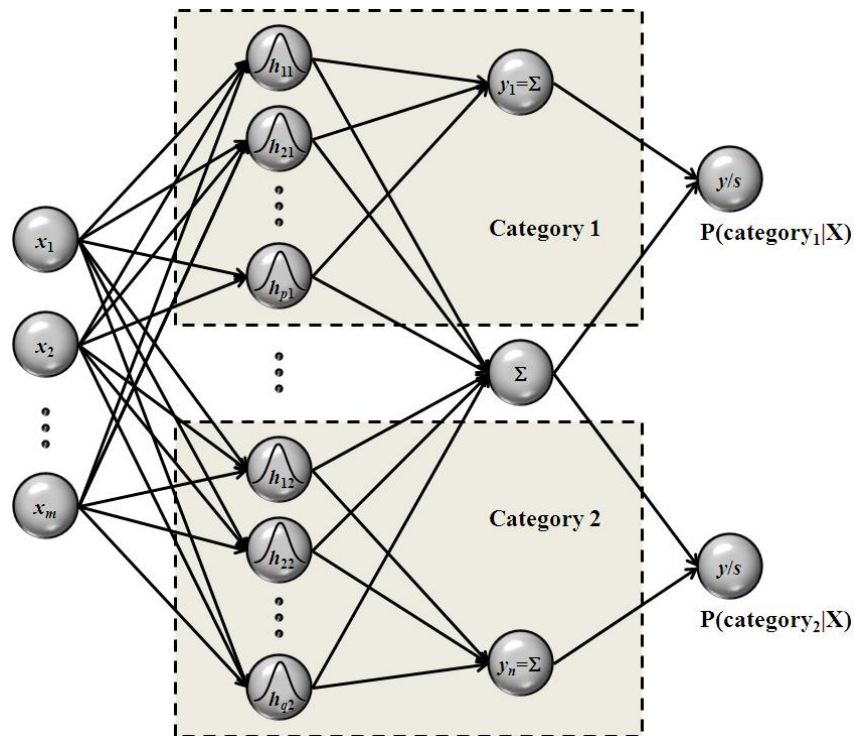


**Figure 6: Example Radial Basis Function Neural Networks**

A method similar to RBFNN is the GRNN. Developed in 1991 by Donald F. Specht (1991), GRNNs are one pass learning algorithms which are faster than FFNNs and does not need to train weights like RBFNNs and FFNNs. GRNNs were developed for regression, but can be easily adapted for use in classification as shown in Figure 7.

The basic idea behind GRNNs is to find an underlying distribution to a training set of data points with a known response. Then we can compare unknown data points to the distribution to identify the response. Parzen windows are used to develop the

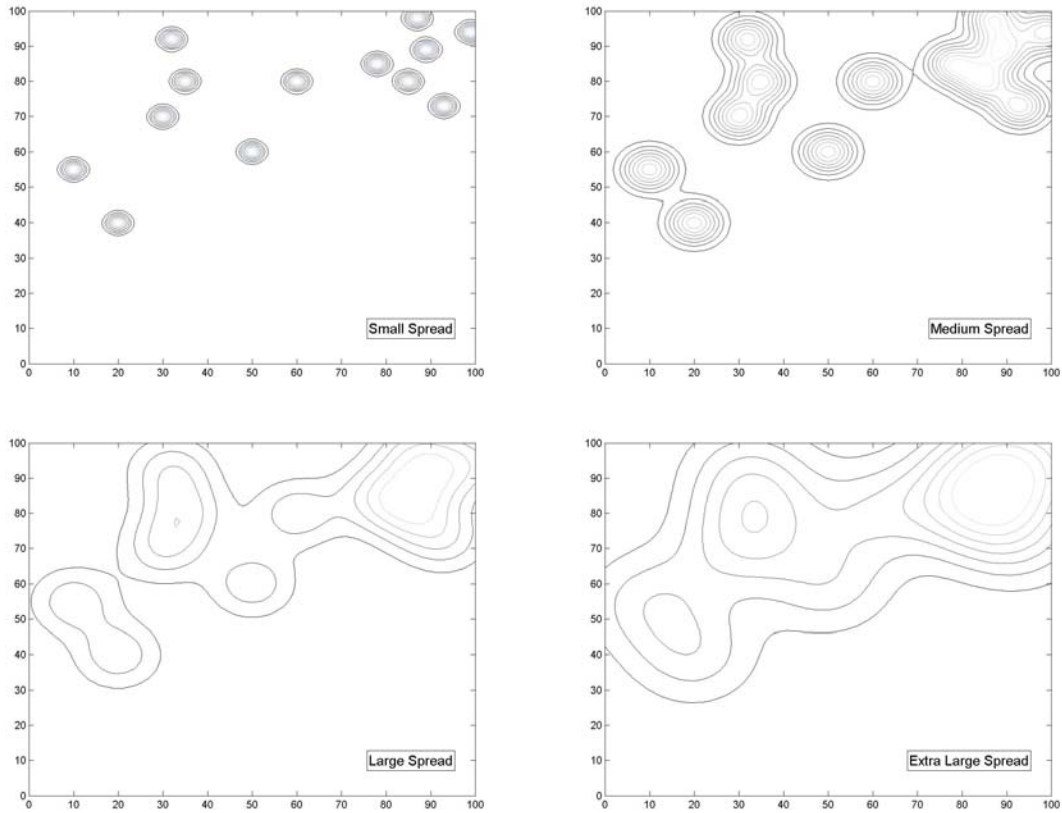
underlying distribution. This works by applying a Gaussian distribution to each data point within the training set. Then each point within the space is compared to the distributions created by the training set and the results are summed to obtain the estimated response value. Figure 8 is a representative example of what a Parzen windows distribution looks like. We can see individual points with a single Gaussian kernel in the foreground and in the background we can see what happens when points are close together and get added together.



**Figure 7: Two Category Generalized Regression Neural Network**

When used for classification, multiple distributions are created. Each of these distributions is associated to a category. Then, the unknown set of features is compared against each category. The value derived from each category is related to the probability

that the feature set came from that category. Figure 7 is a graphical representation of a classification GRNN with two categories.



**Figure 8: Contour Plots of a Parzen Windows Distribution**

Probabilistic Neural Networks (PNNs) are similar to GRNNs. They both generate Parzen windows to estimate the distribution of each category. In a previous article about PNNs (Specht, 1990), Specht noted that we could readily change the kernel of the Parzen windows with the application of different distance measures. He also noted that certain kernels may be better for certain data sets. He supplied a short list of different kernels that could be used. Table 2 displays all the kernels from Specht (1990) where  $n$  is the



number of observations,  $p$  is the number of features,  $\mathbf{X}$  is the test point,  $X_i$  is element  $i$  of  $\mathbf{X}$ ,  $\mathbf{X}_A$  is the training set,  $X_{Aij}$  is element from  $\mathbf{X}_A$  row  $i$  and column  $j$ , and  $\lambda$  is the spread of the function.

**Table 2: Alternate Parzen Windows Kernels from (Specht, 1990)**

$f_A(\mathbf{X}) = \frac{1}{n(2\lambda)^p} \sum_{i=1}^n 1, \quad \text{when all }  X_i - X_{Aij}  \leq \lambda$	(1)
$f_A(\mathbf{X}) = \frac{1}{n\lambda^p} \sum_{i=1}^n \prod_{j=1}^p \left[ 1 - \frac{ X_i - X_{Aij} }{\lambda} \right], \quad \text{when all }  X_i - X_{Aij}  \leq \lambda$	(2)
$f_A(\mathbf{X}) = \frac{1}{n(2\pi)^{2/p} \lambda^p} \sum_{i=1}^n \exp \left[ \frac{-\sum_{j=1}^p (X_i - X_{Aij})^2}{2\lambda^2} \right]$	(3)
$f_A(\mathbf{X}) = \frac{1}{n(2\lambda)^p} \sum_{i=1}^n \exp \left[ -\frac{1}{\lambda} \sum_{j=1}^p  X_i - X_{Aij}  \right]$	(4)
$f_A(\mathbf{X}) = \frac{1}{n(\pi\lambda)^p} \sum_{i=1}^n \prod_{j=1}^p \left[ 1 + \frac{(X_i - X_{Aij})^2}{\lambda^2} \right]^{-1}$	(5)
$f_A(\mathbf{X}) = \frac{1}{n(2\pi\lambda)^p} \sum_{i=1}^n \prod_{j=1}^p \left[ \frac{\sin \left( \frac{(X_i - X_{Aij})}{2\lambda} \right)}{\frac{X_i - X_{Aij}}{2\lambda}} \right]^2$	(6)

Equation (7) is a GRNN developed for numeric data using the Euclidian distance kernel, equation (3). Equation (8) is a GRNN developed for binary data using the Hamming distance kernel, equation (4). For these equations,  $n$  is the number of observations,  $p$  is the number of features,  $\mathbf{X}$  is the test vector of size  $p$ ,  $X_j$  is the  $j^{\text{th}}$  element from  $\mathbf{X}$ ,  $\mathbf{Z}$  is the training set of size  $n$  by  $p$ ,  $\mathbf{Z}_i$  is the  $i^{\text{th}}$  row vector from  $\mathbf{Z}$ , and  $Z_{ij}$

is element from  $\mathbf{Z}$  row  $i$  and column  $j$ ,  $\mathbf{Y}$  is a vector of size  $n$  containing training responses relative to  $\mathbf{Z}$ ,  $Y_i$  is the  $i^{\text{th}}$  element from  $\mathbf{Y}$ , and  $\sigma$  is the spread of the Euclidian function while  $\lambda$  is the spread of the Hamming (City Block) function.

$$\hat{Y}(\mathbf{X}) = \frac{\sum_{i=1}^n Y_i \exp\left[-\frac{(\mathbf{X}-\mathbf{Z}_i)^T (\mathbf{X}-\mathbf{Z}_i)}{2\sigma^2}\right]}{\sum_{i=1}^n \exp\left[-\frac{(\mathbf{X}-\mathbf{Z}_i)^T (\mathbf{X}-\mathbf{Z}_i)}{2\sigma^2}\right]} \quad (7)$$

$$\hat{Y}(\mathbf{X}) = \frac{\sum_{i=1}^n Y_i \exp\left[-\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}|\right]}{\sum_{i=1}^n \exp\left[-\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}|\right]} \quad (8)$$

A problem that can arise through the application of equation (7) in MATLAB is the distance measure within the exponential portion of the function can be large, causing the value within the exponential to become a large negative number. The large negative value can force the exponential to become small enough that MATLAB rounds the number to zero. This can cause a problem when the sum of exponentials in the denominator ends up equaling zero resulting in a divide by zero error. To fix this problem, we can adjust the distance measure by shifting all the distances so the largest (least negative) value is equal to zero. This will force an exponential in the sum of exponentials in the denominator to be equal to one, thus guaranteeing the denominator will be greater than zero. This will avoid a possible divide by zero, but will not affect the overall equation. The following is the proof that the addition of a shift variable  $\alpha$  does not affect the output from equation (7).

$$\hat{Y}(\mathbf{X}) = \frac{\sum_{i=1}^n Y_i \exp \left[ -\frac{(\mathbf{X} - \mathbf{Z}_i)^T (\mathbf{X} - \mathbf{Z}_i)}{2\sigma^2} - \alpha \right]}{\sum_{i=1}^n \exp \left[ -\frac{(\mathbf{X} - \mathbf{Z}_i)^T (\mathbf{X} - \mathbf{Z}_i)}{2\sigma^2} - \alpha \right]} \quad (9)$$

$$\hat{Y}(\mathbf{X}) = \frac{\sum_{i=1}^n Y_i \exp \left[ -\frac{(\mathbf{X} - \mathbf{Z}_i)^T (\mathbf{X} - \mathbf{Z}_i)}{2\sigma^2} \right] \exp(-\alpha)}{\sum_{i=1}^n \exp \left[ -\frac{(\mathbf{X} - \mathbf{Z}_i)^T (\mathbf{X} - \mathbf{Z}_i)}{2\sigma^2} \right] \exp(-\alpha)} \quad (10)$$

$$\hat{Y}(\mathbf{X}) = \frac{\exp(-\alpha) \sum_{i=1}^n Y_i \exp \left[ -\frac{(\mathbf{X} - \mathbf{Z}_i)^T (\mathbf{X} - \mathbf{Z}_i)}{2\sigma^2} \right]}{\exp(-\alpha) \sum_{i=1}^n \exp \left[ -\frac{(\mathbf{X} - \mathbf{Z}_i)^T (\mathbf{X} - \mathbf{Z}_i)}{2\sigma^2} \right]} \quad (11)$$

$$\hat{Y}(\mathbf{X}) = 1 \frac{\sum_{i=1}^n Y_i \exp \left[ -\frac{(\mathbf{X} - \mathbf{Z}_i)^T (\mathbf{X} - \mathbf{Z}_i)}{2\sigma^2} \right]}{\sum_{i=1}^n \exp \left[ -\frac{(\mathbf{X} - \mathbf{Z}_i)^T (\mathbf{X} - \mathbf{Z}_i)}{2\sigma^2} \right]} \quad (12)$$

Therefore to shift the largest distance to zero we set  $\alpha$  using equation (13).

$$\alpha = \max_l \left[ -\frac{(\mathbf{X} - \mathbf{Z}_l)^T (\mathbf{X} - \mathbf{Z}_l)}{2\sigma^2} \right] \quad (13)$$

The shift value is added to the GRNN using Hamming distance, equation (8), in the same way as it was added to equation (7). The result is equation (14) where we set  $\alpha$  using equation (15).

$$\hat{Y}(\mathbf{X}) = \frac{\sum_{i=1}^n Y_i \exp \left[ -\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}| - \alpha \right]}{\sum_{i=1}^n \exp \left[ -\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}| - \alpha \right]} \quad (14)$$

$$\alpha = \max_i \left[ -\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}| \right] \quad (15)$$

### 2.2.2. Preparing Data for Generalized Regression Neural Networks

When preparing data for modeling with a GRNN, two important considerations are data standardization and validation. Since we are applying a single spread value for all features, not standardizing will bias results based on the variance and values for individual features. A common standardization technique is to normalize observations for each feature in the data set. This can be done by applying equation (16) to each of the  $n$  observations for an individual feature  $\mathbf{X}$ . This helps balance data sets where the features have greatly different scales. We can standardize the entire data set before choosing a validation set to be fast and efficient, but this may skew the results since data that may be used for testing or validation are also being used to normalize the training data.

$$X_i' = \frac{(X_i - \bar{\mathbf{X}})}{\sqrt{\text{var}(\mathbf{X})}} \quad (16)$$

If there is a concern about including testing and validation data in the calculation of the sample mean and variance for normalization, then the values can be calculated from the training set and applied to observations from the testing and validation set separately.

A decision on the validation method needs to be made when partitioning the data into training and testing sets. There are different validation methods that can be used for constructing and validating ANNs. Two common methods are the hold out method and  $K$ -fold cross validation. They both have their strengths. The hold out method is good when there is a lot of data and it is acceptable to dispose of some data, while the  $K$ -fold cross validation is good for small data sets where all data points are valuable, to include data sets where one category has very few observations.

The hold out method randomly separates the data into sets (Kohavi, 1995). For example, a standard hold out method would contain 60% of the data for training, 20% of the data for testing parameters, and 20% for validating the model. This method is good since the validation data is completely separate from parameter setting, therefore avoiding any bias that may occur.

$K$ -fold cross validation is where the data is separated into  $K$  sets (Kohavi, 1995). Working with  $K$  data sets, one set is removed from the data and used as a test set to identify optimal parameters or to evaluate the model using the rest of the data as the training set. This is then repeated using another data set until all data sets have been used. Then the average or Mean Square Error (MSE) is calculated from these responses and is used in evaluating how well the model performs. This method is effective since it uses all data for training and validation, but can be very computationally expensive.

A specific version of  $K$ -fold cross validation is 'leave-one-out' (Kohavi, 1995), where  $K$  is equal to the number of observations. Here, each of the  $K$  sets has a single observation. This version is commonly used and easy to implement. We use the 'leave-one-out' method for the majority of our research and average the results across all  $K$

models. We do this because we assume this method would normally be used on a training set of a hold out method, since the data would be smaller, and there is more value placed on the retention of each data point. Another reason to use 'leave-one-out' is because we assume that at least one of the categories of interest we are modeling is disproportionately small and each point adds value to the model.

### **2.2.3. Artificial Neural Network Performance Measures**

ANNs, when used for classification, rely on the confusion matrix for a measure of performance. It is used to determine how accurate a model is when compared to actual results. We can derive a number of performance measures from the confusion matrix, such as Apparent Classification Accuracy (ACA), Apparent Error Rate (APER), Precision, Recall, and F-Score.

In a confusion matrix, each row represents the predicted results from the model, while the columns represent the actual results from the system. The sum of the intersection of each predicted and actual results are displayed in their corresponding box in the matrix.

Looking at Figure 9, we can see how a confusion matrix is put together. To explain this better we consider a notional example in Figure 10. This example shows the confusion matrix of a model determining if a response is a member of one of three groups. We can see there were a total of 17 total observations tested. The model predicted eight members of group 1, when in actuality there were six members of group 1. Of the six members of group 1, the model was able to predict only five of them. The other three members predicted as group 1 were actually members of group 3, while the last group 1 member was predicted to be a member of group 2.

		Actual Membership	
		$\pi_1$	$\pi_2$
Predicted Membership	$\pi_1$	$N_{11}$	$N_{12}$
	$\pi_2$	$N_{21}$	$N_{22}$

$n$  = the number of responses

$\pi_n$  = response  $n$  {binary}

$N_{ij} = \pi_i \cap \pi_j, i, j = 1, 2, \dots, n$

**Figure 9: Confusion Matrix**

		Actual Membership		
		Group 1	Group 2	Group 3
Predicted Membership	Group 1	5	0	3
	Group 2	1	5	1
	Group 3	0	2	0

**Figure 10: Confusion Matrix Notional Example**

The Apparent Classification Accuracy (ACA) (Kuncheva, 2004) is a way to quantitatively measure how accurate a model is as a whole. It is calculated from confusion matrices using equation (17). It represents the ratio of correctly predicted responses to the total number of responses. Referring to the example in Figure 10, the ACA would be 10/17 or 0.588. A drawback from using ACAs is if the data is severely imbalanced. For example if 98% of the data is of one category, then the ACA would be 0.98 if the model classified all the data as being from the dominant category. This clearly would not be useful for a study where classification results for a non-dominant category was of interest.

$$ACA = \frac{\sum_{i=1}^n N_{ii}}{\sum_{i=1}^n \sum_{j=1}^n N_{ij}} \quad (17)$$

The Apparent Error Rate (APER) (Kuncheva, 2004) is related to ACA in that it quantitatively measures how inaccurate a model is as a whole. To calculate it you can either use equation (18) or use equation (17) and (19). It represents the ratio of incorrectly predicted responses to the total number of responses. Referring to the example in Figure 10 the APER is 7/17 or 0.412. APER has the same drawback with imbalanced data as ACA, since they are analyzed similarly.

$$APER = \frac{\sum_{i=1}^n \sum_{j=1}^n N_{ij}, i \neq j}{\sum_{i=1}^n \sum_{j=1}^n N_{ij}} \quad (18)$$

$$APER = 1 - ACA \quad (19)$$

Precision (van Rijsbergen, 1979) is a measure of how accurate the model predictions are for a specific classification. It is calculated from confusion matrices (Figure 9), using equation (20) for classification  $i$ . It represents the proportion of correctly classified points for a specific classification over all points predicted as that classification. Referring to the example in Figure 10, the precision for Group 1 is 5/8 or 0.625.

$$\text{precision}(i) = \frac{N_{ii}}{\sum_{j=1}^n N_{ij}} \quad (20)$$



Recall (van Rijsbergen, 1979) is a measure of how well the model predicted a specific classification. It is calculated from confusion matrices (Figure 9), using equation (21) for classification  $i$ . It represents the proportion of correctly classified points for a specific classification relative to all the actual points for that classification. Referring to the example in Figure 10 the recall for Group 1 is  $5/6$  or  $0.833$ .

$$\text{recall}(i) = \frac{N_{ii}}{\sum_{i=1}^n N_{ij}} \quad (21)$$

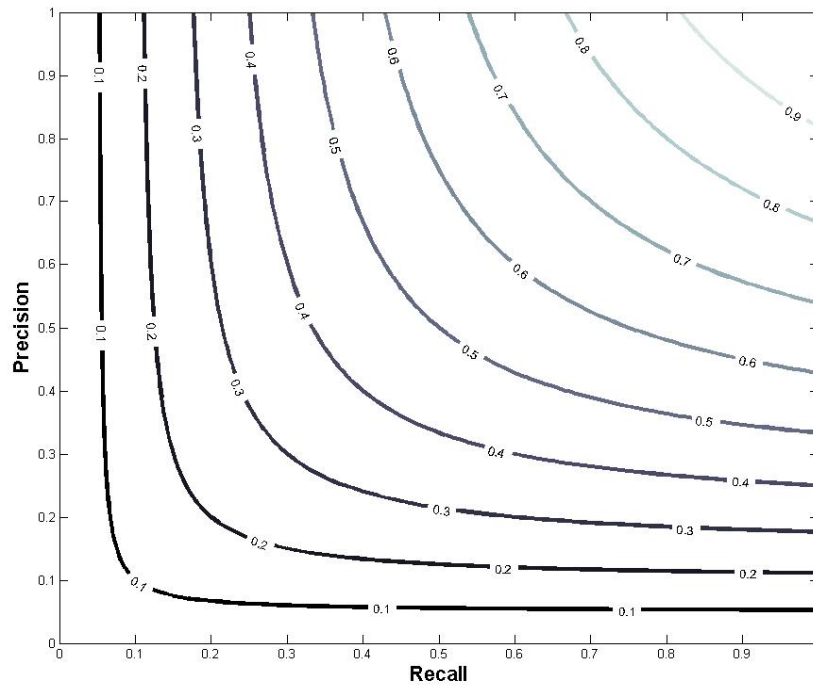
F-Measure (van Rijsbergen, 1979) is a combination of both precision and recall. It is calculated from equations (20) and (21), using equation (23) for classification  $i$ . Referring to the example in Figure 10, the membership F-Measure would be  $0.714$ . A drawback to this measure is if the model does not classify anything for category  $i$ , it results in a division by zero error. Therefore, we have modified equation (22) to assign any undefined value as zero, as in equation (23). Figure 11 is a graph depicting how F-Measure relates to precision and recall. We can see it ranges from zero to one, and depicts a balance between the two performance measures.

$$\text{F-Measure}(i) = \frac{2 \cdot \text{precision}(i) \cdot \text{recall}(i)}{\text{precision}(i) + \text{recall}(i)} \quad (22)$$

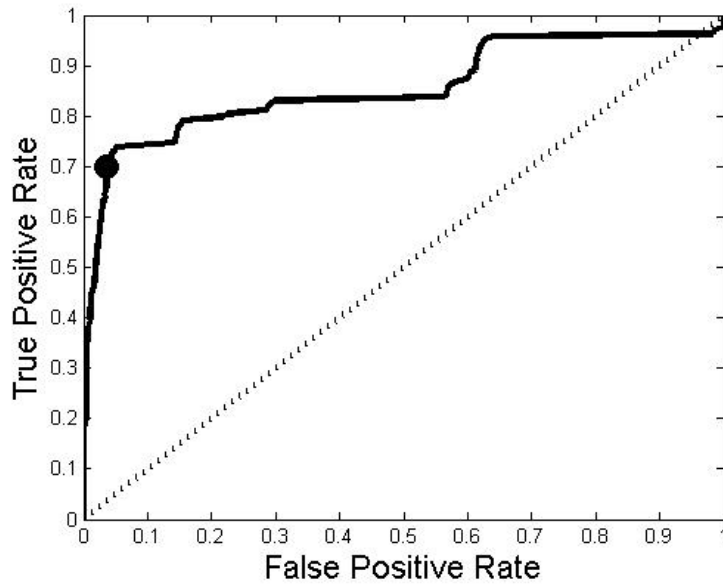
$$\text{F-Measure}(i) = \begin{cases} \frac{2 \cdot \text{precision}(i) \cdot \text{recall}(i)}{\text{precision}(i) + \text{recall}(i)} & N_{ii} > 0 \\ 0 & N_{ii} = 0 \end{cases} \quad (23)$$

A Receiver Operating Characteristic (ROC) curve is a graphical display of the sensitivity to a binary classifier. When looking at output from an ANN, there is usually some cut off used to convert it to binary. For example if an ANN response value was

0.55 and the cut off was set to 0.5, then our reported response would be 1. But if the cut off was set to 0.6, then our response would be set to 0. The model may be very sensitive to the cut off and a ROC curve can identify this. A ROC curve plots the true positive rate versus the false positive rate from a confusion matrix, as we adjust the cutoff point. In the example ROC curve in Figure 12, we can see that if our current cut off rate is the large dot, adjusting the cut off to the right to gain more true positives greatly increases the false positives. If we adjust the cut off to decrease the false positives, then we greatly reduce the true positives. The diagonal dashed line through the center of the chart indicates the 50/50 division where above the line indicates the model predicts better, and below the line is where prediction is better using a  $U(0,1)$  draw with a cut off of 0.5.



**Figure 11: Contour plot of F-Measure related to recall and precision**



**Figure 12: Receiver Operating Characteristic Curve Notional Example**

### 2.3. Summary

This chapter reviews research on player classification in an MMOG and discusses ANNs and their use in classification. The literature review highlights works done by Chen et al. (2009) on bots vs. human players and Ahmad et al. (2009) identifying gold farmers. The work and data from Chen et al. directly lead to the example application of our hybrid GRNN in Chapter IV, while the work from Ahmad et al. and the Virtual World Exploratorium using *EverQuest II* data contributed to the application of our full analysis algorithm in Chapter VI.

The discussion on ANNs highlights the GRNN. Application of the GRNN focuses on the standard Euclidian distance and a modification using the Hamming distance. The section also highlights ‘leave-one-out’ as a preferred validation method and F-Measure as a preferred performance measure for our research.

### **III. Feature Selection for Player Classification**

One of the first challenges when creating a multivariate model for classification is feature selection. It may be possible to collect a large number of features associated with a particular response. There can be two problems with this. One problem is not all of these features are significant to the response. The trick is to identify which features are significant features and which are noise or non-significant features. The second problem is the extra features combined with a large number of observations can make training and running the model extremely time consuming. Thus, a reduction in features can make the model more time efficient. There are methods to reduce the feature set of data, but none are specifically developed for the Generalized Regression Neural Network (GRNN). It is likely that a feature reduction technique may perform better for certain models, and a feature selection technique developed for a specific modeling technique would perform optimally for that modeling technique. Therefore, since there are no GRNN specific feature reduction methods, a feature set obtained by using a standard feature reduction technique may not be optimal for a GRNN model.

This chapter is divided into four sections. The first section reviews some feature selection techniques. The second section covers our new feature selection technique developed for the GRNN. The third section is an example application of our new GRNN feature selection technique. The fourth section is a summary of the chapter.

### 3.1. Review of Select Feature Selection Techniques

There are many different techniques for feature selection. We cover four different methods. They are factor/primary component analysis, stepwise regression, signal to noise ratio in a Feed Forward Neural Network (FFNN), and feature selection using a Radial Basis Function Neural Network (RBFNN).

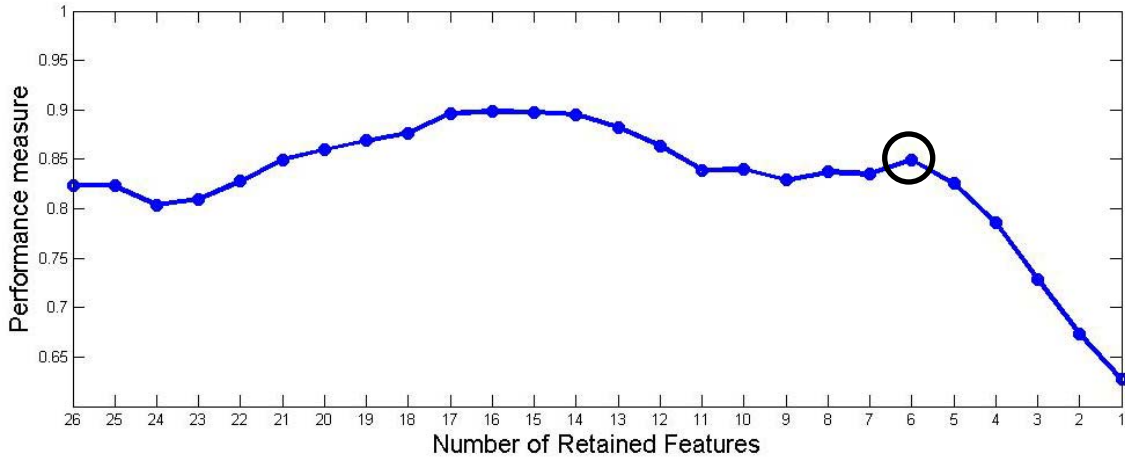
Factor analysis (FA) and Primary Component Analysis (PCA) (Duda, 2001) are two common techniques for feature selection. Both techniques reduce dimensionality of a data set by forming linear combinations of the features. These linear combinations relate to either the correlations among the features or variance of the features in both FA and PCA. These techniques combine original features into a new (reduced) feature set. A problem with these techniques is they do not reduce the number of features based on significance to the response. Therefore, the new reduced feature set from either FA or PCA will be comprised of features that may or may not be significant to the response.

Stepwise regression (stepwisefit, 2010) is another technique to reduce features from a data set. This technique generates a linear model and determines weights relating the significance of each feature. Then, depending on which technique you use, it adds or deletes the most/least significant feature from the model. This technique is effective, but is limited to linear regression models.

FFNNs tend to be very time consuming, but the structure has been shown to work well for feature selection as in Bauer et al. (2000). The weights to the nodes in the hidden middle layer of a two layer FFNN can be used to identify the significance of a feature compared to a noise feature. As in Bauer et al. (2000), a Signal to Noise Ratio (SNR) can be developed using the weights from Figure 5 and equation (24). It evaluates

the ratio of the weight of a feature to the weight of a noise feature generated with random numbers from a uniform distribution. The feature with the smallest SNR is eliminated from the set and SNRs are re-evaluated with the reduced feature set. The Apparent Classification Accuracy (ACA) can be calculated at each step to see how the model worsens with each removed feature. A plot such as Figure 13 can be used to identify a point where the ACA starts to fall off dramatically. The point circled in Figure 13 indicates the cut off between significant and insignificant features. Thus the remaining features are significant to the model. A major drawback to this method is the FFNN has a random element to it. This makes it difficult to generate consistent model weights, thus possibly changing the least significant feature.

$$SNR_i = 10 \log_{10} \left( \frac{\sum_{j=1}^J (w_{i,j}^1)^2}{\sum_{j=1}^J (w_{Noise,j}^1)^2} \right) \quad (24)$$



**Figure 13: Plot of apparent classification accuracies after stepwise feature reduction**

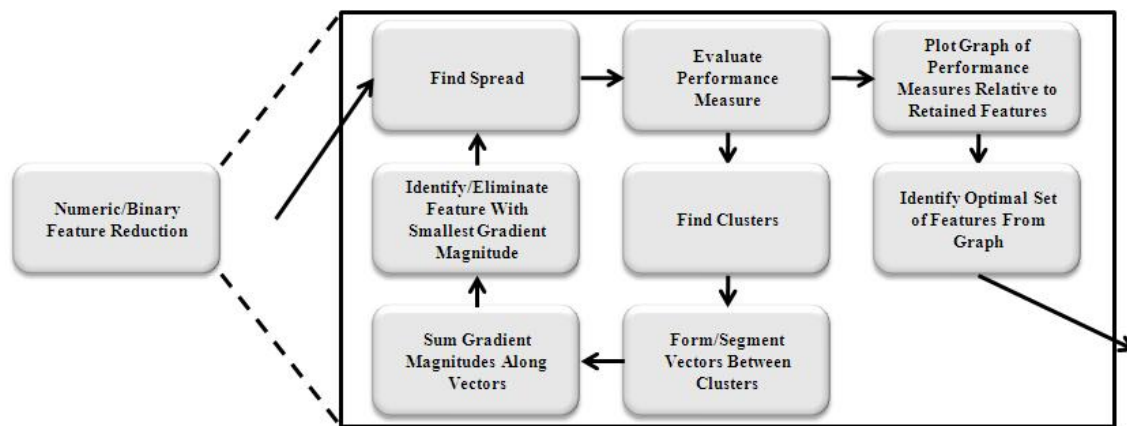
RBFNN is a neural network technique that is faster and less random than the FFNN. Research performed by Flietstra et al. (2003) use a combination of clustering and gradient analysis to reduce the number of features and size of the training set. They use clustering to reduce the number of exemplars and gradient analysis to reduce the number of features. Similar to the FFNN SNR technique, they identify the optimal number of features using a plot of performance measures relative to the remaining number of features, as in Figure 13. A drawback to using RBFNNs is the use of a second layer of hidden weights that need to be trained similar to FFNNs, adding complexity and time.

### **3.2. New Feature Selection Technique**

We propose a new feature selection technique. Inspired by work from Flietstra (2003) and Ruck (1990), our technique is based on using Parzen Windows distributions selected by data type in building a GRNN. It separates the training data into categories by type and then using the selected Parzen Windows distribution, identifies the feature that has the smallest change between the categories. This is done by analyzing the gradient of vectors between the categories, with the intent to identify the changes in the gradient related to when the vector crosses the boundaries between the categories. The feature with the smallest change should be the least effective feature for discriminating between the categories and is removed from the training set. This method is repeated until all features have been removed. Then, the analyst can compare the performance of the training sets at each stage and determine the best training set to use.

Figure 14 is an overview of our new feature selection technique. It can be broken into three main steps. The first step is determining a spread parameter to be used in the second and third steps. The second step is evaluating the performance of the model. This

is done prior to the removal of a feature. This performance measure is retained for comparison with the subsequent reduced model (one less feature). The third step is the removal of a feature. It contains sub-steps of finding clusters, forming vectors between the clusters, calculating the gradients along the formed vectors, and then eliminating the feature(s) with the smallest gradient magnitude. All three steps are repeated until all of the features are removed. These steps are elaborated on in subsection 3.2.1.



**Figure 14: Flow Chart Depicting Generalized Regression Neural Network Feature Reduction for Numeric and Binary Data Types**

### 3.2.1. Technique Overview

The first step in the feature reduction method is to determine an optimal spread. This spread will be used in evaluating the performance of the training set and for determining the gradient of the vector generated between the categories in the feature reduction step. Therefore it is important to find an optimal or near optimal spread. There are different methods for determining the spread in a GRNN, but one of the most common is an exhaustive search. This is where you create models with differing spread



values and select the spread with the best performance. This can be time consuming on large data sets, but for smaller data sets it is sufficiently quick.

The second step is recording a performance measure. This is done so we can evaluate how well the model does with the beginning set of features. We collect the performance measure prior to the removal of a feature in our algorithm since we need a starting performance for a full model. The performance measure is then collected for each reduced model along with the associated remaining features. This is done until all the features have been removed. A graph such as Figure 13 can be generated showing the performance measure for each model to determine the optimal number of features.

The third step is identifying a feature for removal. Since there are no calculated weighting functions like a FFNN, we must look at other methods to classify the effectiveness of each feature. We start by looking for a significant change in the gradient at or between each of the boundaries for classification categories. In examining these gradients, we look for dramatic changes in any of its partial derivatives. We assume that a feature with a significant ability to discriminate between categories will have a dramatic change (large magnitude) in the partial derivatives around the boundaries between classification categories. Therefore, the feature(s) with the smallest change are the least effective feature(s) for discriminating between categories and is removed from the model.

This approach requires searching the data space to find the boundaries between classification categories. We standardize the data so we do not need to worry about features with different ranges and scales. Then, we search for the boundaries. One option is to exhaustively search the data. This can be done by specifying a number of equally separated levels for each feature and then creating a series of test points using all

possible combinations of the divisions for each feature. The problem with this is the sheer number of test points possible. To evenly search a space with  $p$  features and  $m$  divisions of each feature, requires  $m^p$  test points. For example, if we had 3 features and 10 divisions for each feature, we would have  $10^3$  or 1,000 test points. This could easily get out of hand as seen with 17 features and only 3 divisions, resulting in  $3^{17}$  or 129,140,163 test points.

To avoid the problems with the volume of test points, we narrow our search space. Instead of testing all the points, we look in areas where we know there is a boundary between the categories, such as the space directly between training points from differing categories. Instead of connecting each point from one category to each point from another category, we identify multiple clusters of observations within each category. Then, we identify the centroid for each cluster. Now, these centroids are representative samples of each category and vectors linking centroids from differing categories contain a point or points identifying the gradient change between categories. Therefore, all the vectors between centroids from differing categories are identified and points along these vectors are collected and used to compute gradients and partial derivatives for selecting a feature(s) for removal as described previously. This approach provides a more effective use of space and significantly reduces the number of test points relative to an exhaustive search across all features.

To identify clusters we use *X*-means (Pelleg, 2000). This method determines the optimal number and centroids of clusters from a set of data. It is available in an executable form for Windows and with a careful set up of the data, we are able to execute

the routine from MATLABs DOS command function, thus allowing the use of *X*-means in MATLAB scripts.

The training data is separated into corresponding categories as described above. Then *X*-means is applied to each of the categories to identify clusters of similar features within each category. The centroids of each cluster are retained as representations of the clusters. Then, vectors are generated between all the centroids from differing categories. These vectors are then divided up into an equal numbers of representative points. These points are used to obtain gradients and the magnitude of each gradient is summed. It is possible that a pair of centroids from differing categories could be equal. These special pairs can be skipped since the vector between them has no length. After all the test points have been collected and their magnitudes summed, the summed gradients are compared and the feature with the smallest summed magnitude is identified. Since this feature has the least change across the data sets, it is likely not as significant as the other features. However, it is possible to have multiple minimum values such as when there are multiple features with a gradient equal to zero. If the minimum value is not equal to zero, only a single feature is removed and our code uses the first minimum it identifies. If the minimum value is equal to zero, we assume that there is no significant change and remove all of the features resulting in a zero magnitude gradient. There may be better methods when dealing with multiple minimums, but that is for future research.

Now, a new training set is formed using all the features, less the one feature (or multiple zero gradient features) that was deemed least significant. The remaining features and performance measures are stored for reference. We then go back and calculate a new spread and repeat the entire process with the remaining features. This is

done until all but one feature is removed. After we have removed all but one feature, we then plot the performance measures with respect to the number of features left as seen in Figure 13. This plot aids in identifying which features we should use for our model. The cut off is dependent on the user. We could use the maximum performance measure value, a severe drop in performance, or even have a minimum performance measure threshold. Looking at the example in Figure 13, the analyst would most likely chose six features, since ACA drops off sharply after this point. After the cut off is identified, a model can be generated based on the features that have been identified as significant.

### 3.2.2. Gradient of Generalized Regression Neural Networks

For this research, the gradient of the Gaussian distributions developed for the GRNN must be calculated at several points. With the interest in dividing the data up by binary and numeric data points, we need to find the gradient of Gaussian distributions with both Euclidian and Hamming distances. Note that the Hamming distance is the same as the city block distance when applied to single binary digits. We show the development of the gradients for both cases in the following discussion. Since both GRNN functions use the same basic Gaussian distribution formula from equation (7), we can separate the basic formula  $Y(\mathbf{X})$  into a numerator  $g(\mathbf{X})$  and a denominator  $h(\mathbf{X})$ . Where,  $\mathbf{X}$  is the 1 by  $p$  matrix associated with a test point and  $p$  is the number of features in the data set.

$$\hat{Y}(\mathbf{X}) = \frac{g(\mathbf{X})}{h(\mathbf{X})} \quad (25)$$

The partial derivatives for the numerator for the Euclidian distance can be seen below.

$$g(\mathbf{X}) = \sum_{i=1}^n Y_i \exp \left[ -\frac{(\mathbf{X} - \mathbf{Z}_i)^T (\mathbf{X} - \mathbf{Z}_i)}{2\sigma^2} \right] \quad (26)$$

$$\frac{\partial g(\mathbf{X})}{\partial X_j} = \frac{1}{-\sigma^2} \sum_{i=1}^n Y_i \exp \left[ -\frac{(\mathbf{X} - \mathbf{Z}_i)^T (\mathbf{X} - \mathbf{Z}_i)}{2\sigma^2} \right] (X_j - Z_{ij}) \quad (27)$$

Where  $n$  is the number of observations,  $p$  is the number of features,  $\mathbf{X}$  is the test point,  $\mathbf{Z}$  is the training set of size  $n$  by  $p$ ,  $\mathbf{Z}_i$  is the  $i^{\text{th}}$  row vector from  $\mathbf{Z}$ , and  $Z_{ij}$  is element from row  $i$  and column  $j$  from  $\mathbf{Z}$ ,  $\mathbf{Y}$  is the training responses relative to  $\mathbf{Z}$ ,  $Y_i$  is the  $i^{\text{th}}$  element from  $\mathbf{Y}$ , and  $\sigma$  is the spread factor. The partial derivatives for the denominator for the Euclidian distance can be seen below.

$$h(\mathbf{X}) = \sum_{i=1}^n \exp \left[ -\frac{(\mathbf{X} - \mathbf{Z}_i)^T (\mathbf{X} - \mathbf{Z}_i)}{2\sigma^2} \right] \quad (28)$$

$$\frac{\partial h(\mathbf{X})}{\partial X_j} = \frac{1}{-\sigma^2} \sum_{i=1}^n \exp \left[ -\frac{(\mathbf{X} - \mathbf{Z}_i)^T (\mathbf{X} - \mathbf{Z}_i)}{2\sigma^2} \right] (X_j - Z_{ij}) \quad (29)$$

Now we can use the quotient rule to combine the partial derivatives for the numerator and the denominator as shown in equation (30). We use equation (30) with our calculated partial derivatives when evaluating the gradient.

$$\frac{\partial \hat{Y}(\mathbf{X})}{\partial X_j} = \frac{\frac{\partial g(\mathbf{X})}{\partial X_j} h(\mathbf{X}) - g(\mathbf{X}) \frac{\partial h(\mathbf{X})}{\partial X_j}}{[h(\mathbf{X})]^2} \quad (30)$$

The same method can be used for the city block distance (Hamming distance for binary data), except using the derivatives for the numerator and denominator below to fill into equation (30). The partial derivative for the numerator for the Hamming distance, equation (8), can be seen below.

$$g(\mathbf{X}) = \sum_{i=1}^n Y_i \exp \left[ -\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}| \right] \quad (31)$$

$$\frac{\partial g(\mathbf{X})}{\partial X_j} = \begin{cases} \left( -\frac{1}{\lambda} \right) \sum_{i=1}^n Y_i \exp \left[ -\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}| \right] (1) & \forall X_j > Z_{ij} \\ \left( -\frac{1}{\lambda} \right) \sum_{i=1}^n Y_i \exp \left[ -\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}| \right] (-1) & \forall X_j < Z_{ij} \\ \text{Does Not Exist} & \forall X_j = Z_{ij} \end{cases} \quad (32)$$

Where  $n$  is the number of observations,  $p$  is the number of features,  $\mathbf{X}$  is the test point,  $\mathbf{Z}$  is the training set of size  $n$  by  $p$ ,  $\mathbf{Z}_i$  is the  $i^{\text{th}}$  row vector from  $\mathbf{Z}$ , and  $Z_{ij}$  is element from row  $i$  and column  $j$  from  $\mathbf{Z}$ ,  $\mathbf{Y}$  is the training responses relative to  $\mathbf{Z}$ ,  $Y_i$  is the  $i^{\text{th}}$  element from  $\mathbf{Y}$ , and  $\lambda$  is the spread factor. The partial derivatives for the denominator for the Hamming distance can be seen below.

$$h(\mathbf{X}) = \sum_{i=1}^n \exp \left[ -\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}| \right] \quad (33)$$

$$\frac{\partial h(\mathbf{X})}{\partial X_j} = \begin{cases} \left( -\frac{1}{\lambda} \right) \sum_{i=1}^n \exp \left[ -\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}| \right] (1) & X_j > Z_{ij} \\ \left( -\frac{1}{\lambda} \right) \sum_{i=1}^n \exp \left[ -\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}| \right] (-1) & X_j < Z_{ij} \\ \text{Does Not Exist} & X_j = Z_{ij} \end{cases} \quad (34)$$

Since we are working with absolute values, we need to note that first the partial derivatives are step functions and second that the derivatives of both the numerator and the denominator do not exist when  $X_i = Z_i$ . For our algorithm, we have inserted a check

to make sure the test points do not equal one or zero and if they do, the test point is adjusted by a small factor to avoid a value that does not exist.

Again, since our algorithm is executed in MATLAB, we include a shift factor to adjust the exponentials to avoid the value being rounded to zero and thus having zero in the denominator, similar to the shift factor seen in section 2.2.1. First we multiply equation (30) by a factor equal to one for both Hamming and Euclidian distances

$$\frac{\partial \hat{Y}(\mathbf{X})}{\partial X_j} = \frac{[\exp(\alpha)]^2 \frac{\partial g(\mathbf{X})}{\partial X_j} h(\mathbf{X}) - g(\mathbf{X}) \frac{\partial h(\mathbf{X})}{\partial X_j}}{[\exp(\alpha)]^2 [h(\mathbf{X})]^2} \quad (35)$$

then distribute  $\exp(\alpha)$  to each sub function

$$\frac{\partial \hat{Y}(\mathbf{X})}{\partial X_j} = \frac{\frac{\partial g(\mathbf{X})}{\partial X_j} h(\mathbf{X}) [\exp(\alpha)]^2 - g(\mathbf{X}) \frac{\partial h(\mathbf{X})}{\partial X_j} [\exp(\alpha)]^2}{[h(\mathbf{X}) \exp(\alpha)]^2} \quad (36)$$

to obtain equations (37), (38), (39), and (40) for Euclidian distances and equations (42), (43), (44), and (45) for Hamming distances. Where  $\alpha$  is defined by equation (41) for Euclidian distance and (46) for Hamming distances.

$$g(\mathbf{X}) \exp(\alpha) = \sum_{i=1}^n Y_i \exp \left[ -\frac{(\mathbf{X} - \mathbf{Z}_i)^T (\mathbf{X} - \mathbf{Z}_i)}{2\sigma^2} - \alpha \right] \quad (37)$$

$$\frac{\partial g(\mathbf{X})}{\partial X_j} \exp(\alpha) = \frac{1}{-\sigma^2} \sum_{i=1}^n Y_i \exp \left[ -\frac{(\mathbf{X} - \mathbf{Z}_i)^T (\mathbf{X} - \mathbf{Z}_i)}{2\sigma^2} - \alpha \right] (X_j - Z_{ij}) \quad (38)$$

$$h(\mathbf{X}) \exp(\alpha) = \sum_{i=1}^n \exp \left[ -\frac{(\mathbf{X} - \mathbf{Z}_i)^T (\mathbf{X} - \mathbf{Z}_i)}{2\sigma^2} - \alpha \right] \quad (39)$$

$$\frac{\partial h(\mathbf{X})}{\partial X_j} \exp(\alpha) = \frac{1}{-\sigma^2} \sum_{i=1}^n \exp \left[ -\frac{(\mathbf{X} - \mathbf{Z}_i)^T (\mathbf{X} - \mathbf{Z}_i)}{2\sigma^2} - \alpha \right] (X_j - Z_{ij}) \quad (40)$$

$$\alpha = \max \left( -\frac{(\mathbf{X} - \mathbf{Z}_i)^T (\mathbf{X} - \mathbf{Z}_i)}{2\sigma^2} \right) \quad (41)$$

Similarly

$$g(\mathbf{X}) \exp(\alpha) = \sum_{i=1}^n Y_i \exp \left[ -\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}| - \alpha \right] \quad (42)$$

$$\frac{\partial g(\mathbf{X})}{\partial X_j} \exp(\alpha) = \begin{cases} \left( -\frac{1}{\lambda} \right) \sum_{i=1}^n Y_i \exp \left[ -\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}| - \alpha \right] (1) & X_j > Z_{ij} \\ \left( -\frac{1}{\lambda} \right) \sum_{i=1}^n Y_i \exp \left[ -\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}| - \alpha \right] (-1) & X_j < Z_{ij} \\ \text{Does Not Exist} & X_j = Z_{ij} \end{cases} \quad (43)$$

$$h(\mathbf{X}) \exp(\alpha) = \sum_{i=1}^n \exp \left[ -\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}| - \alpha \right] \quad (44)$$

$$\frac{\partial h(\mathbf{X})}{\partial X_j} \exp(\alpha) = \begin{cases} \left( -\frac{1}{\lambda} \right) \sum_{i=1}^n \exp \left[ -\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}| - \alpha \right] (1) & X_j > Z_{ij} \\ \left( -\frac{1}{\lambda} \right) \sum_{i=1}^n \exp \left[ -\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}| - \alpha \right] (-1) & X_j < Z_{ij} \\ \text{Does Not Exist} & X_j = Z_{ij} \end{cases} \quad (45)$$

$$\alpha = \max \left( -\frac{(\mathbf{X} - \mathbf{Z}_i)^T (\mathbf{X} - \mathbf{Z}_i)}{2\sigma^2} \right) \quad (46)$$



Now that we have the set of partial derivatives for each test point, we can combine their absolute values to form a modified gradient for each test point that represents the magnitudes of the partial derivatives, see equation (47). This will indicate the change in the distribution at point  $\mathbf{X}$ .

$$\Delta\hat{Y}(\mathbf{X}) = \left( \left| \frac{\partial\hat{Y}(\mathbf{X})}{\partial X_1} \right|, \left| \frac{\partial\hat{Y}(\mathbf{X})}{\partial X_2} \right|, \dots, \left| \frac{\partial\hat{Y}(\mathbf{X})}{\partial X_p} \right| \right) \quad (47)$$

We will sum the modified gradients for all of the test points to identify the feature with the greatest sum of magnitudes. The summed gradients are used to identify the boundaries between the different categories and evaluate the value of each factor.

### 3.3. University of Wisconsin Breast Cancer Data Example

We apply our feature selection method to breast cancer data obtained from University of California Irvine Machine Learning Repository (Breast Cancer Wisconsin (Original) Data Set, 1992). This data set was chosen since it was used by (Fleitstra, 2003) to test their feature/architecture selection technique for the RBFNN.

#### 3.3.1. Data description

The data was initially collected from the University of Wisconsin Hospitals, Madison by Dr. William H. Wolberg for the diagnosis of breast cytology (Wolberg, 1990). It contains nine features and one response. The nine features are Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell size, Bare Nuclei, Bland Chromatin, Normal Nucleoli, and Mitoses. The response is a binary result of benign (0) or malignant (1). The data set contains 699 observations with 16 having missing data.

Initially, we take the 699 observations and eliminate the 16 observations with missing data. Then, similar to the Fleitstra article (2003), we add five noise features. They are columns of random variates from a uniform distribution over the unit interval. Two more features are repeated features with a noise element added. The noise element is a random draw from the Normal distribution with mean of zero and standard deviation of 0.04. The two repeated features are features that were identified by Fleitstra et al. (2003) to be significant (Bare Nuclei) and relatively insignificant (Mitosis). The modified data set has 16 features and 683 observations. The observations can be broken into 444 benign (0) and 239 Malignant (1). The data set is reviewed in Table 3 with associated feature number.

**Table 3: Feature Overview of Breast Cytology Data**

Cancer	#features	#observations	ratio of 0/1	
		16	683	444/239
column #	features	Data types	min	max
1	Clump Thickness	Integer	1	10
2	Uniformity of Cell Size	Integer	1	10
3	Uniformity of Cell Shape	Integer	1	10
4	Marginal Adhesion	Integer	1	10
5	Single Epithelial Cell Size	Integer	1	10
6	Bare Nuclei	Integer	1	10
7	Bland Chromatin	Integer	1	10
8	Normal Nucleoli	Integer	1	10
9	Mitoses	Integer	1	10
10	Noise U(0,1)	Numeric	0.0005	0.9991
11	Noise U(0,1)	Numeric	0.0078	0.9995
12	Noise U(0,1)	Numeric	0.0003	0.9995
13	Noise U(0,1)	Numeric	0.0007	0.9954
14	Noise U(0,1)	Numeric	0.0010	0.9998
15	Bare Nuclei +N(0,0.04)	Numeric	0.9119	10.0809
16	Mitoses + N(0,0.04)	Numeric	0.9005	10.0923
17	benign (2) or malignant (4)	Binary (converted to 0,1)	2	4

### 3.3.2. Analysis

For our analysis we employed 'leave-one-out' for our validation method. We decided on using this since we were setting our algorithm up to be employed on large imbalanced data sets. We also focused on ACA for our performance measure, since the data is relatively balanced when compared to the larger massive multiplayer online games (MMOG) data sets we use and we wish to compare our results with Fleitstra et al. (2003).

We analyzed the data using five different numbers of sections for the test vectors between cluster centroids. This was done to see if the number of divisions of the vectors between clusters affects the feature reduction. We divided the vectors into 500, 100, 10, and 2 sections. The resulting order of features reduced were the same for all the set ups. The order of feature reduction can be seen in Table 4.

Looking at Table 4 we note that the first two features removed were noise features. We also note that early on (within the first nine), all the pure noise features have been removed. We would expect all the noise features to be removed first, but it is possible that some of the features collected could negatively affect the classification or that the noise features were better than some collected features.

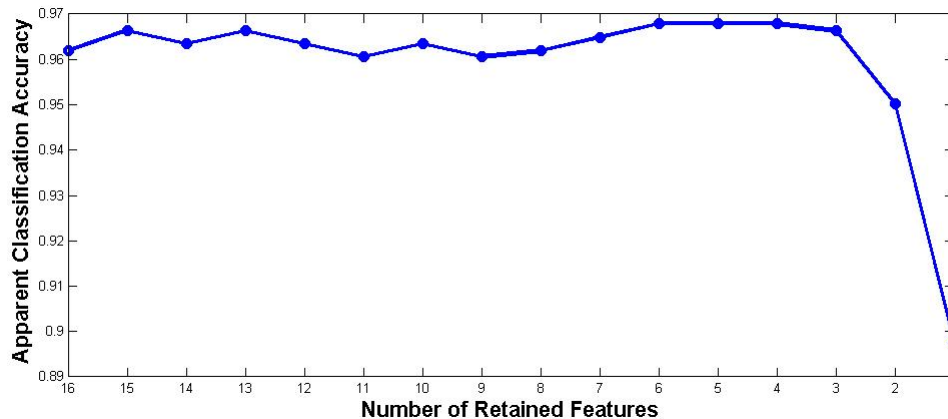
We assumed that feature 16 would be removed before feature 9, along with feature 15 removed before feature 6. Noting that our assumption was correct for 6 and 15, but not for 9 and 16, we examined the amount of added noise. The original response values for feature 9 ranged from 1-10, thus the amount of noise added from a  $N(0,0.04)$  would likely be so small as to not significantly affect the results. This explains why feature 16 was not removed before feature 9.

**Table 4: Summary of Breast Cytology Feature Selection**

<b>Cycle</b>	<b>Removed Feature</b>	<b>ACA</b>	<b>Modeled Features</b>
0	-	0.9619	1-16
1	11(noise)	0.9663	1-10,12-16
2	13(noise)	0.9634	1-10,12,14-16
3	9	0.9663	1-8,10,12,14-16
4	8	0.9634	1-7,10,12,14-16
5	10(noise)	0.9605	1-7,12,14-16
6	14(noise)	0.9634	1-7,12,15,16
7	5	0.9605	1-4,6,7,12,15,16
8	15(6+noise)	0.9619	1-4,6,7,12,16
9	12(noise)	0.9649	1-4,6,7,16
10	7	0.9678	1-4,6,16
11	4	0.9678	1-3,6,16
12	3	0.9678	1,2,6,16
13	2	0.9663	1,6,16
14	1	0.9502	6,16
15	6	0.8975	16
16	16(9+noise)	-	-

Looking at Table 5 and Figure 15, we note that the first 13 features removed have little effect on ACA. This would be the logical place to cut off our feature reduction. Therefore our final model contains features 16, 6, 2, and 1. These features effectively represent the diagnosis of breast cytology.

Since we used the leave-one-out method, we can just use the ACA from the test to describe the accuracy. From Figure 15 we see an ACA of 0.9619 with all the features while with only 4 features we obtain an ACA of 0.9678, increasing the ACA while greatly reducing the number of features used.



**Figure 15: Plot of apparent classification accuracies versus number of features removed for University of Wisconsin Breast Cancer Data**

This example demonstrates that our new method works well for feature selection. We can see that all the pure noise features are removed early on, along with the redundant features. We are left with 4 features, resulting in an ACA better than using all 16 features. We did retain a feature that was both deemed to be insignificant and has noise added to it, but looking at Table 3 we can see the noise is small and most likely didn't really affect it. Since we use the data in a similar approach to that of Fleitstra et al. (2003), we can compare our findings. Before we compare, we need to note that our data is not identical, since we generated our own noise. We also do not know what validation method they used and both of these factors impact results and our comparison. Therefore, we will display the differing results, but cannot definitively conclude one method better than the other. In trying to keep with their analysis, we use the same distributions for noise and also use the number of features with the minimum APER (maximum ACA) as our retained features.

Fleitstra et al. (2003) retains 10 features with an APER of 0.0458, while we retain 4 features with an APER of 0.0322. We retain fewer features with a lower APER. Unfortunately, we do not know all of the 10 features Fleitstra retains, but we do know that they retain both redundant features, so we at least have two features in common. We attempted to keep the data as much the same as Fleitstra, but there are differences since we were required to generate our own random features and validation sets. These differences could drive some of the performance, but not large enough to say that the method did not perform well. Therefore, even though our feature selection technique allowed us to model the data with a better APER with fewer features, we can't be certain that the greater performance is from the technique or the data differences and conclude that our method is equivalent to Fleitstra.

### **3.4. Conclusion**

This chapter presents our feature selection technique that begins by using Parzen Windows distributions selected by data type in building a GRNN. Features are compared by examining the magnitude of the partial derivatives at selected points along gradients between points or clusters from different classification categories. Our discussion details the basic algorithm and equations, to include a modification to eliminate problems from rounding errors. It concludes with an example analysis using the University of Wisconsin breast cancer data from University of California Irvine Machine Learning Repository. Noise features were added to the data, and our feature reduction technique was effective in reducing the data from 16 features to just 4, while increasing the ACA from 0.9619 to 0.9678.

While coding up this technique attention needs to be given to validation and how the calculations are performed. This algorithm can be costly in both time and memory when applied to large data sets. Our initial code was developed for a small data set, with only 16 features and about 700 observations. It took only a few minutes to complete. When we then changed the data to a much larger dataset, with 27 features and over 2,000,000 observations, we quickly ran out of memory. With a few memory conserving tricks such as only working on parts of the data and keeping data on the hard drive instead of in memory, we were barely able to get the technique to function. Therefore, we reduced the number of observations to over 21,000 observations to make the data set more manageable. We calculated the time it would take to complete our feature reduction technique for the reduced set and determined it would take about three months. We then had to look at how we were performing our calculations and leveraging some of MATLABs strengths in matrix multiplication. After all the extra work, we now have an algorithm that completes in about three days. With some of this in mind, it is important to think about how these calculations will be applied to the intended data set along with problems that may arise from the data set.

## **IV. Hybrid Generalized Regression Neural Network for Classification**

There are many data sets that are not one type or another, but a mix of data types. Our hybrid Generalized Regression Neural Network (GRNN) method applies the concept that a mixed data set may be modeled better using data specific GRNN kernels. Specifically we separate a mixed data set into numeric and binary data types. Categorical data is handled by transforming each category into a binary variable. Then each data type is processed using the corresponding kernel. The final results are then combined using a convex combination to retain the 0-1 output format.

This chapter is divided into four sections. The first section discusses the background to our hybrid GRNN. The second section covers the methodology to our hybrid GRNN. The third section is an example application of the new hybrid GRNN method for player classification using a large Massive Multiplayer Online Game (MMOG) database. The fourth section is a summary of the chapter.

### **4.1. Background**

In a previous article about Probabilistic Neural Networks (PNNs) (Specht, 1990), Specht noted that we could change the kernel of the Parzen windows and that certain kernels may be better for certain data sets. In particular, he noted that Euclidian distance was good for numeric data and Hamming distance was good for binary data. Since PNNs and GRNNs are similar in the use of Parzen windows, we can use selected kernels best suited to a data type to generate models. It can be shown that City Block distance is equal to Hamming distance when using binary data, so equation (48) is a GRNN developed for binary data, while equation (49) is a GRNN developed for numeric data.



For these equations,  $n$  is the number of observations,  $p$  is the number of features,  $\mathbf{X}$  is the test vector,  $\mathbf{Z}$  is the training set of size  $n \times p$ ,  $Y$  is the training responses relative to  $\mathbf{Z}$ , and  $\sigma$  or  $\lambda$  is the spread of the function. Equation (48) is for the standard or Euclidian distance based GRNN, while equation (49) is for the binary or Hamming distance based GRNN.

$$\hat{Y}_E(\mathbf{X}) = \frac{\sum_{i=1}^n Y_i \exp\left[-\frac{(\mathbf{X}-\mathbf{Z}_i)^T(\mathbf{X}-\mathbf{Z}_i)}{2\sigma^2}\right]}{\sum_{i=1}^n \exp\left[-\frac{(\mathbf{X}-\mathbf{Z}_i)^T(\mathbf{X}-\mathbf{Z}_i)}{2\sigma^2}\right]} \quad (48)$$

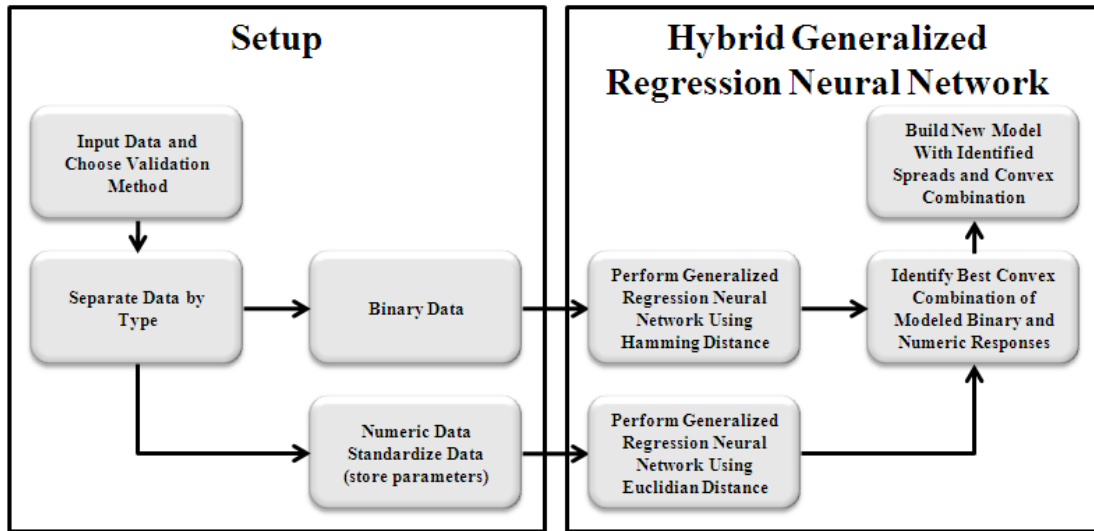
$$\hat{Y}_H(\mathbf{X}) = \frac{\sum_{i=1}^n Y_i \exp\left[-\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}|\right]}{\sum_{i=1}^n \exp\left[-\frac{1}{\lambda} \sum_{j=1}^p |X_j - Z_{ij}|\right]} \quad (49)$$

## 4.2. Methodology

The flow of the hybrid GRNN is outlined in Figure 16. First the data is separated by type, binary and numeric. Then the data is standardized so we can use one spread for each data type. The standardizing only applies to the numeric data since binary data all have the same scale. Then, using both equations (48) and (49) we build GRNN models for each data type. The building of these models includes optimizing spreads for each model. Then the raw probabilities output from the retained models are combined using a convex combination, as in equation (50). The convex combination is optimized by cycling through several parameters for  $\beta$  to identify the best outcome for a pre-specified performance measure. Then when the best combination is identified it is used to create

the final combined raw probabilities that are used to identify the final output of the model.

$$\hat{Y}(\mathbf{X}) = \beta \hat{Y}_H(\mathbf{X}) + (1 - \beta) \hat{Y}_E(\mathbf{X}), \beta = [0, 1] \quad (50)$$



**Figure 16: Flow Chart of the Hybrid Generalized Regression Neural Network**

An added benefit of using this method is that it lends itself to parallel processing. With large mixed datasets, performing the calculations can become severely time-consuming. With this method, since we are developing independent GRNNs for different data types, they can be processed on separate computer nodes, thus working in parallel with each interim GRNN calculated with less data. Both of these aspects can greatly reduce total processing time.

### 4.3. Bot Traffic Example

To test the Hybrid GRNN, we chose data from research performed by Chen et al. (2009) on identifying bots. They investigated automatic, game independent, bot

identification through network traffic analysis. An MMOG developer is very interested in identifying bots. Bots can have a very negative effect on the community within the MMOG and are generally forbidden by game developers. They give unfair advantages to players through reduction in time invested by players and they unbalance game economies.

The basic idea behind the research from Chen et al. (2009) was to see if the Transmission Control Protocol (TCP) traces from a bot were statistically different from the TCP traces from a human player. A TCP trace is the collection of all the headers from the information packets sent between two computers. A TCP trace has very little information as seen in Figure 17. The basic TCP trace information contains a time stamp, sender IP address, IP address, packet flag, number of bytes sent, and error checking information. The authors used this data to calculate response times, sending patterns, volume of information, and sensitivity to network conditions.

```
12:18:10.790326 IP 192.168.0.180.1153 > 61.220.62.131.5000: . ack 60 win 64109
12:18:10.824979 IP 61.220.62.131.5000 > 192.168.0.180.1153: P 60:69(9) ack 1 win 17359
12:18:10.992232 IP 192.168.0.180.1153 > 61.220.62.131.5000: . ack 69 win 64100
12:18:11.026253 IP 61.220.62.131.5000 > 192.168.0.180.1153: P 69:129(60) ack 1 win 17359
12:18:11.192439 IP 192.168.0.180.1153 > 61.220.62.131.5000: . ack 129 win 65535
```

**Figure 17: Lines from a Transmission Control Protocol Trace**

Using the commercial MMOG game *Ragnarok Online*, the authors were able to take multi-hour TCP traces. Each trace was collected from the network connected to the game client. The data collected incorporated different network conditions and different bots along with different players possessing a range of MMOG experience levels. A summary of the TCP traces are shown in Table 5.

**Table 5: Transmission Control Protocol Traces Summary (Chen, 2009)**

Category	Player	ID	Network*	Period	# Conn	Pkt	Bytes	Pkt Rate†	Avg RTT	Loss	
Human Player	Gino	<i>A1</i>	HiNet	1.8 hr	12	51,823	3.5 MB	0.9 / 3.9 pkt/s	82.0 ms	0.03%	
		<i>A2</i>	2M/512Kbps	5.6 hr	14	147,814	10.5 MB	0.8 / 3.4 pkt/s	95.4 ms	0.03%	
	Kiya	<i>B1</i>	APOL 2M/512Kbps		0.4 hr	45	15,228	1.0 MB	1.2 / 4.5 pkt/s	81.6 ms	0.01%
		<i>B2</i>			2.3 hr	108	59,247	3.8 MB	1.1 / 3.3 pkt/s	108.8 ms	0.12%
		<i>B3</i>			2.1 hr	189	47,721	3.2 MB	0.9 / 2.8 pkt/s	125.5 ms	0.23%
		<i>B4</i>			5.0 hr	326	129,177	8.4 MB	1.1 / 3.3 pkt/s	109.8 ms	0.09%
Kuan-Ta	<i>C1</i>	ASNET†	0.8 hr	2	9,681	0.6 MB	0.8 / 1.4 pkt/s	191.8 ms	1.73%		
Jihh-Wei	<i>D1</i>	TANET	2.4 hr	28	48,617	3.2 MB	0.8 / 2.6 pkt/s	45.1 ms	0.01%		
Bot	Kore	<i>K1</i>	TANET	13.4 hr	104	245,709	13.6 MB	0.7 / 2.3 pkt/s	33.0 ms	0.01%	
		<i>K2</i>		26.5 hr	306	479,374	30.4 MB	1.0 / 2.1 pkt/s	45.6 ms	0.04%	
		<i>K3</i>		32.7 hr	37	271,416	13.3 MB	0.6 / 0.7 pkt/s	96.5 ms	0.004%	
		<i>K4</i>		ETWEBS-TW	13.0 hr	38	225,528	11.5 MB	0.9 / 2.0 pkt/s	65.7 ms	0.01%
		<i>K5</i>			5.7 hr	31	110,883	6.0 MB	1.1 / 2.1 pkt/s	90.6 ms	0.20%
	DreamRO	<i>R1</i>	TANET	3.0 hr	7	46,381	2.6 MB	0.9 / 1.7 pkt/s	83.4 ms	0.03%	
		<i>R2</i>		4.8 hr	21	77,675	4.4 MB	0.9 / 1.9 pkt/s	65.2 ms	0.02%	
		<i>R3</i>		42.3 hr	42	652,877	34.1 MB	0.8 / 1.8 pkt/s	85.3 ms	0.05%	
		<i>R4</i>		ETWEBS-TW	11.2 hr	77	320,686	25.1 MB	1.7 / 3.5 pkt/s	85.2 ms	0.05%
		<i>R5</i>			23.1 hr	176	672,325	53.3 MB	1.7 / 3.6 pkt/s	79.4 ms	0.16%
<i>R6</i>	10.5 hr	36	209,347	13.1 MB	1.0 / 2.4 pkt/s	87.7 ms	0.05%				
Total	2 B / 4 P	19		206.6 hr	1,599	3,821,509	241.6 MB				

\* This column lists network names looked up using WHOIS service.

† Access link bandwidth: ASNET (2M/512Kbps), ETWEBS-TW (2M/256Kbps), and TANET (100Mbps)

‡ Packet rate column format is "client data packet rate / server data packet rate," i.e., pure TCP ack packets do not count.

After reviewing the article, we saw an opportunity to confirm their results while developing an improved method. We also noted the use of long, multi-hour, traces for their analysis and hypothesized that through the use of Artificial Neural Networks (ANNs) we could obtain similar results with less data. The article contains a web address to their freely accessible research data, and since we were also interested in obtaining actual data from an MMOG, we decided to use this data for our research. Unfortunately, upon familiarizing ourselves with the data, we realized the data was incomplete. While corresponding with Chen, we were able to obtain some of the incomplete data, but parts of the original data have been permanently lost.

Concerned with the amount of time and data required to identify bots, we decided to explore using traces observed over a shorter time with other analytical methods. We also wanted to look at the effectiveness of our hybrid GRNN versus other ANNs such as a Feed Forward Neural Network (FFNN), a standard GRNN, and a Probabilistic Neural Network (PNN). Using most of the raw data from Chen et al. (2009) we began by

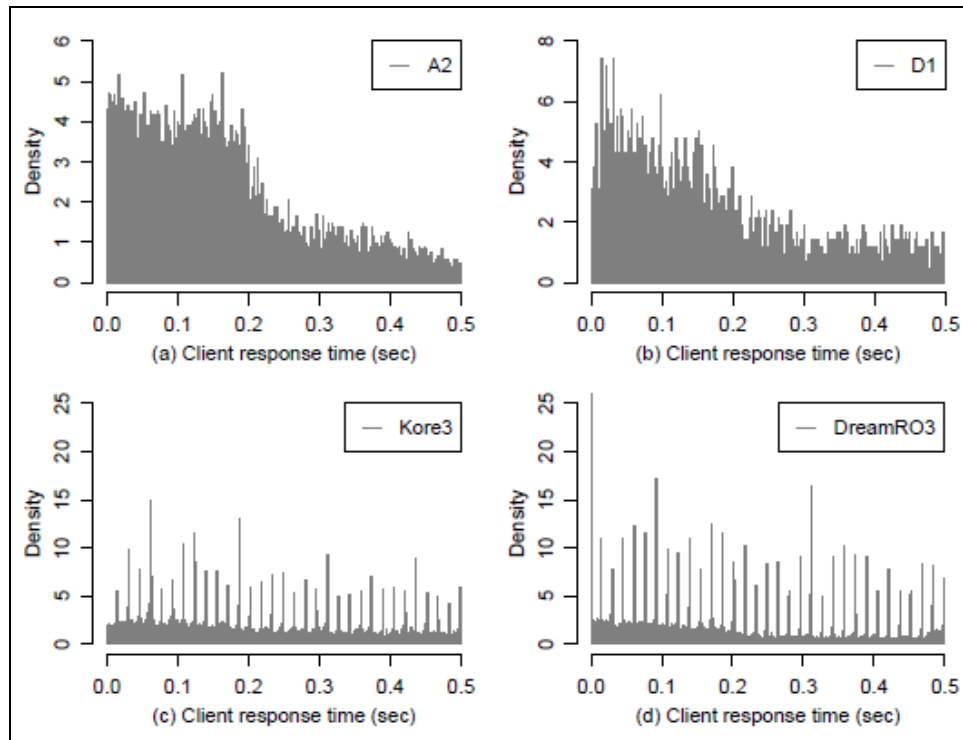
breaking each trace into five minute chunks and collected statistics for each five minute period. We felt that five minutes was a short amount of time in comparison to an hour or more and should contain enough packets for our approach.

#### **4.3.1. Original TCP Analysis**

Chen et al. (2009) propose four different methods to determine if the human and bot playing patterns are statistically different. The first method is to examine the timing of client commands relative to the arrival time of the most recent server data packet. Figure 18 shows representative histograms of the interarrival times. Figure 18a and Figure 18b are from players, while Figure 18c and Figure 18d are from bots. It is noted that players show randomness in these interarrival times, while bot interarrival times indicate a triggering mechanism since there are evenly spaced peaks in the bots histogram. Due to this triggering mechanism, they suggest simultaneous testing of multimodality and regularity. Multimodality tests are looking for multiple peaks in the histogram of client response times. Regularity tests are looking for response times in multiples of a certain value. As shown in Figure 19a, when classifying a bot using this technique there is a maximum 95% correct rate. With an increase in client packets, there is an increased false positive rate and a decreased false negative rate.

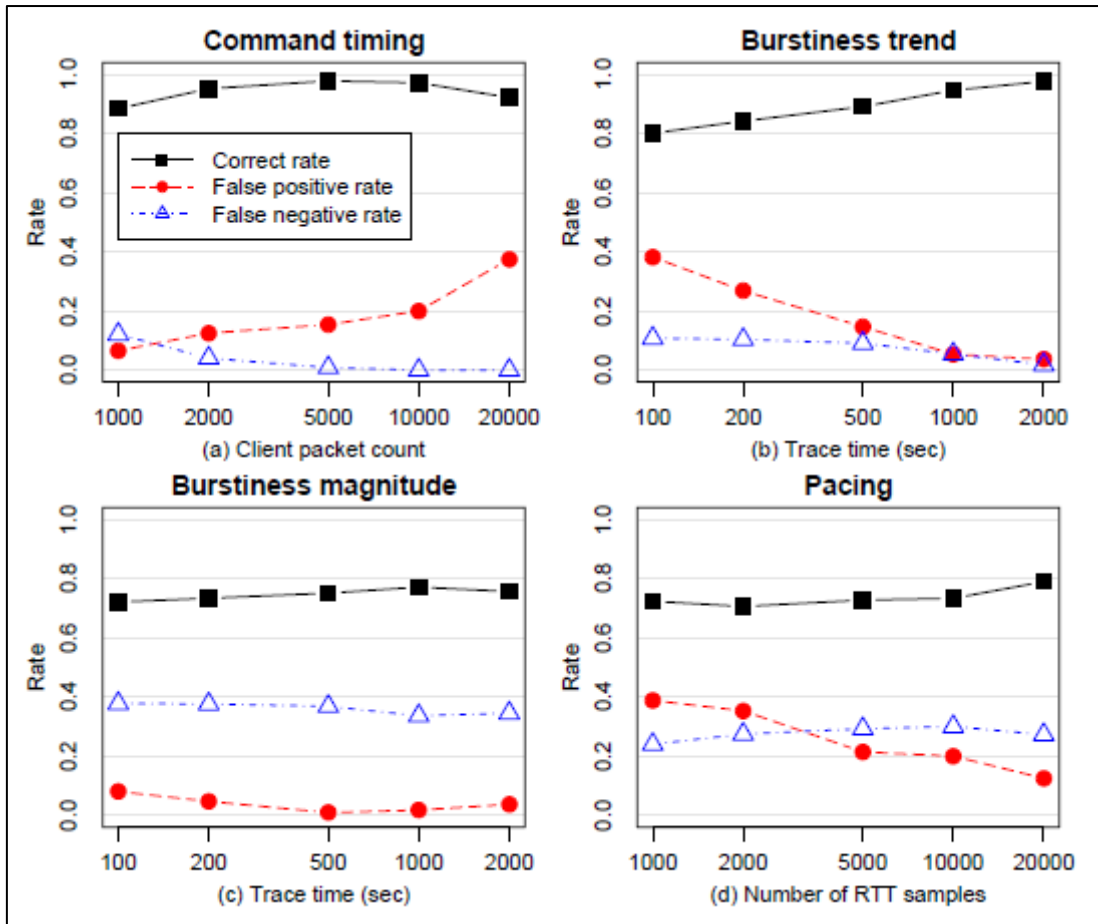
The second method is to observe the traffic burstiness of the packet arrival process. Traffic burstiness is the variability of packet counts sent in successive order. It is an indicator of how traffic fluctuates over time. The idea is that since there is a periodicity to bot traffic, the burstiness should be smoother when compared to human players. The authors note that bot traffic burstiness shows an initial falling trend and then a rising trend. Using this idea, they analyze the trace looking for this trend. As seen in

Figure 19b, when classifying a bot using this technique there is a maximum 95% correct rate. With an increase in trace time, there is a rapid decrease in the false positive rate and a small decrease in the false negative rate



**Figure 18: Histogram of client response times shorter than 0.5 seconds (Chen, 2009)**

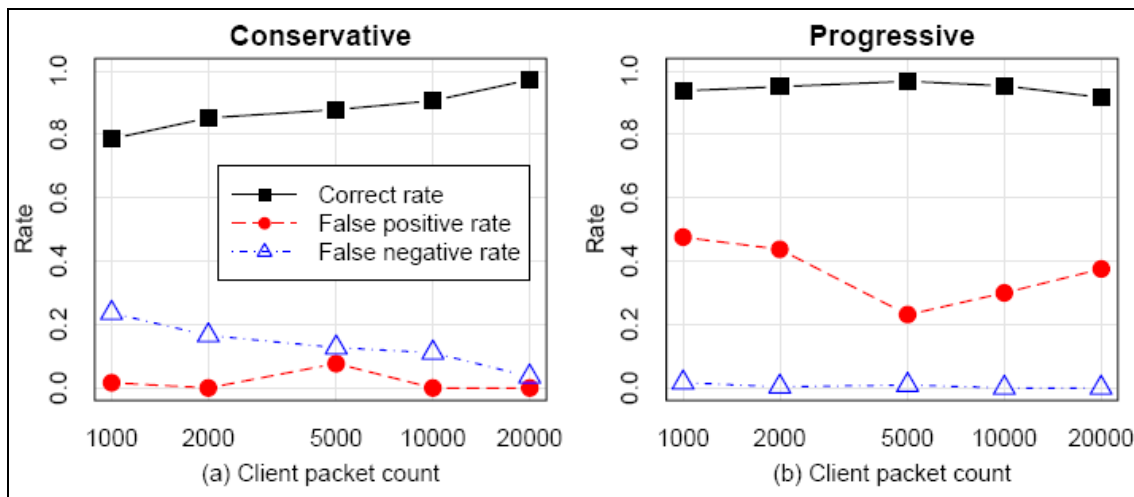
The third method, like the second method, uses traffic burstiness. The authors note the ratio of client burstiness versus the server burstiness is less than one for bots and greater than one for players. Therefore, they look for this in burstiness magnitude. As seen in Figure 19c, when classifying a bot using this technique there is a maximum 75% correct rate. There is a constant false positive rate of 40% and a constant false negative rate of 10%.



**Figure 19: Evaluation results for the proposed decision schemes with different input size (Chen, 2009)**

The fourth method is to identify particular patterns in human behavior caused by sensitivity to network conditions. The hypothesis is that human players subconsciously adapt to network delay. Therefore, we expect a negative correlation between the round trip times and the packet rate. As seen in Figure 19d when classifying a bot using this technique, there is a maximum 80% correct rate. The false positive rate decreases as the number of round trip time samples increase, while the false negative rate remains about the same.

Using just the command timing and burstiness trend methods, the authors also investigate an integrated approach, where they apply both methods simultaneously. They used a conservative approach where both methods needed to agree for bot detection and a progressive approach where only one method had to identify a bot. Figure 20, shows that the conservative approach has an 80% correct detection rate with a low false positive rate of around 1%. The correct detection rate increases to around 95% after 20,000 client packets. The progressive approach starts and maintains around a 95% correct detection rate, but has a high false positive rate of about 40%.



**Figure 20: Evaluation results for the integrated schemes (Chen, 2009)**

This analysis is a good initial start, but has some immediate limitations. The data was collected solely from one MMOG. The observations used to identify trends and patterns in the packet traces could be MMOG specific. The same idea goes to the bots. Only two bot programs were tested, and therefore the packet patterns may be associated with just those two bots. The results are also dependent on how many client packets you



have to analyze. Looking at the raw data, to get the 20,000 client packets we would need over an hour long packet trace. If this method was used to classify bots, a lot of time and resources would need to be used to collect an hour's worth of TCP packets from each player to be analyzed.

#### **4.3.2. Initial TCP Analysis with Artificial Neural Networks**

To set up the data for our analysis, traces are broken into five minute sections and then features are collected and calculated for each of the five minute traces. We arbitrarily decided on using five minute traces. We feel this should be a sufficient amount of time since it contains around 1500 client and server packets total. Thus, it should have a sufficient number of observations to determine the mean and variance we are looking for to characterize our features. Longer or shorter traces may also work well.

As seen in Table 6, ten numeric features and five binary features were chosen to represent the data. The numeric features were chosen since they were referenced in the article by Chen et al. (2009). They referenced the distributions of the response times and the interarrival rates. We use the mean and variance to represent these distributions. The last four features were chosen since we hypothesized that a player would tire after a time playing, while a bot would remain consistent. The authors also noted that there was a change in servers every time a player changed zones within the game. Therefore, we added our first binary variable to identify if there was a server change. There were 143 player observations and 726 bot observations.

We separated the data using a simple hold out method. It is pessimistically biased (Kuncheva, 2004), and compared to other methods of training the data such as cross-validation or jackknife, it requires less computation. We decided to split the data into

60% training, 20% testing, and 20% validation sets. Results are reported using Apparent Classification Accuracy (ACA) since we are concerned with finding both bots and players.

**Table 6: Feature Overview of Initial Transmission Control Protocol Data from *Ragnarok Online***

	<b>#features</b>	<b>#observations</b>	<b>ratio of 0/1</b>	
<b>Bot</b>	15	869	143/726	
<b>column #</b>	<b>features</b>	<b>Data types</b>	<b>min</b>	<b>max</b>
1	Mean client response	Numeric	0.0339	0.7513
2	variance client response	Numeric	0.0013	0.5115
3	mean server response	Numeric	0.0967	1.6558
4	variance server response	Numeric	0.0098	406.9200
5	mean client interarrival rate	Numeric	0.3932	5.6178
6	variance client interarrival rate	Numeric	0.0154	398.6100
7	mean server interarrival rate	Numeric	0.1927	6.1613
8	variance server interarrival rate	Numeric	0.0101	175.0200
9	packets sent from client to server	Numeric	0.1807	2.5442
10	packets sent from server to client	Numeric	0.1636	5.1918
11	change zone	Binary	0	1
12	trace 0-1 hr	Binary	0	1
13	trace 1-2.5 hr	Binary	0	1
14	trace 2.5-4 hr	Binary	0	1
15	trace > 4 hr	Binary	0	1
16	human (1) or bot (2)	Binary (converted to 0,1)	1	2

We used a hybrid GRNN method to model the data, separately processing the binary features using a Hamming distance GRNN and the numeric data using a Euclidian distance GRNN. To combine the two separate models, we created a convex combination of the Hamming distance GRNN with the Euclidian distance GRNN, as seen in equation (50), and then developed confusion matrices. We optimized the parameters for both individual GRNNs and the convex combination parameter to maximize ACA. We also

calculated confusion matrices using standard FFNN, GRNN, and PNN for comparison. We used MATLAB ANN functions to calculate the comparison confusion matrices.

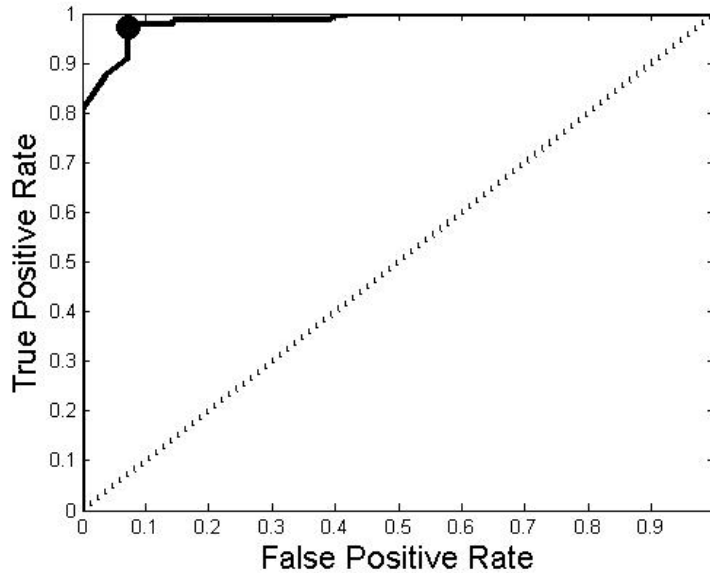
The results for the hybrid GRNN, along with GRNNs for both the binary and numeric data, can be seen in Table 7. The majority of the classification information is obtained from the binary model, since the binary and the hybrid GRNN confusion matrices are almost identical. Note that the hybrid GRNN is slightly better than the binary GRNN, indicating some useful classification information was gained from the numeric part. The increase in ACA may not be statistically significant, but with only a single replication, we can't perform a t-test to evaluate the significance. Also, this information is highly dependent on the placement of the bot cut-off. Figure 21 is the ROC curve generated from the hybrid GRNN. The large dot on the graph identifies the current cut-off. The graph shows that the cut-off is ideal for identifying bots. It maximizes the probability of bot detection, with the lowest chance of identifying false bots. It also shows that if we change the cut-off to reduce the number of false bots detected, we will dramatically reduce the number of bots detected.

**Table 7: Hybrid confusion matrices for Bot detection**

SIM	TRUE					
	Hybrid		Binary		Numeric	
	player	bot	player	bot	player	bot
player	26	4	24	5	24	58
bot	2	142	4	141	4	88
ACA	0.9655		0.9843		0.6437	

Part of this analysis was also to determine the effectiveness of the hybrid GRNN. Table 8 shows a comparison between FFNN, GRNN, PNN, and the hybrid GRNN. We can see that the hybrid GRNN has the best ACA over all. Even though the FFNN does

correctly identify three more bots than the hybrid GRNN, the hybrid GRNN correctly identifies five more players.



**Figure 21: ROC curve of Hybrid GRNN for Bot detection**

**Table 8: Comparative Confusion Matrices for Bot Detection**

SIM	TRUE							
	FFNN		GRNN		PNN		hybrid GRNN	
	player	bot	player	bot	player	bot	player	bot
player	21	1	25	5	24	7	26	4
bot	7	145	3	141	4	139	2	142
ACA	0.9540		0.9540		0.9368		0.9655	

### 4.3.3. Second TCP Analysis with Artificial Neural Networks

After we completed our first analysis we obtained more bot data. Since this imbalanced our data even more and we were interested in testing our methodology for an even larger and more imbalanced data set, we performed the analysis a second time with this larger bot data set. Table 9 is a summary of the increased data set. Note we switched

the labeling for the response to make the binary 1 represent the smaller category (players), which is also the category of interest.

**Table 9: Feature Overview of Second Transmission Control Protocol Data from *Ragnarok Online***

	#features	#observations	ratio of 0/1	
<b>Bot</b>	15	1886	1743/143	
column #	features	Data types	min	max
1	Mean client response	Numeric	0.0158	1.9543
2	variance client response	Numeric	0.0002	7.3513
3	mean server response	Numeric	0.0967	3.2598
4	variance server response	Numeric	0.0098	406.9200
5	mean client interarrival rate	Numeric	0.3932	79.3510
6	variance client interarrival rate	Numeric	0.0154	18327.0000
7	mean server interarrival rate	Numeric	0.1927	122.3800
8	variance server interarrival rate	Numeric	0.0101	44729.0000
9	packets sent from client to server	Numeric	0.0211	2.5442
10	packets sent from server to client	Numeric	0.0127	5.1918
11	change zone	Binary	0	1
12	trace 0-1 hr	Binary	0	1
13	trace 1-2.5 hr	Binary	0	1
14	trace 2.5-4 hr	Binary	0	1
15	trace > 4 hr	Binary	0	1
16	human (1) or bot (0)	Binary	0	1

We chose to alter this analysis by using the ‘leave-one-out’ validation method since we are setting up to perform a similar analysis with a much larger data set that is severely imbalanced. In addition, we wanted to focus on the category that has the fewer number of observations, so we selected F-Measure for players as our performance measure. For comparison we also included ACA and F-Measure for bots.

Looking at Table 10 it can be seen when looking at F-Measure for players, labeled FM (player), that the binary GRNN did not identify any players. This could be

due to the extreme imbalance of observations combined with the difficulty in using binary data for discrimination. The numeric GRNN identified 96 of the 143 players. When combined into our hybrid GRNN, we can see that it performed better than the numeric GRNN alone. This indicates that even though the binary GRNN did not identify any players on its own, it does add value to our hybrid GRNN. When looking at the other performance measures, our hybrid GRNN did better than both the binary GRNN and the numeric GRNN separately.

**Table 10: Hybrid Confusion Matrices for Bot Detection**

SIM	TRUE					
	Binary GRNN		Numeric GRNN		Hybrid GRNN	
	bot	player	bot	player	bot	player
<b>bot</b>	1743	143	1737	47	1738	40
<b>player</b>	0	0	6	96	5	103
<b>FM(player)</b>	0		0.8142		0.8207	
<b>bot</b>	1743	143	1739	50	1740	42
<b>player</b>	0	0	4	93	3	101
<b>FM(bot)</b>	0.9606		0.9867		0.9872	
<b>bot</b>	1743	143	1737	47	1738	40
<b>player</b>	0	0	6	96	5	103
<b>ACA</b>	0.9242		0.9751		0.9761	

We also compare the results of our hybrid GRNN to PNN, regular GRNN and FFNN, as in Section 4.3.2. Table 11 shows the comparison with other ANNs. Using F-Measure for players we can see that our hybrid GRNN does better than FFNN, and comparable to PNN, but not as well the standard GRNN. Using the F-Measure for bots and ACA, our hybrid GRNN performs better than FFNN and PNN, but falls a little short of the standard GRNN (within 0.001).

**Table 11: Comparative Confusion Matrices for Bot Detection**

SIM	TRUE							
	PNN		GRNN		FFNN		Hybrid GRNN	
	bot	player	bot	player	bot	player	bot	player
<b>bot</b>	1731	34	1734	34	1707	126	1738	40
<b>player</b>	12	109	9	109	36	17	5	103
<b>FM(player)</b>	0.8259		0.8352		0.1735		0.8207	
<b>bot</b>	1733	36	1736	36	1707	126	1740	42
<b>player</b>	10	107	7	107	36	17	3	101
<b>FM(bot)</b>	0.9869		0.9878		0.9547		0.9872	
<b>bot</b>	1731	34	1734	34	1707	126	1738	40
<b>player</b>	12	109	9	109	36	17	5	103
<b>ACA</b>	0.9756		0.9772		0.9141		0.9761	

For large scale data sets, the performance measure is not the only factor to consider. Processing time can be an important issue. For both the GRNN and the PNN, determining the spread parameter takes up the majority of the processing time. While with FFNN, it may take a long time to identify an optimal solution. In Table 12, we can see the time in seconds for determining the models used in Table 11. The table is only one observation, but gives an idea of how long this process takes. The search for spreads for the PNN and all of the GRNNs were ranged from 0.1 to 1 in increments of 0.1 while the FFNN was calculated in MATLAB using the default goal of 0 and only 50 epochs. The FFNN was repeated until a model that achieved the goal was reached. The hybrid GRNN did not employ any parallel processing, i.e. the binary model was calculated, then the numeric model was calculated, then they were combined. We can see the PNN, GRNN and FFNN all take about the same amount of time to compute, while our hybrid GRNN was much faster, approximately one eighth of the time.

**Table 12: Times for Creating Confusion Matrices for Bot Detection**

<b>ANN Method</b>	<b>Time in Seconds</b>
<b>PNN</b>	8778.4700
<b>GRNN</b>	8666.8721
<b>FFNN</b>	8076.7130
<b>Hybrid GRNN</b>	1114.2307

#### **4.3.4. Conclusion**

Our initial experiment resulted in showing ANNs could easily be implemented by a game company and within a few minutes a player could be identified as a bot or a human accurately. In fact, looking at our hybrid GRNN confusion matrix, 99% of the bots were identified correctly, with only 6% of the players falsely identified as bots. Using the ROC curve in Figure 21, we can adjust these results to increase the number of identified bots or decrease the number of false positives. We can also see that the hybrid GRNN method has the best overall performance in the initial data set.

The results for the second experiment are similar to the initial experiment, except our hybrid GRNN narrowly failed to outperform the standard GRNN for all three performance measures. The area where it did outperform the other techniques is in the amount of time it required to complete. The time performance of our hybrid GRNN was 7.8 times faster than the standard GRNN, bringing 2.4 hours processing time down to about 18 minutes. When compared to the slight decrease in classification performance, the increase in time performance could be critical. We also take into account that this modeling technique is intended to be used on extremely large datasets, around ten times the size of the second bot data set.



#### **4.4. Summary**

This chapter introduces our hybrid GRNN technique. Our technique leverages the concept of modeling mixed data types by using individualized GRNNs and then combining them back using a convex combination of the individual GRNNs. It reviews work by Chen et al. (2009) on bot detection from TCP traces and uses the same data as an example application of our technique. The example is based on a mixed data set containing binary and numeric data, but the technique is not restricted to only numeric and binary data.

## V. Spread Estimation for Classification with Large Data Sets

The use of a Generalized Regression Neural Network (GRNN) requires a spread parameter. The spread is a smoothing parameter for the Gaussian distributions within the Parzen Window. Choosing a proper spread is an important part of developing a GRNN model. To identify an optimal spread for the model, it is best to identify the smallest parameter that gives the best selected classification performance measure. Using the smallest spread parameter within a desired performance range is generally considered more representative since it should reduce the number of false classifications. For our model, the spread is set for both calculating the model's performance measure and identifying a feature for removal, and is recalculated each time a feature is removed, since the training data and possibly the performance measure, changes with the removal of a feature.

The standard method of finding a spread is to input different spreads and select the spread associated with the best performing model. Since the GRNN model needs to be developed multiple times with different spread parameters for comparison, this process can be time consuming. The addition of very large data sets with features that have large ranges can complicate this even further by taking longer to calculate and requiring more spread parameters to check. Another problem that can arise when using the standard method and performance measures like F-Measure, is that all of the performance measure results are equal. This makes it difficult to compare the results from different parameters when there may actually be a difference between the outputs.

To combat this problem we tried using common alternatives, but felt the results were not comparable with the standard method. Therefore, we developed a method that attempts to identify a spread which adequately represents the distribution of the data within each category of the training set. This method is faster than a standard exhaustive search, and results in comparable spreads.

This chapter begins with a background review of different spread finding methods. We then present the methodology for our new spread finding method. The next section compares our method to the reviewed methods using multiple data sets. The last section provides a summary of the chapter.

## **5.1. Background**

The standard method of finding a spread is to test different spreads and choose the one that gives the model the best performance when evaluating the test set. With the data standardized, we only need one spread for all classes. To be thorough when identifying the best spread, users typically cycle through values from 0.1 to a user specified cap (i.e. 1) with a preferred step size of 0.1. GRNN models are fairly robust (Specht, 1990) to spread parameters so a step size of 0.1 should not adversely affect the model, especially if the data has been standardized. Standardizing will help compensate for data with extremely small variances or extremely large variances. When selecting the optimal spread, there may be multiple spread values with the same value of the performance measure. The best performance measure relative to the test set, with the smallest spread should be chosen.

The standard spread finding method is effective and quick for smaller data sets, but with large data sets, it can be very time consuming. An alternative to spending long

run times evaluating different spread parameters is using a heuristic method. According to class notes (Bauer, 2008), there exists a spread heuristic for finding spread for radial basis functions which are similar to GRNNs in that they both use Parzen Windows to develop an underlying distribution. Equation (51) shows how to use the size of the training set ( $M$ ) and the number of features ( $N$ ) to identify the spread heuristic.

$$\sigma = \frac{1}{(2M)^{\frac{1}{N}}} \quad (51)$$

Another quick heuristic method to selecting spread is to use the standard deviation of the training set as the spread. There are a variety of ways to calculate such a standard deviation to characterize the spread of the underlying distribution. Some suggestions follow

- The standard deviation can be calculated across all training points with a single mean.
- The training set can be divided into categories and the standard deviation within the categories can be averaged across categories.
- The training set can be divided into multiple centroids and the standard deviation within the centroids can be calculated and then the standard deviation across all centroids can be averaged.

Both using the spread heuristic and standard deviation are extremely fast ways of identifying a spread parameter compared to the standard method. This is because they do not require generating multiple GRNNs. Our method is similar in that aspect, but differs in the idea that our method focuses on adequately representing the distribution of data within each category instead of the recommended training set statistics.

## 5.2. Methodology

Our research investigates data from Massive Multiplayer Online Games (MMOGs) which can be extremely large data sets. Therefore, to reduce the time it takes to model this data, we develop a method to identify a spread that allows the model to adequately cover each classification space. As outlined in Figure 22, we first separate the training data set into its different categories and then identify clusters within each category. We use *X*-means (Pelleg, 2000) to identify clusters for each category for our research. *K*-means could also be used to identify clusters, but it requires the analyst to specify the number of clusters, a priori. We then compute the centroid for each cluster and assign each point within the categories to its nearest centroid. Then for each point assigned to the centroid we find the distance to its nearest neighbor within that centroid. We then find the average nearest neighbors distance for the entire training set and half it. This gives us a value for  $\sigma$  that covers the majority of the category space.

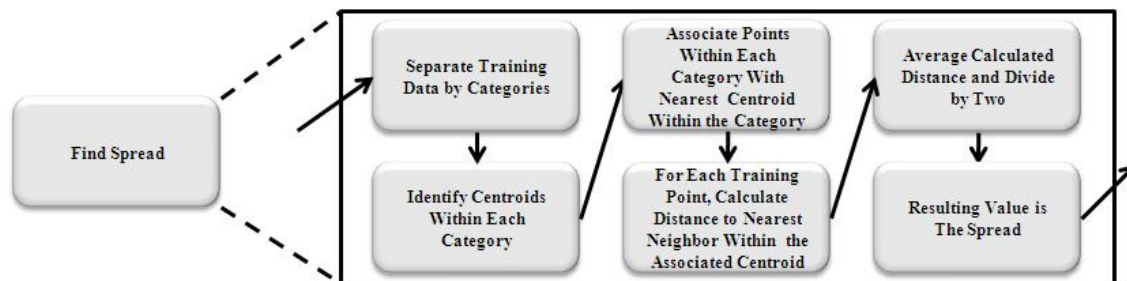


Figure 22: Flow Chart of Faster Spread Finding For Extremely Large Data Sets

## 5.3. Multiple Data Sets Comparisons

To test the effectiveness of our heuristic when compared to the standard exhaustive approach, we looked at different data sets and different sizes of data from our

largest data set. Our goal was to find a fast technique that gave us comparable results to exhaustive search. We compared performance against the standard technique using both a GRNN employing Euclidian distance and a GRNN using Hamming distance.

### **5.3.1. Data Overview**

We used three different data sets to test the different spread finding techniques, initially looking at comparisons with a GRNN employing Euclidian distance. They are the Wisconsin Breast Cancer (Cancer) data outlined in section 3.3.1, the Bot detection (Bot) data outlined in section 4.3, and the *EverQuest II* (EQ2) data outlined in section 6.1. Cancer represents a small data set, since it only has 16 features and 683 observations. Bot represents a larger dataset with both binary and numeric data. It has five binary features and ten numeric features with 1,886 observations. We also looked individually at both the binary and numeric features of the Bot data, labeled Bot-bin and Bot-num respectively, along with all features combined. EQ2 represents a very large data set with both binary and numeric data along with severely imbalanced categories. It has seven binary features and 20 numeric features with 21,377 observations where the category ratio is 216/21,161. We looked individually at both the binary and numeric data for EQ2, labeled EQ2-bin and EQ2-num respectively, along with a full feature set. Since we are also interested in determining if there was an effect to the spread finding technique with a reduced feature set; we randomly sampled the EQ2 feature set to look at  $\frac{1}{4}$ ,  $\frac{1}{2}$ , and  $\frac{3}{4}$  of the features labeled EQ2-1/4, EQ2-1/2, and EQ2-3/4 respectively.

For our second set of comparisons we looked at a GRNN employing Hamming distance. We used five different subsets of the EQ2 data. The first four sets are categorical data points converted into binary by creating a feature for each category type.

The fifth set is a group of binary features from the original set. Table 13 lists the binary sets with the number of features and observations.

**Table 13: EQ2 data sets with the number of features and observations**

<b>Data Set</b>	<b>Features</b>	<b>Observations</b>
<b>EQ2-1</b>	16	21377
<b>EQ2-2</b>	41	21377
<b>EQ2-3</b>	14	21377
<b>EQ2-4</b>	13	21377
<b>EQ2-5</b>	24	21377

### 5.3.2. Analysis

We performed comparisons using both an exhaustive search optimizing ACA (ACA) and an exhaustive search optimizing F-Measure (F-M) with results from our heuristic (OH), the notes heuristic (NH) in equation (51), standard deviation of all data (STDEV), average standard deviation between categories (CAT STDEV), and average standard deviation within centroids between categories (CENT STDEV).

Table 14 displays the identified spreads for each data set for the different spread finding methods. All methods are compared to ACA and F-M. To compare techniques using ACA, we bolded the result from the technique with the closest spread to the ACA spread. We see that none of the techniques did a great job, but if we placed a minimum of 0.1 on all the techniques, similar to the standard technique, OH would be the best match for four data sets and a very close second for another (OH 0.1 difference from Bot goal while CENT STDEV 0.09 difference from Bot goal). For the last five data sets none of the techniques were close. We also looked at using F-M instead of ACA. In Table 14 we highlighted in grey the spread of the closest technique to the spread of F-M. These results show six of ten data sets where the OH spread was the closest to the goal F-M.

Similarly as shown for ACA, if we place a minimum on the spread, OH is the best match for two additional data sets and a close second for another.

**Table 14: Spreads by Collection Type and Data Set**

Data set	ACA	F-M	OH	NH	STDEV	CAT STDEV	CENT STDEV
Cancer	<b>0.8</b>	0.8	<b>0.9373</b>	0.6368	1.0007	1.0007	0.4877
Bot-bin	<b>0.1</b>	0.1	0.0003	<b>0.1926</b>	0.3688	0.3688	0.2266
Bot-norm	<b>0.1</b>	0.1	<b>0.0696</b>	0.4389	1.0003	1.0003	0.5734
Bot	<b>0.2</b>	0.2	0.0915	0.5775	1.0003	1.0003	<b>0.2973</b>
EQ2-bin	<b>0.1</b>	0.1	0.0006	0.2180	0.3465	0.3465	<b>0.0704</b>
EQ2-norm	<b>2</b>	0.4	0.4176	0.5867	<b>1.0000</b>	<b>1.0000</b>	0.9374
EQ2	<b>2.1</b>	0.1	0.5935	0.6737	<b>1.0000</b>	<b>1.0000</b>	0.9642
EQ2-1/4	<b>1.8</b>	0.3	0.0688	0.2180	<b>1.0000</b>	<b>1.0000</b>	0.4319
EQ2-1/2	<b>1.8</b>	0.1	0.2036	0.4669	<b>1.0000</b>	<b>1.0000</b>	0.7277
EQ2-3/4	<b>2</b>	0.1	0.4090	0.6018	<b>1.0000</b>	<b>1.0000</b>	0.6846

We are not just worried about how close we are to the exhaustive search spread; we are also concerned with the time it takes to run the spread finding technique. Looking at Table 15, we see that using NH is by far the fastest method. Table 16 displays the fraction of time for each technique compared to the exhaustive search.

**Table 15: Spread Collection Times by Collection Type and Data Set**

	Exhaustive Search	OH	NH	STDEV	CAT STDEV	CENT STDEV
Cancer	76.3427	30.2516	0.0260	0.1069	0.0533	26.8458
Bot-bin	484.6531	73.9003	0.0505	0.0979	0.0519	14.3626
Bot-norm	437.3439	71.8993	0.0506	0.0990	0.0575	67.7969
Bot	537.0606	110.4224	0.0257	0.1233	0.2554	106.4531
EQ2-bin	41481.9545	1984.3360	0.0434	0.1388	0.0995	265.6507
EQ2-norm	81544.5572	2650.7080	0.0262	0.2041	0.7308	2582.6680
EQ2	74930.9486	3154.2850	0.0262	0.2205	0.2651	3060.4550
EQ2-1/4	42012.5673	1010.3420	0.0260	0.1598	0.1102	903.0723
EQ2-1/2	65508.8103	2164.7110	0.0261	0.1792	0.2839	2088.0550
EQ2-3/4	84648.9670	3031.4750	0.0264	0.1913	0.3369	2962.6040



**Table 16: Spread Collection Times Relative to Exhaustive Search Collection Time by Collection Type and Data Set**

	<b>OH</b>	<b>NH</b>	<b>STDEV</b>	<b>CAT STDEV</b>	<b>CENT STDEV</b>
<b>Cancer</b>	3.9626E-01	3.4000E-04	1.4010E-03	6.9900E-04	3.5165E-01
<b>Bot-bin</b>	1.5248E-01	1.0400E-04	2.0200E-04	1.0700E-04	2.9635E-02
<b>Bot-norm</b>	1.6440E-01	1.1600E-04	2.2600E-04	1.3100E-04	1.5502E-01
<b>Bot</b>	2.0561E-01	4.7900E-05	2.3000E-04	4.7500E-04	1.9821E-01
<b>EQ2-bin</b>	4.7836E-02	1.0500E-06	3.3500E-06	2.4000E-06	6.4040E-03
<b>EQ2-norm</b>	3.2506E-02	3.2100E-07	2.5000E-06	8.9600E-06	3.1672E-02
<b>EQ2</b>	4.2096E-02	3.4900E-07	2.9400E-06	3.5400E-06	4.0844E-02
<b>EQ2-1/4</b>	2.4049E-02	6.2000E-07	3.8000E-06	2.6200E-06	2.1495E-02
<b>EQ2-1/2</b>	3.3045E-02	3.9900E-07	2.7400E-06	4.3300E-06	3.1874E-02
<b>EQ2-3/4</b>	3.5812E-02	3.1100E-07	2.2600E-06	3.9800E-06	3.4999E-02

When focusing on just the binary data we only compared OH and NH to the F-M. We use F-M since one of the categories is extremely small compared to the other and we are interested in identifying these observations. Table 17 displays the spread values determined by standard F-M, OH, and NH. It also contains computing times for both OH and F-M. It can be seen that OH and F-M are comparable in their spreads, while NH attempts to use a larger value. The difference between F-M and OH is primarily because they have different minimum allowed values. If they were both set at 0.1, then they would be equal. For the timing, we can again see that OH is a much faster method averaging a 95% reduction rate.

**Table 17: Spread collection times relative to normal collection time along with spread values**

<b>Data Set</b>	<b>F-M</b>	<b>OH</b>	<b>NH</b>	<b>Time for F- Measure</b>	<b>Time for OH</b>
<b>EQ2-1</b>	0.1	0.001	0.5135	84750	3777
<b>EQ2-2</b>	0.1	0.001	0.7710	126400	5900
<b>EQ2-3</b>	0.1	0.001	0.4669	78839	5480
<b>EQ2-4</b>	0.1	0.003	0.4403	99865	1278
<b>EQ2-5</b>	0.1	0.001	0.6413	103660	6690

### 5.3.3. Conclusion

Our analysis results clearly show that NH is a very fast method for determining spread. When compared to exhaustive search, it has up to seven orders of magnitude in reduction. If speed was the sole factor it would be the method of choice. However, when considering accuracy of selecting the optimal spread values, our heuristic does better. In terms of processing time it shows a reduction of up to two orders of magnitude, and with the application of a minimum value it consistently achieves the closest spread to the exhaustive F-Measure spread search. Even though we have stated that GRNNs are fairly robust to spread parameters, large deviations from an optimal spread can severely affect the GRNN output. Therefore, accuracy is important and should be considered along with the speed of obtaining a spread value.

Note that the EQ2 data has 27 features, and for our feature reduction technique, we need to determine the spread 27 times. Therefore, it would take about 23.5 days of computing just to determine the spreads using the exhaustive method, while it would take about one day to compute the spreads using our heuristic.

Noting that the spread values obtained are not exact with our heuristic, a possible variation to identify a spread would be a heuristic to identify a better set of bounds for an exhaustive spread search. Since the heuristics are not bounds nor seem to have any patterns that may indicate how far the true value may be, there is no easy way to modify the equation to reduce the bounds on the exhaustive search. An idea that may work is to test the initial heuristic value and values just outside the heuristic to identify a search direction where the test performance measure improves. Thus continuing testing new spreads in the search direction until the performance measure fails to improve. This

method would not be as fast as the heuristic alone, but would be faster than the exhaustive search. This method, would potentially result in a better spread estimate than the heuristic, but possibly not identifying the best spread parameter as in a full exhaustive search.

#### **5.4. Summary**

This chapter introduces a new spread finding technique. Our heuristic technique attempts to identify a spread that will allow the training data to cover the space associated with each category of response. Our technique is not exact when compared to typical exhaustive searches, but is more accurate than other suggested techniques. Our technique runs faster than exhaustive searches as seen in our example, and will dramatically reduce the running time of our feature reduction technique that requires finding a spread many times.

A possible variation for future work was suggested in 5.3.3. It suggested using the heuristic as an initial spread and using it along with spreads around it to identify a search direction and then use exhaustive search from there to identify a local optimal spread.

## **VI. Analysis of ‘Gold Farmers’ in *EverQuest II* Using Feature Selection and Hybrid Generalized Regression Neural Networks**

In our search to identify research involving classification of players in Massive Multiplayer Online Games (MMOGs), we discovered the Virtual World Exploratorium (VWE). Through a previous agreement with Sony Entertainment, this group has access to massive amounts of data collected from *EverQuest II*. Fortunately, we were able to contact researchers in this group who allowed us to work with them and apply our full hybrid Generalized Regression Neural Network (GRNN) technique to their data. The subgroup we worked with focuses on the identification of gold farmers. Gold farming refers to the practice of trading virtual in-game resources such as currency, items, and avatars for real-world currency. Gold farming is considered a deviant behavior for three main reasons (Keegan, 2010). First, in game economies are carefully developed by the game developers and gold farmers upset the balance of these economies. Second, gold farmer’s activities often adversely affect the playing experience of other players. Third, gold farming assigns a real-world value to virtual property bringing with it questions about property rights and taxation along with criminal activities such as money laundering.

We felt identifying gold farmers using the *EverQuest II* data was a good opportunity to apply our full technique. This is because the data is actual MMOG data with associated truth data and identifying gold farmers is identifying deviant activity. The process of identifying other deviant activities should have similar properties and

challenges; such as an extremely unbalanced proportion between deviant (the category of interest) players and all other players.

This chapter is divided into five sections. The first section is a brief introduction to the analysis. The second section is a description of the *EverQuest II* data. This will include a description of the original size of the data and the subsequent reduction of the data being analyzed. The third section highlights many of the problems associated with working with this data. The fourth section focuses on the analysis of the *EverQuest II* data using our hybrid GRNN with our feature selection and spread finding heuristic. The fifth and final section contains the conclusion to the chapter.

### **6.1. Data Description**

Anonymized database dumps were collected from Sony Online Entertainment's MMOG *EverQuest II*. This data contains attribute data on individual characters. The full data set is approximately 30 terabytes. Using primarily information within the character attributes we were able to obtain 2.1 million observations with 21 thousand identified gold farmers. The gold farmers were players whose accounts had been canceled with reasons stating bot, farmer, launderer or spammer. The reasons were manually input into the database so many accounts had to be individually checked. The use of 2.1 million observations can be extremely taxing on both memory and CPU time, so a one percent random sampling of the data was used for our analysis. This sampling maintained the proportion of gold farmer and non-gold farmer. Table 18 is an overview of the data used where category '0' is a normal player and category '1' is a gold farmer.

**Table 18: Feature Overview of the 1% Sample Data Collected From *EverQuest II***

<b>1% EQ2 Data</b>	<b>#features</b>	<b>#observations</b>	<b>ratio of 0/1</b>	
	27	21377	21161/216	
<b>column #</b>	<b>features</b>	<b>Data types</b>	<b>min</b>	<b>max</b>
1	RACE	Categorical	0	15
2	CLASS_ID	Categorical	0	40
3	GENDER	Binary	0	1
4	BANK_COIN	Integer	0	721130000
5	PERSONAL MONEY	Integer	0	458990000
6	CHAR_LEVEL	Integer	1	70
7	ARTISAN_CLASS_ID	Categorical	0	13
8	TRADESKILL_EXPERIENCE	Integer	0	40673
9	TRADESKILL_LEVEL	Integer	1	70
10	GUILD_BIN	Binary	0	1
11	LAST_NAME_BIN	Binary	0	1
12	NPC_KILLS	Integer	0	872120
13	TOTAL_DEATHS	Integer	0	2692
14	TOTAL_QUESTS_COMPLETED	Integer	0	1771
15	CURRENT_QUESTS_ACTIVE	Integer	0	85
16	BIO_TEXT_BIN	Binary	0	1
17	STAT_ITEMS_CRAFTED	Integer	0	164420
18	AGE_SECONDS	Integer	0	29977000
19	STAT_RECIPES_KNOWN	Integer	0	3889
20	CITY_ALIGNMENT	Categorical	0	2
21	ACTIVE_GUILD	Binary	0	1
22	ACTIVE_GLOBAL_PERSONA	Binary	0	1
23	PVP_DEATHS	Integer	0	1517
24	PVP_TOTAL_KILLS	Integer	0	3085
25	PVP	Binary	0	1
26	USER_GENDER	Binary	1	3
27	COUNTRY (converted to regions)	Categorical	1	25
28	FARMER (PLAYER = 0, FARMER = 1)	Binary	0	1

For this data, all integers were standardized, potentially resulting in non-integer values. The binary numbers were not standardized. The categorical data was converted to binary and feature reduction was performed to reduce the number of features. To convert categorical data to binary, each unique observation was converted to a binary feature. Therefore, a feature like CLASS\_ID which has 41 unique observations, was converted into 41 binary features.

For validation purposes, we generated two extra sets from the original data. The first set, labeled (500/500), contains 1,000 observations consisting of 500 gold farmers and 500 non-gold farmers. The original ratio does not need to be maintained since this is just a check of the model we built, and does not affect parameters or how well the model performs. The second set, labeled (5%), is a random sample of five percent of the data. This set retains the original ratio, so there are 106,879 observations with 1,076 gold farmers and 105,803 non-gold farmers.

## **6.2. Issues Relating to Large Data Set Sets**

For our research, we were fortunate to gain access to the *EverQuest II* data. After the excitement and elation subsided from getting to work with the data, the realization that it added a lot more work set in. Our first step was moving the small, 2.1 million observation, subset of data from the server where all the data is stored to a local desktop personal computer (PC). After we moved the data, we were able to load it into MATLAB running on a Windows PC with 4 Gigabytes of RAM. Unfortunately, after we loaded all the data into a table within MATLAB, we were unable to perform any mathematical operations upon it. The data was too large and even the simplest operation would cause the program to run out of memory. We attempted different methods to

allow us to maintain the integrity of the whole data set, such as breaking the data into smaller tables and storing them on the hard drive and swapping them out to work on them separately and performing calculations on each cell individually. All of the methods we tried were taxing on the computer system and performed very slowly. Therefore, we decided to use a small random sample (maintaining the ratio of categories) from the original data set to do our analysis.

Even with one percent of the original data, the analysis was very time consuming. To speed the analysis up we focused on process and algorithmic improvements, optimizing code, and application of High Performance Computers (HPCs). We modified our analysis in two major ways. The first was to modify the application to use our spread heuristic discussed in Chapter 5. The repetitive nature of recalculating spreads can consume an exorbitant amount of time. With our spread heuristic, we drastically reduced the time required for each feature reduction step. The second way we modified our analysis was to perform a separate feature reduction on each set of binary features developed from our categorical features. When we convert our categorical data to binary data, we increase the number of binary features from 8 to 108 features. Performing feature selection on all 108 features together would take a significant amount of processing time, while performing the feature reduction on each of the categorical data separately, dramatically reduced the number of binary features in our full model in a timely manner.

The original coded algorithm was developed and tested using a very small data set. Therefore, the cleanliness and efficiency of the code was not an issue. When a large data set was employed we realized that we needed to clean up the code by removing



unnecessary calculations and optimize performance by better using MATLAB strengths in matrix multiplication. We also looked at how matrices were multiplied in order to minimize the number of operations performed.

Finally, we moved most of our processing to HPCs. Leveraging AFITs relationship with the High Performance Computing Modernization Program (HPCMP), we were able to run multiple analyses simultaneously to reduce the time required to finish. With a bit more time, much of the algorithm could be parallelized to leverage the HPCs more effectively.

Another problem that arose with the application of this data was due to the imbalance of the categories. With the number of gold farmer observations being so small, we decided to use the leave-one-out method for our validation technique. This greatly increased the number of observations and time required to perform the analysis. Along with the time increase, we had a problem with the model classifying everything as non-gold farmer. This resulted in very good ACA since 99% of the data was non-gold farmer, but F-Measure was undefined for gold farmers. Therefore, we need to employ ROC curves to identify a better cut off and force some gold farmer classifications so we can discriminate between reduced training sets. Throughout the remainder of this discussion we use the F-Measure associated with our category of interest, gold farmers.

An additional problem we ran into with the sparse binary data affected the feature selection technique. The calculated sum of many partial derivative magnitudes were zero. This was a problem since we needed to identify the minimum sum of the partial derivative magnitudes and remove the corresponding feature. We solved this problem by removing all features with a sum of the partial derivative magnitudes equal to zero for

feature selection. The added benefit to this method was that it sped up the feature reduction algorithm by reducing the number of times it cycles through the loop.

Since we have no knowledge about underlying data structure to specify the number of clusters, we used *X*-means to identify the number of clusters and the centroids within each cluster. This worked fine for our numeric data, but we discovered for the binary data that some clusters from differing categories had the same centroid. This created a problem when developing vectors of test points between these clusters, since a vector cannot be formed using the same point. To combat this problem, when generating vectors to calculate the summed magnitudes of the partial derivatives, we checked to see if the starting centroid and the ending centroid were the same. If they were the same, we would skip generating the vector. However, our fix could create a situation where no vector was generated between the two categories. Therefore, to guarantee at least one vector is created between each pair of categories, we identify a minimum of two distinct centroids within each category. To make sure we found two or more centroids for each category, we tested the number of centroids *X*-means identified for a category. If it only found one centroid, we abandoned it and used *K*-means, with  $K = 2$ , to identify two centroids. As stated earlier, this guaranteed a vector was developed between each pair of categories.

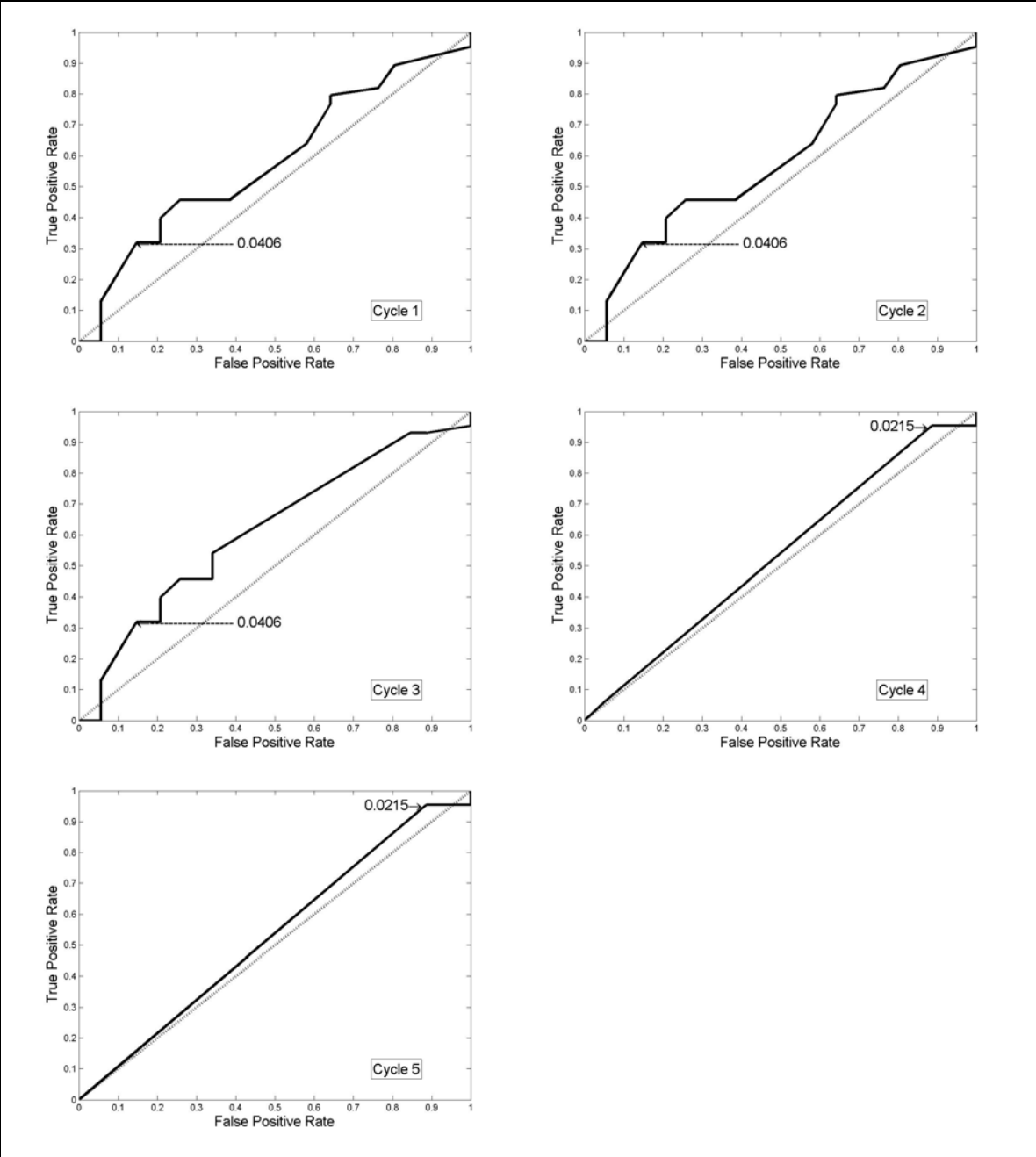
### **6.3. Analysis**

For our analysis we used the leave-one-out validation method. This is because there are so few actual gold farmer observations, and we decided that all 216 observations were too important to eliminate any. We used this validation method every time we evaluated the performance of a model. We focus on F-Measure as our

performance measure. We chose this over our other performance measures because ACA, APER, and maximizing/minimizing confusion matrix values are too heavily swayed by the disproportionate data. Therefore they would not be good representations of the performance of the model. Both recall and precision would be adequate choices for performance measures, but since they have slightly different focuses, neither is best in a general performance measure. Therefore, we decided to use F-Measure since it is a balance of both recall and precision, as seen in Figure 11.

We began our examination with the binary data. We broke this information into five separate data sets (labeled EQ2-B1 through EQ2-B5) and performed a feature reduction on each set. After the feature reduction, we combined all the retained features into a new binary set. Since we frequently obtained zeros for our F-Measure when examining the feature reduction, we created ROC curves for each cycle through the feature reduction technique. Figure 23 is the series of ROC curves built for EQ2-B1. For each cycle the maximum F-Measure is identified relative to the curve, and this is the F-Measure we used as representative of each set of reduced features. All of the binary ROC curves are displayed in Appendix B.

Using this maximum F-Measure for the reduced models, we created a table and graph of retained features versus F-Measure for each of the binary data sets. Table 19 shows the F-Measure relative to the retained features along with the removed features for each cycle using EQ2-B1 data set, while Figure 24 graphically displays the retained features relative to the F-Measure. The decision in which reduced model to use is a balance between the number of features and the accuracy. For all but one of the feature reductions we chose the smallest set with the maximum F-Measure. For EQ2-B3 we

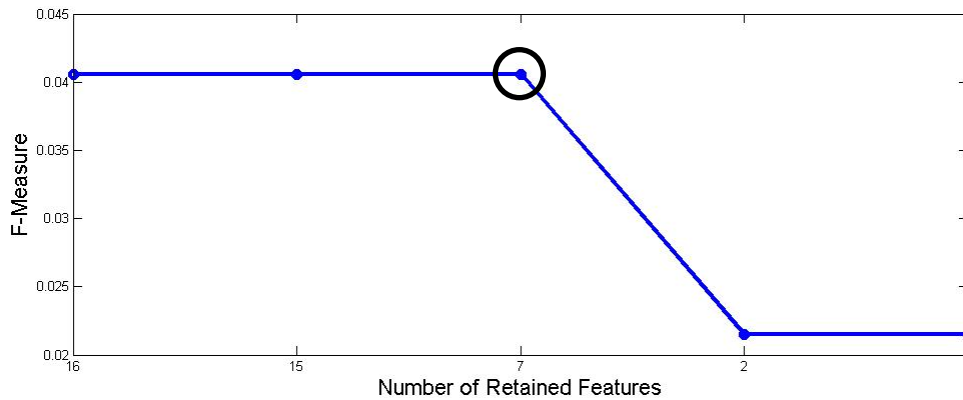


**Figure 23: ROC Curves with F-Measure for EQ2-B1 Feature Reduction Sets.**

noted that the maximum F-Measure (0.0551) contained all the features, while the 2<sup>nd</sup> best F-Measure (0.0535) only required one feature. Therefore, we chose the second best since it only reduces the F-Measure by 0.0016, while reducing the set by 23 features. The shaded cycle in Table 19 represents our selected model for EQ2-B1. Table 20 is a summary of the F-Measures and Modeled Features for each binary data set. The full tables and graphs are located in Appendix C.

**Table 19: Binary Feature Reduction EQ2-B1**

Cycle	Removed Feature	F-Measure	Modeled Features
0	-	0.0406	1-16
1	8	0.0406	1-7,9-16
2	1,5,10,11,12,13,15,16	0.0406	2-4,6,7,9,14
3	3,4,6,7,9	0.0215	2,14
4	14	0.0215	2
5	2	-	-



**Figure 24: Plot of Retained Features vs. F-Measure for EQ2-B1**

Now that we have identified our binary features, we combine them into one data set and generate our binary model. Using our spread finding technique, we calculate raw

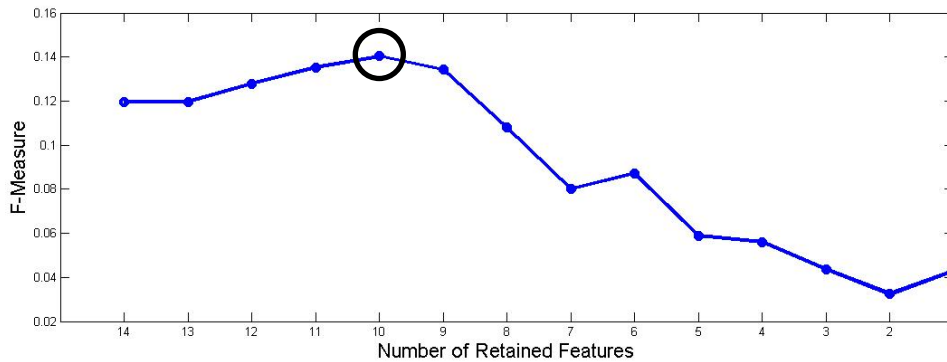
scores for each of the observations to be combined with the raw scores from the numeric data set.

**Table 20: Summary of Feature Reduction for EQ2 Binary Data**

Binary Data Set	F-Measure	Modeled Features
EQ2-B1	0.0406	2-4,6,7,9,14
EQ2-B2	0.0535	24
EQ2-B3	0.0548	1
EQ2-B4	0.0927	1-5,7,10-12
EQ2-B5	0.1485	2,3,5,6,8,10-20,22-24

Similar to the binary data sets, we do a feature selection on the numeric data set.

Table 21 displays the features removed, F-Measure, and the modeled features for each cycle. Figure 25 is the plot of the number of retained features when compared to F-Measure. Using the table and graph we decide on using nine features (shaded in Table 21) for the numeric model.

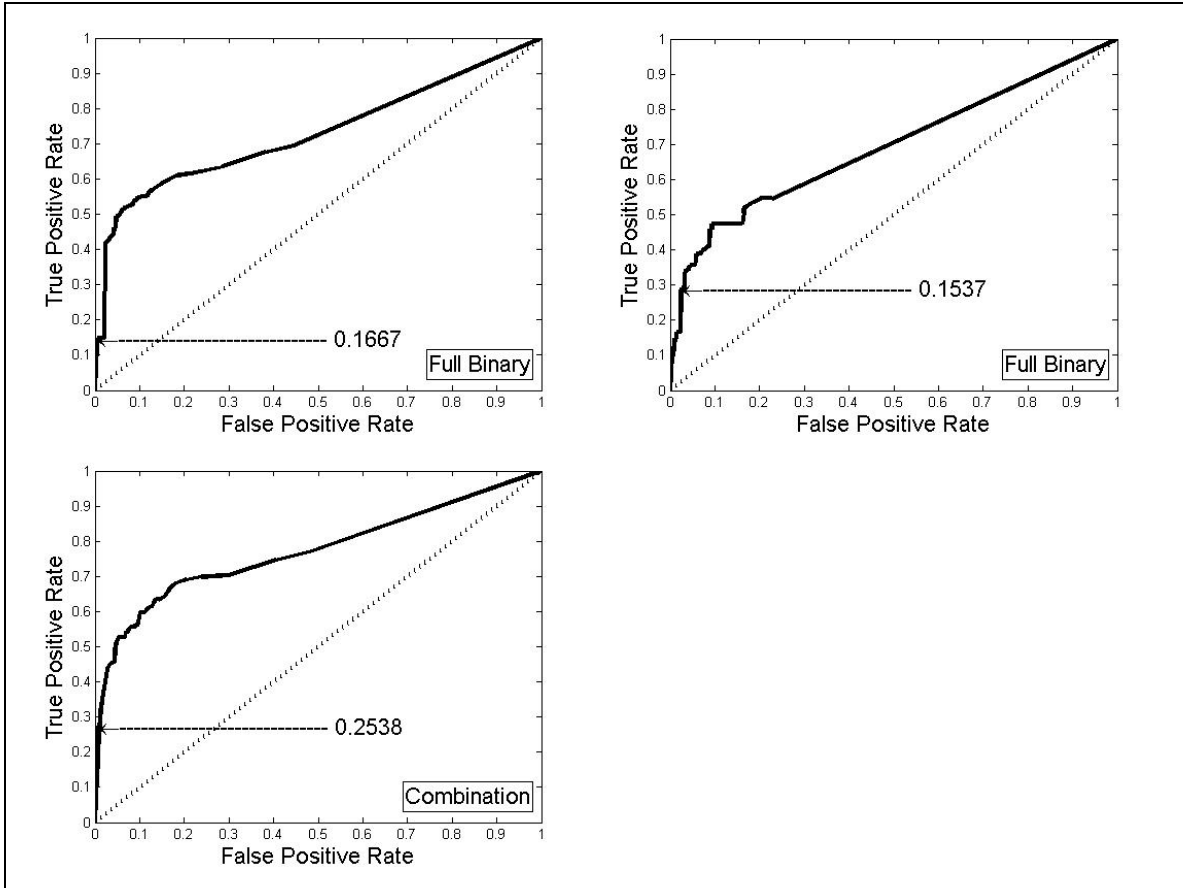


**Figure 25: Plot of Retained Features vs. F-Measure for Numeric Data Set**

**Table 21: Numeric Feature Reduction**

<b>Cycle</b>	<b>Removed Feature</b>	<b>F-Measure</b>	<b>Modeled Features</b>
0	-	0.1197	1-14
1	1	0.1197	2-14
2	4	0.1277	2,3,5-14
3	10	0.1352	2,3,5-9,11-14
4	12	0.1404	2,3,5-9,11,13,14
5	8	0.1343	2,3,5-7,9,11,13,14
6	2	0.1079	3,5-7,9,11,13,14
7	5	0.0800	3,6,7,9,11,13,14
8	3	0.0873	6,7,9,11,13,14
9	14	0.0588	6,7,9,11,13
10	11	0.0561	6,7,9,13
11	9	0.0435	6,7,13
12	6	0.0317	7,13
13	7	0.0429	13
14	13	-	-

Using the raw scores saved from calculating the F-Measure in our feature reduction we can combine it with the raw scores from the combined binary set using a convex combination to identify a new model. Since many of the F-Measures are zero when using a majority of binary data, we use the maximum F-Measure for each convex combination to compare them. Table 22 is the maximum result from the combination, along with the results when the data is only binary and only numeric data. Note that F-Measure for the numeric data is 0.1667 which is larger than the 0.1404 stated in Table 21. This is due to changing the cut off rate in a ROC curve, and since we do that for the binary and final results, we did it for the numeric data for comparison.



**Figure 26: ROC Curves with F-Measure for Final Results**

**Table 22: Results from Convex Combination**

Ratio of Binary to Numeric	F-Measure
1 to 0 (Full Binary)	0.1537
0.07 to 0.93 (Combination)	0.2538
0 to 1 (Full Numeric)	0.1667

Now that we have the results we can use all the parameters to test our new validation set. Table 23 is a summary of all our model parameters. We use the numbered features associated with each data set. The spread for the combined retained binary features is 0.0603 and the spread for the retained numeric features is 0.1247. The convex



combination contains 0.07 of the binary results and 0.93 of the numeric results. The cut off indicates that if the gold farmer model results with a value greater than 0.05 then the observation is from a gold farmer, while a value less than 0.05 is not a gold farmer. The resulting confusion matrix and maximum F-Measure is below in Table 24.

**Table 23: Model Parameters**

<b>Parameter</b>	<b>Value</b>
XB1	2-4,6,7,9,14
XB2	24
XB3	1
XB4	1-5,7,10-12
XB5	2,3,5,6,8,10-20,22-24
XR	2,3,5-9,11,13,14
Binary Spread	0.0603
Numeric Spread	0.1247
Convex Combination	0.0700
Cut off	0.0500

**Table 24: Final Model Confusion Matrix**

<b>Sim</b>	<b>True</b>	
	<b>Non-Gold Farmer</b>	<b>Gold Farmer</b>
<b>Non-Gold Farmer</b>	20971	158
<b>Gold Farmer</b>	190	58
<b>F-Measure (NGF)</b>	0.9918	
<b>F-Measure (GF)</b>	0.2500	
<b>ACA</b>	0.9837	

Looking at the parameters from Table 23 we can see that the convex combination is very small. This indicates that the binary model does not have much of an effect on the results, but there is an effect. On first look the results in Table 24 may not appear to be significant. We see that of the 248 gold farmers identified by the model, only 58 were actual gold farmers and only a small number of gold farmers are identified, 58 out of 216. Another way to interpret the results is to consider that without the model, someone looking for gold farmers would need to investigate 21,377 avatars where only one percent of them are actual gold farmers. With the model, they could narrow the investigation down to 248 avatars where 23% of them are actual gold farmers. It also follows that since we identified 27% of the gold farmers with our reduced set of original features, more gold farmers could be identified by originally starting with more features. Table 25 shows the names of all the retained features from the final model indicated in Table 23. The listing of these features can be used by game experts to gain insight into the behaviors of gold farmers, since they may be able to provide rationale as to why each feature would or would not be associated with a gold farmer.

Since we only used a small portion of the data, we were able to go back into the original dataset and resample validation sets. Using the one percent data as a training set and the parameters identified in Table 23 we processed both validation sets. We standardized each validation observation using the means and standard deviations used to standardize the training set, just as we would to test new observations. Table 26 and Table 27 are the resulting confusion matrices and performance measures from the 500/500 and 5% validation sets, respectively.

**Table 25: Retained Features**

Feature Name	File Name	Feature Label	Data Type	Feature Name	File Name	Feature Label	Data Type
AGE SECONDS	XR	11	Numeric	NORTHERN AFRICA	XB5	11	Binary
ARTISAN CLASS ID 0	XB3	1	Binary	NORTHERN ASIA	XB5	12	Binary
BARBARIAN	XB1	2	Binary	NORTHERN EUROPE	XB5	13	Binary
CENTRAL AFRICA	XB5	2	Binary	NPC KILLS	XR	6	Numeric
CENTRAL AMERICA	XB5	3	Binary	OGRE	XB1	14	Binary
CENTRAL EUROPE	XB5	5	Binary	PACIFIC	XB5	14	Binary
CHAR LEVEL	XR	3	Numeric	PERSONAL MONEY	XR	2	Numeric
CHARACTER GENDER	XB4	1	Binary	PVP	XB4	10	Binary
CHARACTER HAS A BIO	XB4	4	Binary	PVP DEATHS	XR	13	Numeric
CHARACTER HAS A GUILD	XB4	2	Binary	PVP TOTAL KILLS	XR	14	Numeric
CHARACTER HAS A LAST NAME	XB4	3	Binary	SOUTH AMERICA	XB5	15	Binary
CITY ALIGNMENT 0	XB4	5	Binary	SOUTH ASIA	XB5	16	Binary
CITY ALIGNMENT 2	XB4	7	Binary	SOUTH EAST ASIA	XB5	17	Binary
CURRENT QUESTS ACTIVE	XR	9	Numeric	SOUTH EAST EUROPE	XB5	18	Binary
DARKELF	XB1	3	Binary	SOUTH WEST ASIA	XB5	19	Binary
DWARF	XB1	4	Binary	SOUTH WEST EUROPE	XB5	20	Binary
EAST ASIA	XB5	6	Binary	SOUTHERN EUROPE	XB5	22	Binary
EASTER EUROPE	XB5	8	Binary	TOTAL DEATHS	XR	7	Numeric
FEMALE PLAYER	XB4	12	Binary	TOTAL QUESTS COMPLETED	XR	8	Numeric
FROGLOK	XB1	6	Binary	TRADESKILL LEVEL	XR	5	Numeric
GNOME	XB1	7	Binary	WEST INDIES	XB5	23	Binary
HALFLING	XB1	9	Binary	WESTERN AFRICA/EUROPE	XB5	24	Binary
MALE PLAYER	XB4	11	Binary	WIZARD	XB2	24	Binary
NORTH AMERICA	XB5	10	Binary				

When compared to Table 24 the results from Table 27 are similar, while the results from Table 26 are significantly different for F-Measure (NGF) and ACA. This would indicate that category ratio does play a significant part in the evaluation of the data. An interesting note is how F-Measure (GF) is relatively the same for all three data

sets when compared to the changes of F-Measure (NGF) and ACA. F-Measure (GF) is reduced by 0.0810 from Table 24 to Table 27 and is reduced by 0.0642 from Table 26 to Table 27. This is relatively small when compared to the 0.3030 increase in F-Measure (NGF) and the 0.4229 increase in ACA from Table 26 to Table 27. With such a small change, we should be able to claim they are similar. We hypothesize that this is because the model was optimized for this performance measure, and therefore we are confident it will maintain the accuracy of this performance measure with a change in the category ratio and number of observations.

**Table 26: 500/500 Model Resulting Confusion Matrix**

<b>Sim</b>	<b>True</b>	
	<b>Non-Gold Farmer</b>	<b>Gold Farmer</b>
<b>Non-Gold Farmer</b>	500	434
<b>Gold Farmer</b>	0	66
<b>F-Measure (NGF)</b>	0.6974	
<b>F-Measure (GF)</b>	0.2332	
<b>ACA</b>	0.5660	

**Table 27: 5% Model Resulting Confusion Matrix**

<b>Sim</b>	<b>True</b>	
	<b>Non-Gold Farmer</b>	<b>Gold Farmer</b>
<b>Non-Gold Farmer</b>	105568	955
<b>Gold Farmer</b>	235	121
<b>F-Measure (NGF)</b>	0.9944	
<b>F-Measure (GF)</b>	0.1690	
<b>ACA</b>	0.9889	

## 6.4. Conclusion

The primary focus of our research is to identify players in massive multiplayer online games. This research is of interest to both game companies and the government. Game companies can use player classification techniques two ways. One is to identify players and relate them to their activities, so they can improve game play for specified player types. Second they could also identify deviant players such as gold farmers. This would help them identify these deviant players faster and improve game play for all other players. The government wishes to use player classification techniques similar to game companies looking for deviant players. The government is looking for criminal activities such as money laundering and terrorism. These criminal activities are easily hidden within a game, but focus on actions that are not representative of normal game play.

In this chapter, we attempt to classify gold farmers from data obtained from the Sony's online game *EverQuest II*. We chose this example to apply our new spread finding, feature selection, and hybrid GRNN techniques for two reasons. First we have access to a real MMOG database to apply our techniques to. Therefore, we can develop our techniques using real world applications. Second, we do not know of any actual money launderers or terrorists within the dataset, but we do have a list of accounts canceled due to gold farming techniques. Since gold farming is a deviant behavior and the proportion of gold farmers to non-gold farmers is very small, it is similar to our application of identifying criminal activity.

The analysis was successful even though the F-Measure values were relatively low, due at least in part to the reduced data set required for this study. The number of available features to use is extremely large, but our computers and software limited us to

a greatly reduced feature set. We primarily focused on data relating to an avatar's standard information, since it was readily available and allowed us to keep the number of features to a level where our computers could process them without running out of memory. The data used does not contain information about game play such as experience per hour, time played per session, and interaction with other players to name a few. A game developer within *EverQuest II* would have better understanding of the data and could collect many more features to start with than we did.

In addition to the large number of available features, we also had an extremely large number of usable observations, of which we only used one percent. A computer programmer with access to a distributed computing resource could take our techniques and parallelize them to run on multiple machines. This would ease memory restrictions and speed up the computation time, thus allowing the use of more features and observations, while increasing the accuracy of the entire model.

Another potential reason for the low number of classified gold farmers is that we only know which accounts were de-activated due to gold farming. We do not know if any of the remaining observations are actual gold farmers that have not been identified. This could account for some of the observations that were classified as gold farmers but were part of the non-gold farmer category. Further investigation into these observations would be needed to verify this.

GRNNs are readily adaptable to follow the changing of the gold farmer activities. By adding recent observations and eliminating the older observations the model can be updated to represent player behaviors as they change over time. The model can be

adjusted either by just changing the training set or by rebuilding the whole model to identify changes in feature significance.

## **VII. Summary Research Contributions and Future Research**

This chapter summarizes contributions made to the fields of Applied Statistics and Simulation, presented in this document. It also provides areas for future study related to the research presented in this document.

### **7.1. Research Contributions**

This section summarizes contributions to the fields of Applied Statistics and Simulation presented in this document.

#### **7.1.1. Feature Selection Using Generalized Regression Neural Networks**

Our research developed a feature selection technique based on Generalized Regression Neural Networks (GRNNs). It is a multi-step technique that initially clusters the training data to identify multiple centroids within each category of the training data. Then it sums the gradient magnitudes of test points along vectors between the centroids of differing categories. Using these summed gradient magnitudes, it identifies the feature with the smallest magnitude and removes this feature from the training set. The technique repeats this loop, starting with clustering the reduced training set, until all features are removed. After all the features are removed a determination is made based on the performance of each reduced set to determine the desired feature set.

Our feature selection technique can be used with both binary and numeric data. Since this technique is based on and leverages techniques particular to the GRNNs, it is a better alternative to use when modeling with GRNNs than other techniques such as factor



analysis, primary component analysis or signal to noise ratio using Feed forward Neural Networks. This is because our feature selection technique discriminates between features using the same methodology the GRNN uses to discriminate between categories.

### **7.1.2. Hybrid Generalized Regression Neural Network**

Our research developed a hybrid GRNN technique that leverages Parzen Window distributions for kernels that are best suited to a specific data type. Our technique separates features by data type then analyzes them with separate GRNNs using the associated kernel for each data type. Then using a convex combination of the separate GRNNs, it combines the multiple results into a single model output.

A standard GRNN leverages only one kernel for the entire data set. Using a single kernel for models with multiple data types could be less accurate since it applies a kernel that performs optimally for some features, but may not be optimal for the rest of the features. Therefore, using multiple kernels that are each tailored to a specified data type will result in more accurate models.

To combine the separate models back to a single model we find the optimal convex combination of the two models. This convex combination represents a weighting measure that indicates the value of the data types relative to each other in the final model.

An additional benefit to using our hybrid method comes from reducing a large GRNN into two or more smaller GRNNs which can be easily parallelized to reduce overall processing time. The smaller individual GRNNs may also avoid hardware and/or software limitations faced by a very large combined GRNN and allow for including more observations and features in the individual GRNNs, leading to more accurate models. All

these things make our hybrid GRNN ideal for modeling data sets with multiple data types.

### **7.1.3. Spread Estimation Technique for Large Data Sets**

Our research developed a spread heuristic to be used with extremely large data sets. It clusters the data for each category and identifies each observation within a cluster. Then it calculates and assigns the distance between each observation and the furthest observation within each cluster. These assigned distances are then averaged and the result is divided by two. This value provides an estimated spread with the intent it should allow the distribution developed from the training data to cover each category.

Our spread estimation technique is much faster than searching multiple spreads for large data sets, but not as fast as most spread finding heuristics. It is more accurate than most spread finding heuristics and can be deemed an adequate tradeoff of speed and accuracy for large data sets. Without our spread estimation technique, our model would either take an excessive amount of time to complete using an exhaustive search, or produce less accurate results using available spread finding heuristics.

This spread finding technique adds another method to the small number of techniques available. It also fills part of the gap between heuristics and exhaustive search by identifying an optimal value with a performance measure that is better than the standard heuristics, but not as good as an exhaustive search. The speed of calculating our spread runs faster than exhaustive search, but slower than the standard heuristic. Therefore, our spread finding technique is a balance of accuracy and speed.

#### **7.1.4. Develop Framework to Classify Players by Predetermined Categories Using Information Obtained Through In-Game Behaviors**

As a part of our original proposal we developed a framework to classify Massive Multiplayer Online Game (MMOG) players using information obtained through in-game behaviors. Using our spread estimation, feature selection, and hybrid GRNN techniques we were able to identify gold farmers in Sony's online game *EverQuest II*. Our results were promising given the difficulty of the classification process, primarily the extremely unbalanced data set with a small number of observations from the class of interest. As a screening tool our method identifies a significantly reduced set of avatars and associated players with a much improved probability of containing a number of players displaying deviant behaviors. With further efforts at improving computing efficiencies to allow inclusion of additional features and observations with our framework, we expect even better results.

Future investigations into the *EverQuest II* data are required to improve the accuracy of identifying players through their in-game behaviors. An idea presented earlier in the document is to identify better features for our model. We do not fully understand or know all the data available in the *EverQuest II* data and further research into the data could identify features that yield better results with our technique. Also, research in the reliance on training and test proportions between the categories may yield better results.

#### **7.2. Recommendations for Future Work**

This section reviews unfinished avenues of research we began but were not able to complete and other recommendations for further work.

### **7.2.1. Develop Tool to Identify In-Game Player Associations and Movement Patterns**

In our research proposal, we envisioned a series of three related contributions starting with our player classification discussed in Section 7.1.4. The next step is to develop a methodology to identify in-game player associations and generate network representations of these associations. This would lead to more features that could be incorporated into our model to identify players in MMOGs using in-game behaviors. It would also aid in quantifying the impact deviant players have on other players.

Additional insight on player classification and associations can be gained from movement patterns. This would require development of a method to track individual avatars and document their activities. Such a method could lead to developing more features to incorporate into our model for player classification. It could also be used as a verification step in a tool for classifying specific categories of players. After identifying a specified player with a model, it could track them and verify that a player is the type of player specified. This research would need to encompass identifying different methods of how to track cooperative play.

### **7.2.2. Feature Selection Improvements Using Generalized Regression Neural Networks**

While working on our feature selection technique we discovered a computational issue. Using the minimum summed magnitude of gradients to determine a feature to remove could result in ties, primarily with value of zero. We did see this occur numerous times with binary data. Our solution was to eliminate all features with a summed

gradient magnitude of zero, but better solutions may exist. We suggest research into this to determine the best course of action when these ties occur.

### **7.2.3. Processing Improvements for Hybrid Generalized Regression Neural Networks**

While working with the hybrid GRNN, we noted that using large training sets requires a significant amount of computer processing time. It has been noted that clustering and using the centroids to reduce the number of training observations speeds up processing a GRNN, but also distorts the balance between categories. A possible solution to this is to weight each centroid by the number of observations it represents. Further research into such an approach or others to improve processing speeds and effectiveness in describing the data is warranted.

Suggested earlier in this document, future work in parallelization of the code would lead to faster run times. Or hybrid GRNN approach is constructed such that several sections could be parallelized and run on multiple computers at the same time. Two examples of this are when the model is broken by data types and during the evaluation of the GRNNs where the observations to be evaluated could be separated to run on multiple nodes. Parallelization would drastically speed up the performance and also could reduce the reliance on heuristics which would increase the accuracy of the results.

### **7.2.4. Spread Estimation for Large Data Sets**

Two ideas worth investigating relating to work on our spread estimation heuristic are to use our heuristic as bounds for exhaustive search and a start for a gradient search. The basic idea would be to identify a confidence interval about our spread estimation

heuristic. The confidence interval would then become the new search interval for the exhaustive method. Further research would be required to identify if there is a way to modify the estimated spread to create bounds for an exhaustive search and the value this would add.

Also, further research would be required to use the spread as a starting point for a gradient search. Since the spread values are from a linear series, it may be possible to use a spread heuristic to identify an initial spread in a gradient search. Performance measures of the initial spread and two more spreads, a small amount larger and a small amount smaller, could be evaluated. These three performance values could then be fitted to a second degree polynomial where the gradient would be used to direct a search toward a maximum or minimum performance measure. A step could be taken in the direction toward the optimal performance measure to identify a new spread. This new spread could be used to identify another gradient direction or could identify the optimal performance measure.

#### **7.2.5. Further Develop Framework to Classify Players by Predetermined Categories Using Information Obtained Through Observed Behaviors**

Our technique can readily be applied to develop a tool for classifying terrorists among online game players by their in-game behaviors. There is some evidence that terrorists have been identified within MMOGs. Using our techniques and classified information about terrorist activities identified within MMOGs a model for classifying terrorists can be created in a similar fashion as we did for gold farmers. This model can then be used to identify other terrorists within these MMOGs. Once identified by our

approach, these terrorists can then be tracked to identify their activities, learn more about their groups, and intervene to protect national interests.

A further extension of our technique can be applied to real life observations, such as video monitored systems. Paths, dress, and actions could all be features extracted from a monitored system similar to MMOGs. Then, using our modeling technique, we could identify situations or people of interest in the monitored system.

## Appendix A: List of Acronyms

ACA	Apparent Classification Accuracy
AMBR	Adaptive Memory-Based Reasoning
ANN	Artificial Neural Network
APER	Apparent Error Rate
ATPM	Action Transition Probability Matrix
CMDS	Classical Multidimensional Scaling
DTW	Dynamic Time Warping
F2T2EA	Find, Fix, Track, Target, Execute, and Assess
FA	Factor Analysis
FFNN	Feed Forward Neural Network
GRNN	Generalized Regression Neural Network
HMM	Hidden Markov Models
KLE	Klullback Liebler Entropy
MMOG	Massive Multiplayer Online Game
MUD	Multi User Domain
PCA	Primary Component Analysis
PNN	Probabilistic Neural Network
RBFNN	Radial Basis Function Neural Network
ROC	Receiver Operating Characteristic
SNR	Signal to Noise Ratio
SOM	Self-Organizing Map
TCP	Transmission Control Protocol
VBS2	Virtual Battlespace 2
<i>WoW</i>	World of Warcraft
PC	Personal Computer



## Appendix B: Binary Receiver Operating Characteristic Curves for EQ2 Analysis

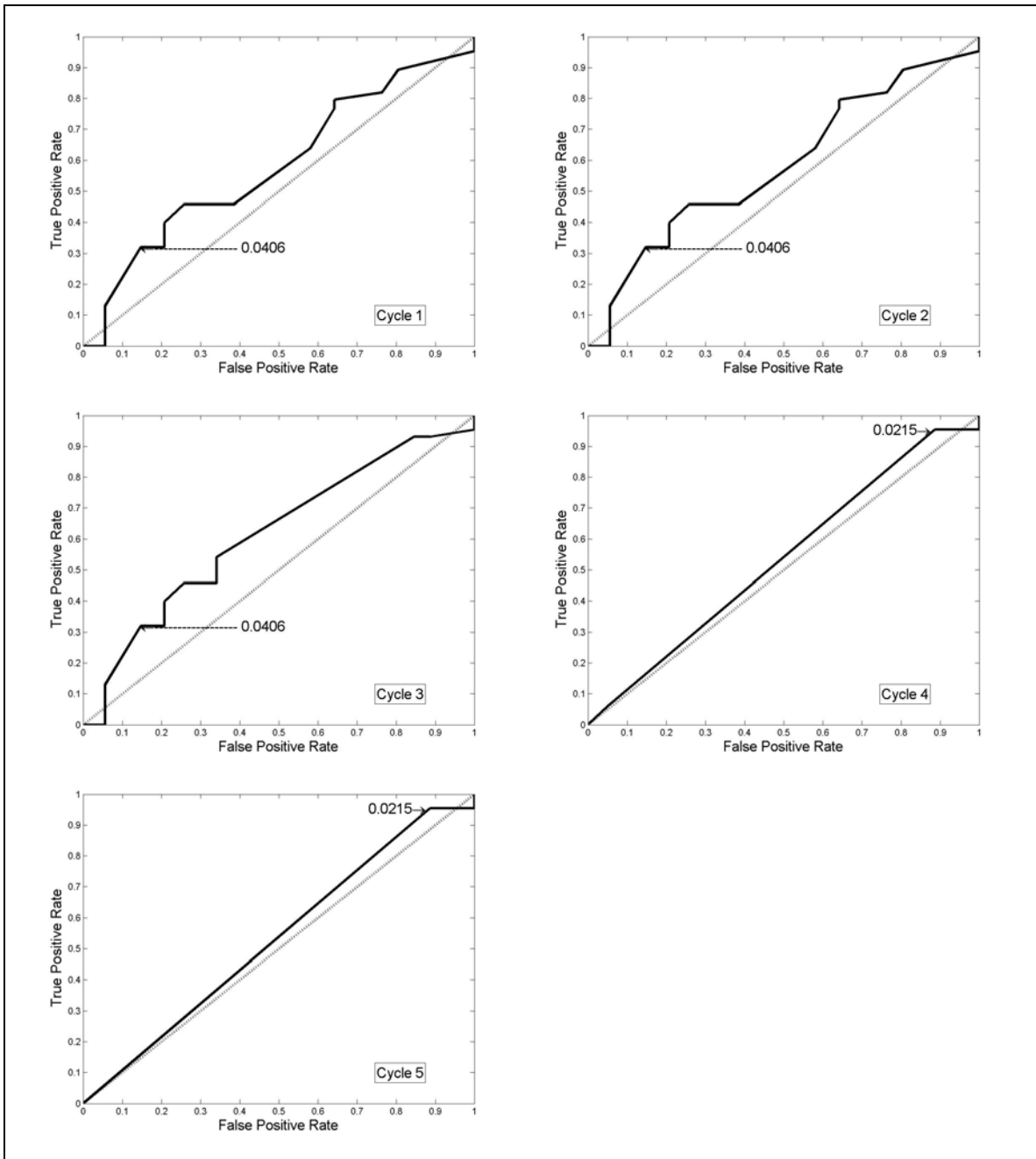
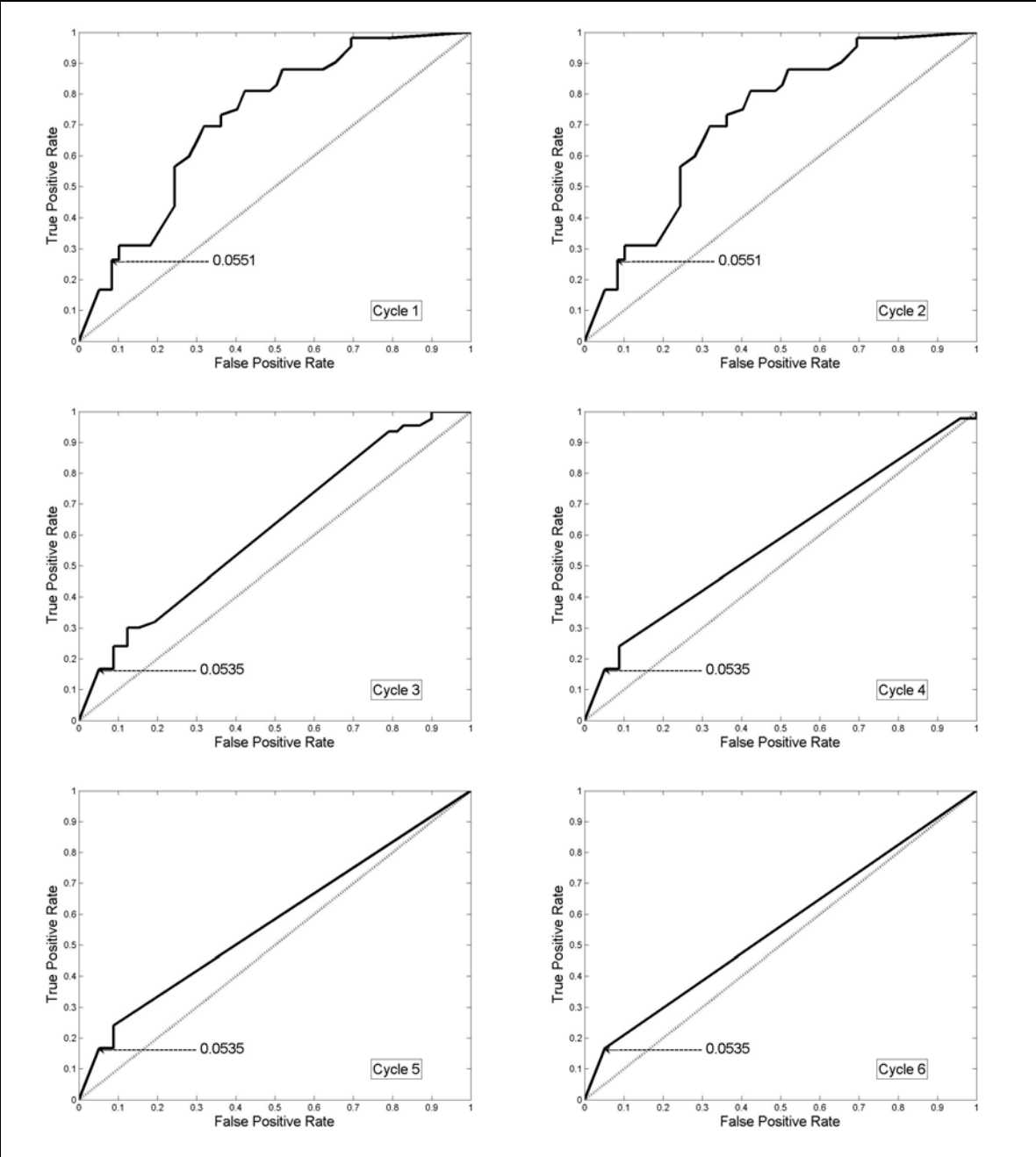
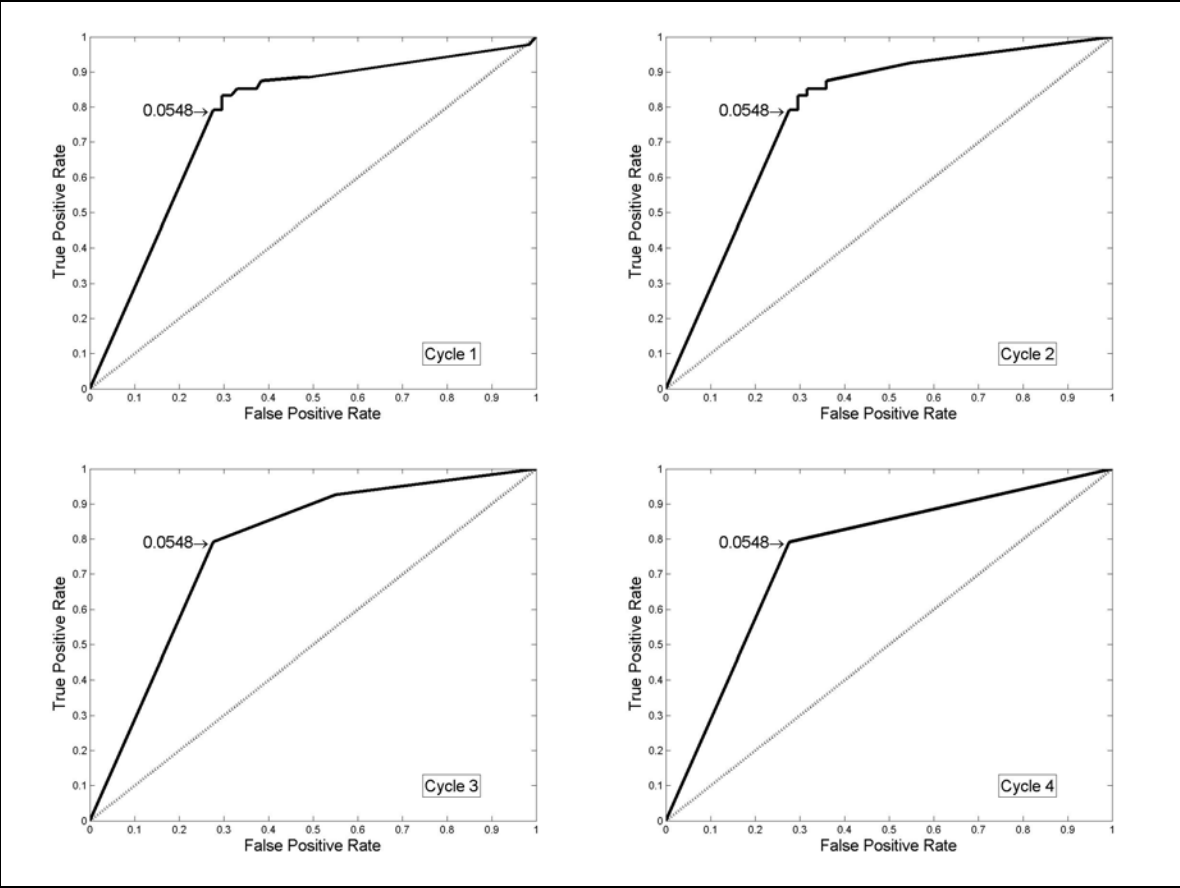


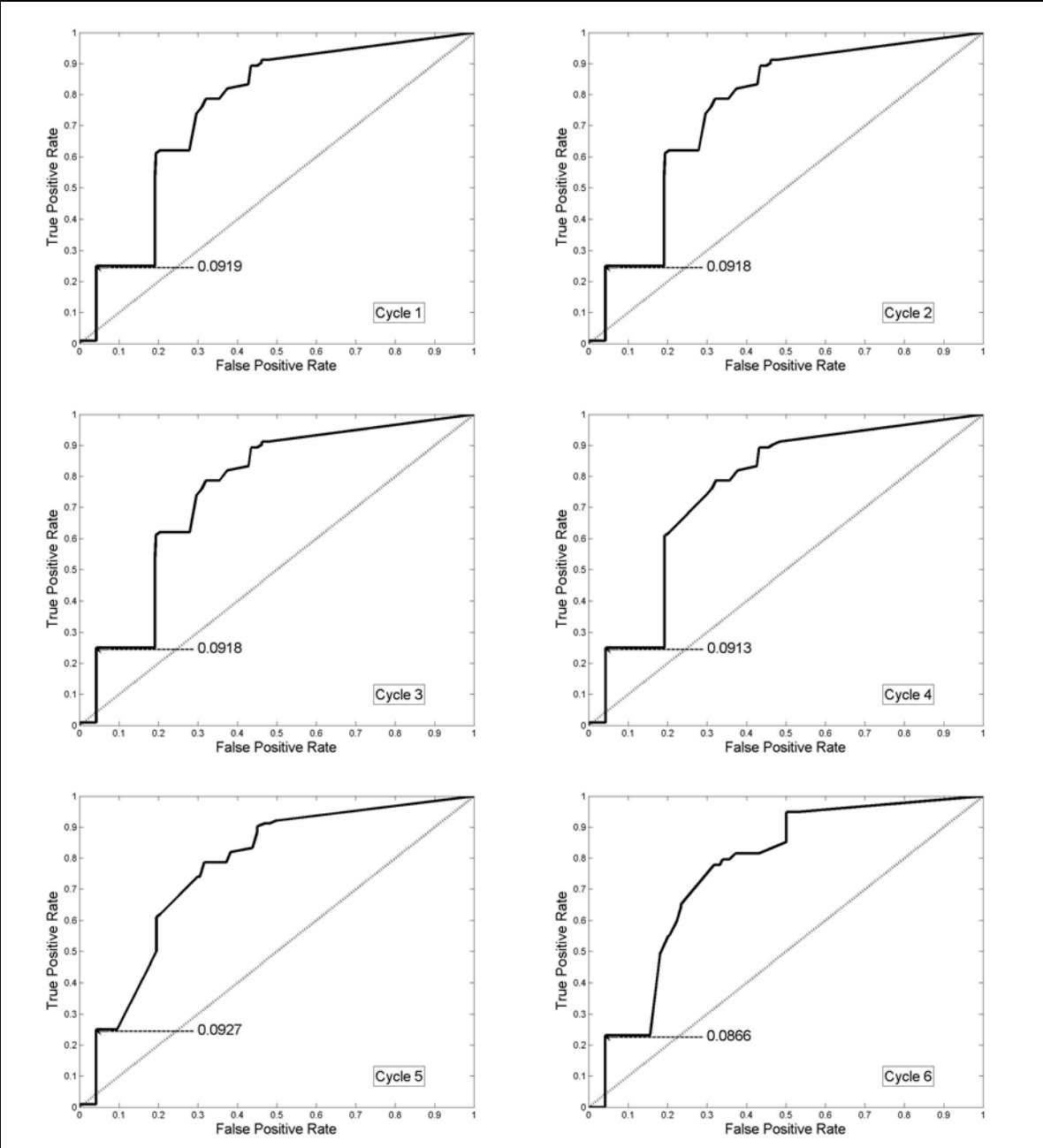
Figure 27: ROC Curves for EQ2-B1 feature reduction sets

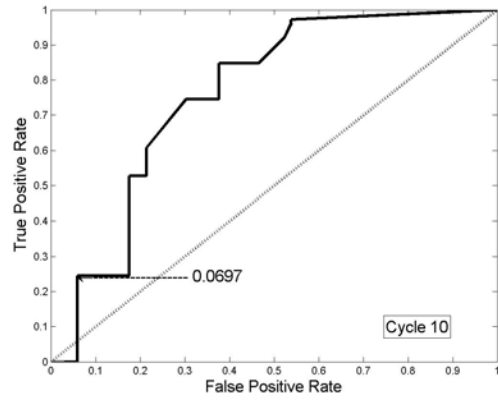
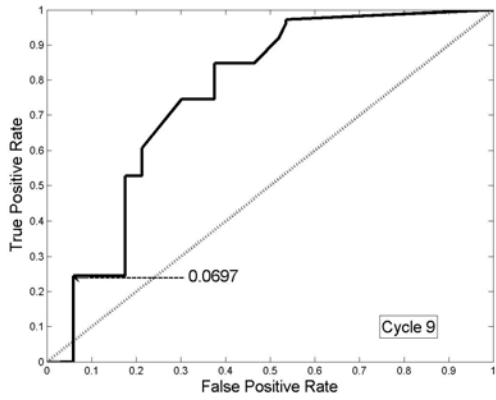
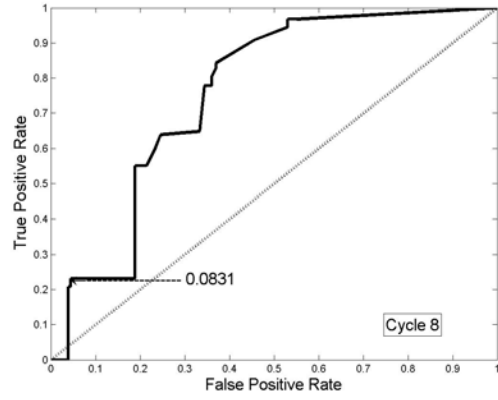
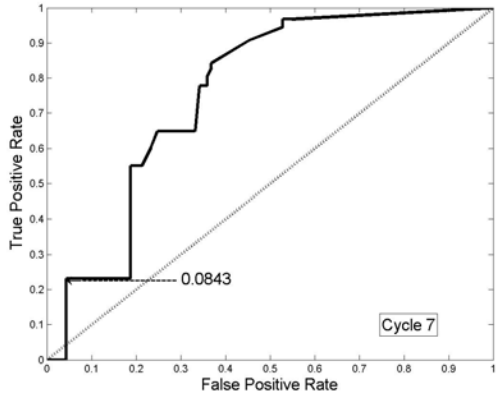


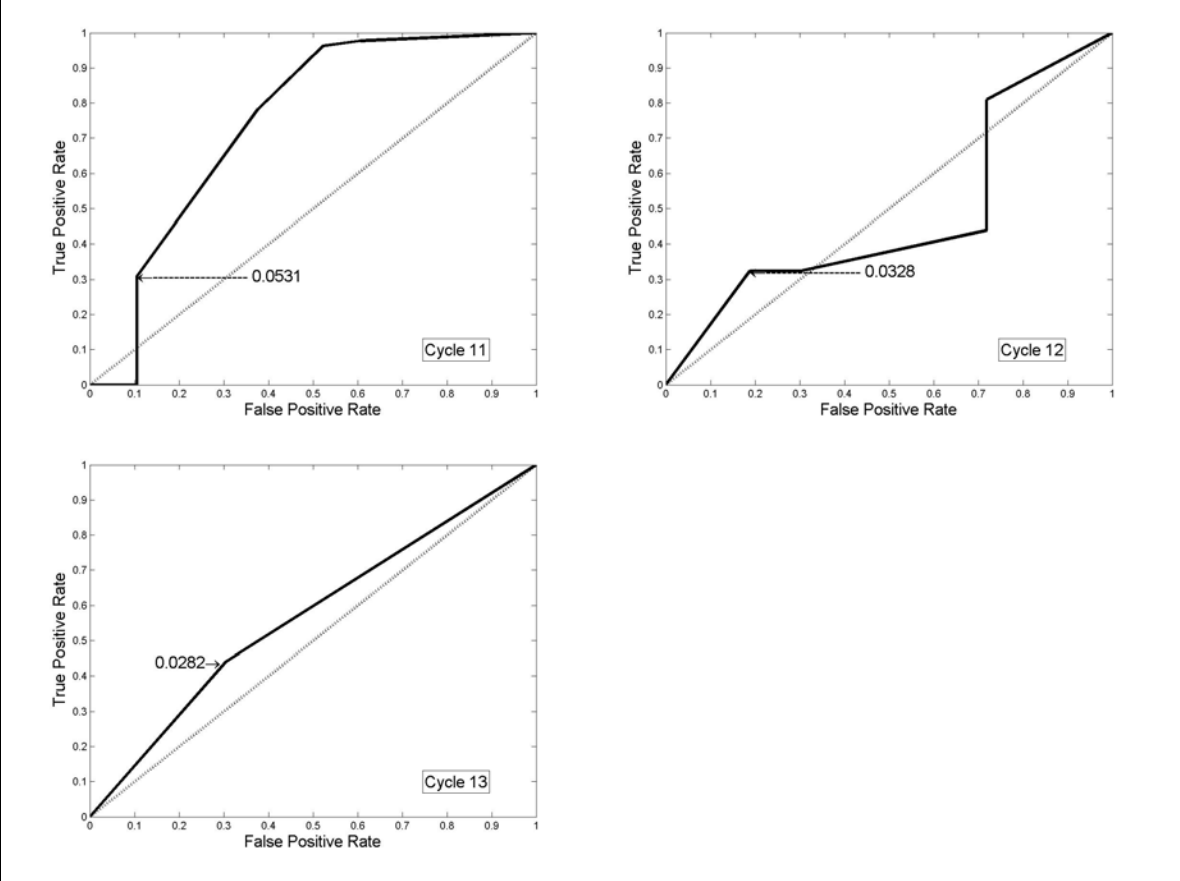
**Figure 28: ROC Curves for EQ2-B2 feature reduction sets**



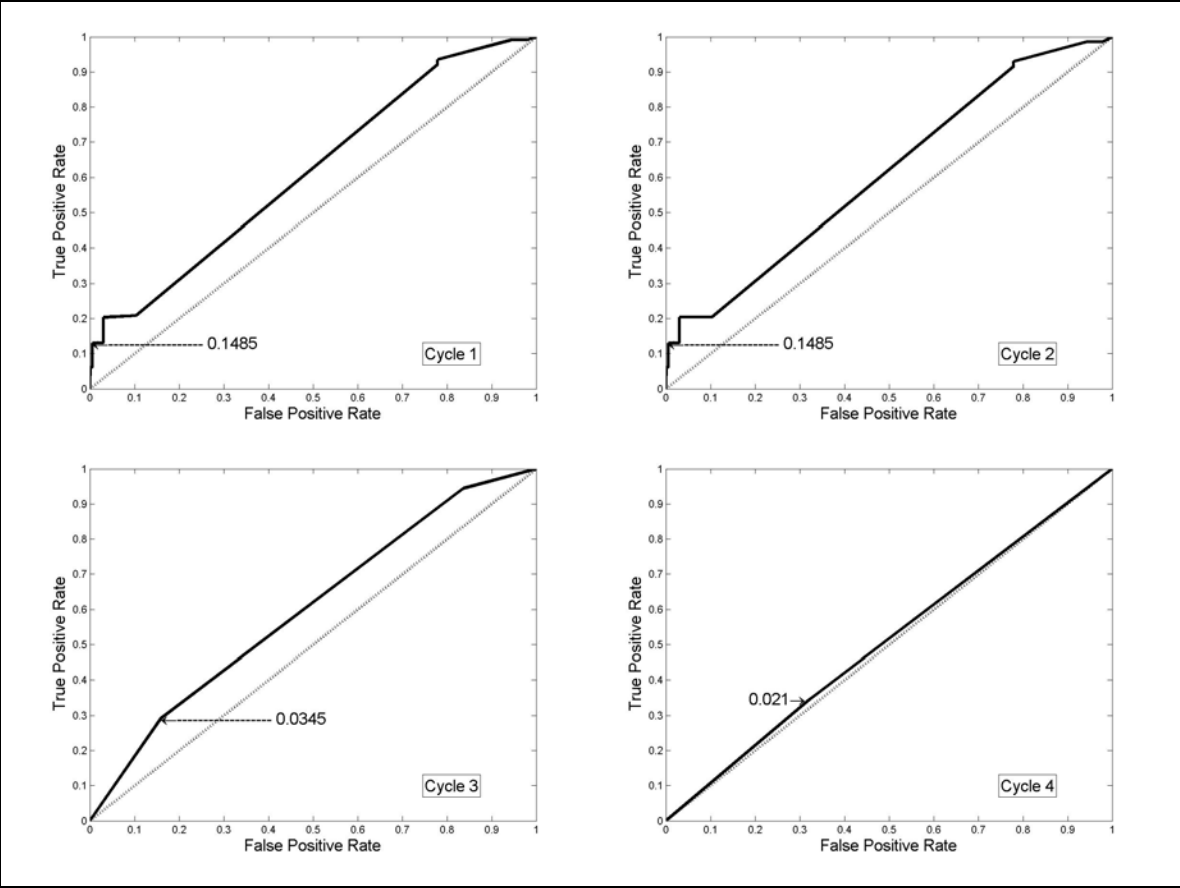
**Figure 29: ROC Curves for EQ2-B3 feature reduction sets**







**Figure 30: ROC Curves for EQ2-B4 feature reduction sets**

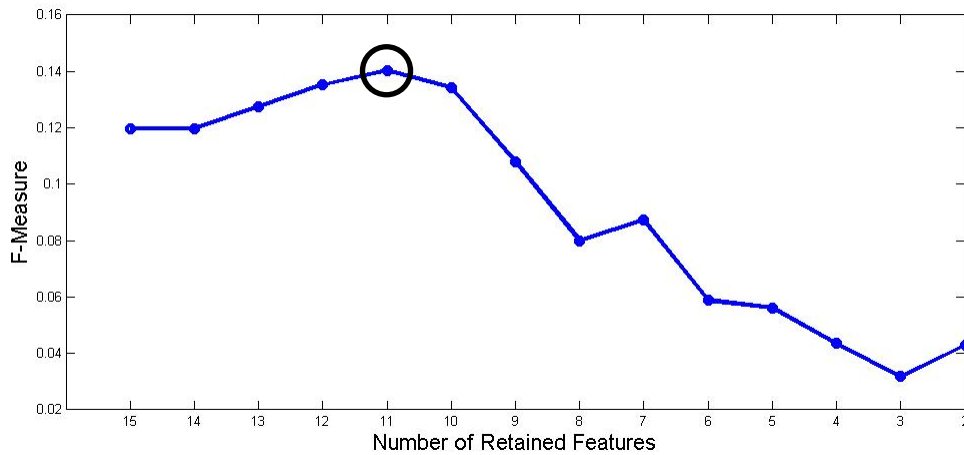


**Figure 31: ROC Curves for EQ2-B5 feature reduction sets**

## Appendix C: Features Reduced vs. F-Measure for EQ2 Analysis

**Table 28: Numeric Feature Reduction**

Cycle	Removed Feature	F-Measure	Modeled Features
0	-	0.1197	1-14
1	1	0.1197	2-14
2	4	0.1277	2,3,5-14
3	10	0.1352	2,3,5-9,11-14
4	12	0.1404	2,3,5-9,11,13,14
5	8	0.1343	2,3,5-7,9,11,13,14
6	2	0.1079	3,5-7,9,11,13,14
7	5	0.0800	3,6,7,9,11,13,14
8	3	0.0873	6,7,9,11,13,14
9	14	0.0588	6,7,9,11,13
10	11	0.0559	6,7,9,13
11	9	0.0436	6,7,13
12	6	0.0324	7,13
13	7	0.0429	13
14	13	-	-

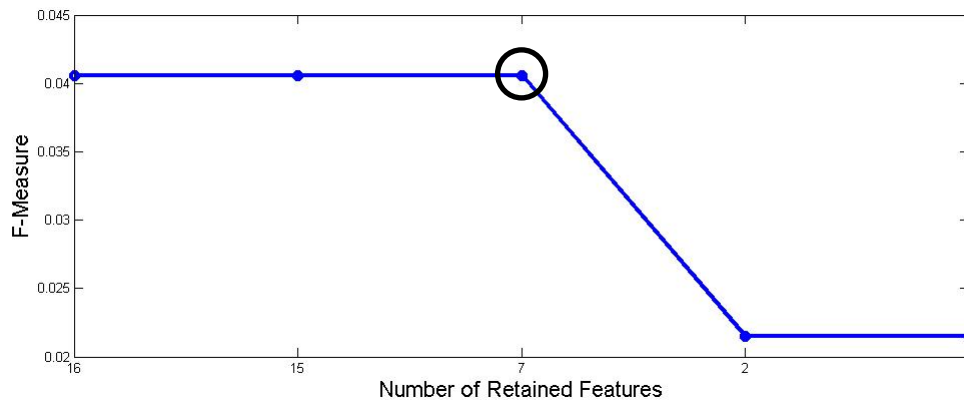


**Figure 32: Plot of Retained features vs. F-Measure for Numeric Data Set**



**Table 29: Binary Feature Reduction EQ2-B1**

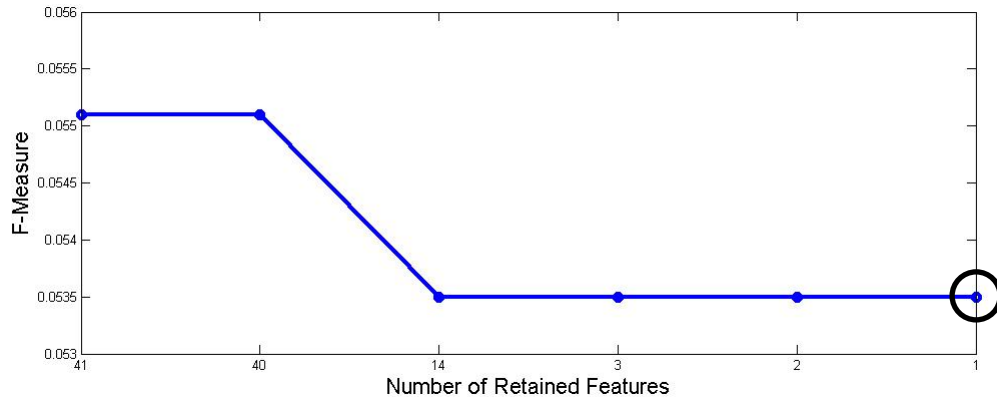
Cycle	Removed Feature	F-Measure	Modeled Features
0	-	0.0406	1-16
1	8	0.0406	1-7,9-16
2	1,5,10,11,12,13,15,16	0.0406	2-4,6,7,9,14
3	3,4,6,7,9	0.0215	2,14
4	14	0.0215	2
5	2	-	-



**Figure 33: Plot of Retained features vs. F-Measure for EQ2-1 Binary Data Set**

**Table 30: Binary Feature Reduction EQ2-B2**

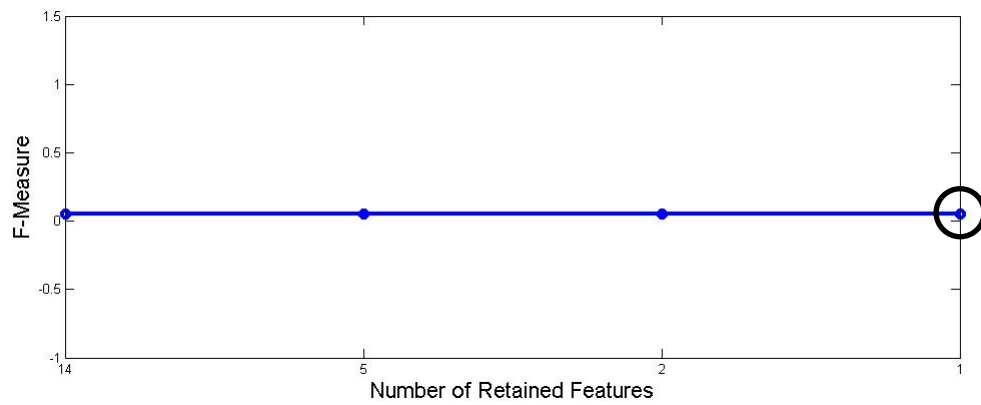
Cycle	Removed Feature	F-Measure	Modeled Features
0	-	0.0551	1-41
1	26	0.0551	1-25,27-41
2	1,2,5,6,9,10,11,13,14, 16,17,18,19,20,22,23, 25,27,28,31,33,34,36, 37,38,39	0.0535	3,4,7,8,12,15,21,24, 29,30,32,35,40,41
3	3,4,7,8,12,15,21,29,32, 35,41	0.0535	24,30,40
4	40	0.0535	24,30
5	30	0.0535	24
6	24	-	-



**Figure 34: Plot of Retained features vs. F-Measure for EQ2-2 Binary Data Set**

**Table 31: Binary Feature Reduction EQ2-B3**

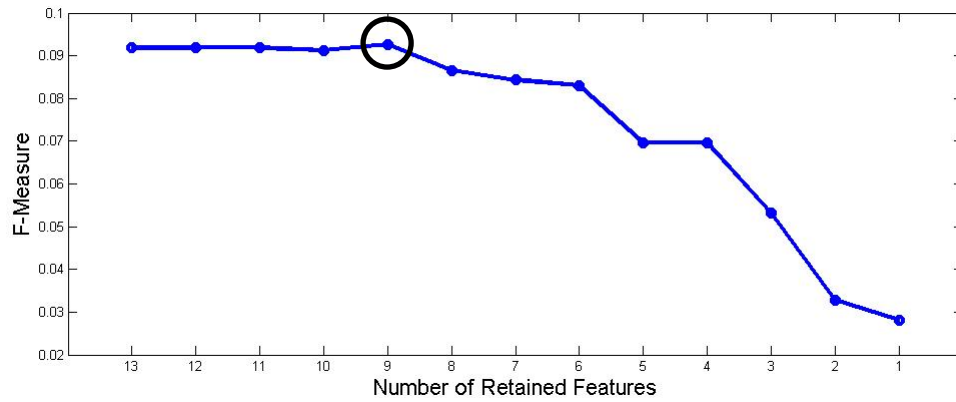
Cycle	Removed Feature	F-Measure	Modeled Features
0	-	0.0548	1-14
1	3,4,5,6,7,8,9,10,12	0.0548	1,2,11,13,14
2	11, 13, 14	0.0548	1,2
3	2	0.0548	1
4	1	-	-



**Figure 35: Plot of Retained features vs. F-Measure for EQ2-3 Binary Data Set**

**Table 32: Binary Feature Reduction EQ2-B4**

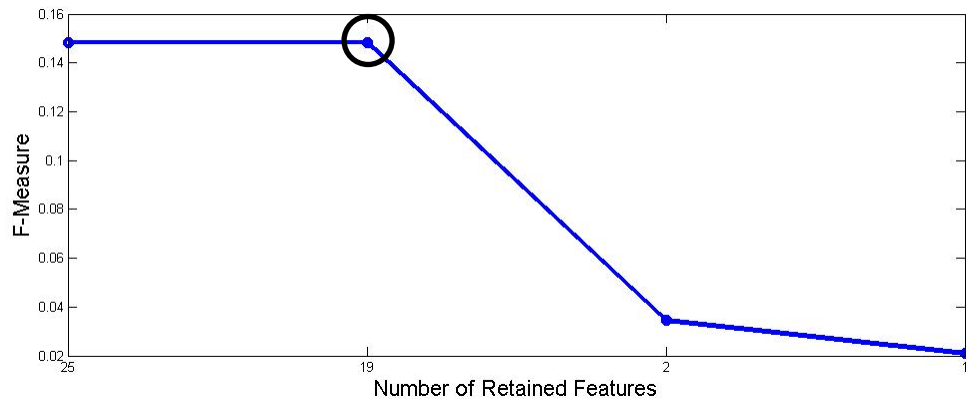
Cycle	Removed Feature	F-Measure	Modeled Features
0	-	0.0919	1-13
1	6	0.0918	1-5,7-13
2	13	0.0918	1-5,7-12
3	8	0.0913	1-5,7,9-12
4	9	0.0927	1-5,7,10-12
5	3	0.0866	1,2,4,5,7,10-12
6	4	0.0843	1,2,5,7,10-12
7	12	0.0831	1,2,5,7,10,11
8	11	0.0697	1,2,5,7,10
9	7	0.0697	1,2,5,10
10	2	0.0531	1,5,10
11	10	0.0328	1,5
12	5	0.0282	1
13	1	-	-



**Figure 36: Plot of Retained features vs. F-Measure for EQ2-4 Binary Data Set**

**Table 33: Binary Feature Reduction EQ2-B5**

Cycle	Removed Feature	F-Measure	Modeled Features
0	-	0.1485	1-25
1	1,4,7,9,21,25	0.1485	2,3,5,6,8,10-20,22-24
2	2,3,5,6,8,11,12, 13,14,15,16,17, 18,19,20,22,23	0.03445	10,24
3	24	0.0210	10
4	10	-	-



**Figure 37: Plot of Retained features vs. F-Measure for EQ2-5 Binary Data Set**

## Bibliography

- Ahmad, Muhammad Aurangzeb, Brian Keegan, Jaideep Srivastava, Dmitri Williams, and Noshir Contractor, "Mining for Gold Farmers: Automatic Detection of Deviant Players in MMOGs", *2009 International Conference on Computational Science and Engineering*, vol4, 2009, pp 340-345
- Balicer, Ran D., "Modeling Infectious Diseases Dissemination Through Online Role-Playing Games", *Epidemiology*, vol 18 numb 2 march 2007 pp260-261
- Bartle, Richard, "Hearts, Clubs, Diamonds, Spades: Players Who Suit MUDs," *The Journal of Virtual Environments*, Vol. 1, No. 1. (1996)
- Bauer, Kenneth W. 786 Class notes (year 2008)
- Bauer, Kenneth W., Stephen G. Alsing, and Kelly A. Green, "Feature Screening Using Signal-To-Noise Ratios", *Neurocomputing*, vol 31, 2000, pp29-44
- BBC News, "Billions Stolen in Online Robbery", (3 July 2009) 2 Sept 2009, <http://news.bbc.co.uk/2/hi/technology/8132547.stm>
- Blizzard Entertainment, "WORLD OF WARCRAFT® SUBSCRIBER BASE REACHES 12 MILLION WORLDWIDE", (7 October 2010) 18 October 2010, <http://us.blizzard.com/en-us/company/press/pressreleases.html?101007>
- Breast Cancer Wisconsin (Original) Data Set, UCI Machine learning repository, 15 July 1992 [http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Original\)](http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original))
- Chen, Kuan-Ta and Li-Wen Hong, "User Identification based on Game-Play Activity Patterns", *NetGames*, Sept 19-20, 2007
- Chen, Kuan-Ta, Jhih-Wei Jiang, Polly Huang, Hao-Hua Chu, Chin-Laung Lei, and Wen-Chin Chen, "Identifying MMORPG Bots: A Traffic Analysis Approach", *EURASIP Journal on Advances in Signal Processing*, vol 2009
- Duda, Richard O., Peter E. Hart, and David G. Stork, *Pattern Classification*, New York: Wiley, 2001, pp 580
- Flietstra, T.D., Bauer, K.W., and Kharoufeh, J.P., "Integrated Feature and Architecture Selection for Radial Basis Neural Networks", *International Journal of Smart Engineering System Design*, 2003, 5: 507-516
- Game Entertainment Europe, "Games", 4 Sept 2008, <http://www.ge-eu.com/games.html>

- GamerDNA, Bartle Test of Gamer Psychology, 26 Oct 2009,  
<http://www.gamerdna.com/quizzes/bartle-test-of-gamer-psychology>
- Keegan, Brian, Muhammad Aurangzeb Ahmed, Dmitri Williams, Jaideep Srivastava, and Noshir Contractor, "Dark Gold: Statistical Properties of Clandestine Networks in Massively Multiplayer Online Games", *IEEE International Conference on Social Computing*, August 2010, pp 201-208
- Kohavi, Ron, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection", *The International Joint Conference on Artificial Intelligence (IJCAI)*, 1995
- Kuncheva, Ludmila I., *Combining Pattern Classifiers: Methods and Algorithms*, John Wiley & Sons, INC. 2004
- Laurens, Peter, Richard F. Paige, Phillip J. Brooke, and Howard Chivers, "A Novel Approach to the Detection of Cheating in Multiplayer Online Games", *12<sup>th</sup> IEEE International Conference on Engineering Complex Computer Systems*, 2007
- Leuski, Anto and Victor Lavrenko, "Tracking Dragon-Hunters with Language Models", *ACM conference on Information and Knowledge Management*, Nov 5-11, 2006
- Lin, Holin and Chuen-Tsai Sun, "The 'White-eyed' Player Culture: Grief Play and Construction of Deviance in MMORPGs", *Proceedings of DiGRA 2005 Conference: Changing Views – Worlds in Play*
- Loeffelholz, Bernard, Earl Bednar, Kenneth W. Bauer, "Predicting NBA Games Using Neural Networks", *Journal of Quantitative Analysis in Sports*, vol 5: Iss 1, Article 7, 2009
- Matsumoto, Yoshitaka, and Ruck Thawonmas, "MMOG Player Classification Using Hidden Markov Models", *Lecture Notes in Computer Science: Entertainment Computing – ICEC 2004*, Springer Berlin/Heidelberg, 2004, pp429-434
- Ohsawa, Yukio, Nels E. Benson, and Masahiko Yachida, "KeyGraph: Automatic Indexing by Co-occurrence Graph based on Building Construction Metaphor," *Proceedings, IEEE International Forum on Research and Technology Advances in Digital Libraries*, 1998, pp12-18
- Pelleg, Dan and Andrew Moore, "X-means: Extending K-means with Efficient Estimation of the Number of Clusters", *Proceedings of the Seventeenth International Conference on Machine Learning*, June 29-July 2 2000, pp 727-734
- Ruck, Dennis W., Steven K. Rogers, and Matthew Kabrisky, "Feature Selection Using a Multilayer Perceptron", *Journal of Neural Network Computing*, vol 2, numb 2, 1990, pp 40-48
- Sargent, Daniel J., "Comparison of Artificial Neural Networks with Other Statistical Approaches", *Cancer*, vol 91, issue S8, 15 Apr 2001, pp1589-1697

- Specht, Donald F., “A General Regression Neural Network”, *IEEE Transactions on Neural Networks*, vol 2, no 6, Nov 1991, pp568-576
- Specht, Donald F., “Probabilistic Neural Networks”, *Neural Networks*, Vol 3, 1990, pp109-118
- Steiner, Peter, “On The Internet, Nobody Knows You’re A Dog”, *The New Yorker*, vol 69, no 20  
5 July 1993, pp 61
- Stepwisefit, MathWorks, 2010, <http://www.mathworks.com/help/toolbox/stats/stepwisefit.html>
- Thawonmas, Ruck and Ji-Young Ho, “Classification of Online Players Using Action Transition Probability and Kullback Leibler Entropy”, *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol 11, no 3, 2007a
- Thawonmas, Ruck and Katsuyoshi Hata, “Aggregation of Action Symbol Sub-sequences for Discovery of Online-game Player Characteristics Using KeyGraph”, *Lecture Notes in Computer Science: Entertainment Computing – ICEC 2005*, Springer Berlin/Heidelberg, 2005a, pp126-135
- Thawonmas, Ruck and Keita Iizuka, “Haar Wavelets for Online-Game Player Classification with Dynamic Time Warping”, *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol 12, no 2, 2007b
- Thawonmas, Ruck and Keita Iizuka, “Visualization of Online-Game Players Based on Their Action Behaviors”, *International Journal of Computer Games Technology*, Vol 2008, ID 906931, 2008
- Thawonmas, Ruck, and Yoshitaka Matsumoto, “Hidden Markov Models with Feature Mapping: An Application to MMOG Player Classification”, *Proc. 6th International Conference on Computer Games: Artificial Intelligence and Mobile Systems (CGAIMS'2005)*, Louisville, Kentucky, USA pp. 95-101, Jul. 2005b
- Thawonmas, Ruck, Ji-Young Ho, and Yoshitaka Matsumoto, “User Type Identification in Virtual Worlds”, *Springer Series on Agent Based Social Systems: Agent-Based Modeling Meets Gaming Simulation*, Springer Japan, 2005c, pp79-88
- Thawonmas, Ruck, Masayasu Hirano, and Masayoshi Kurashige, “Cellular Automata and Hilditch Thinning for Extraction of User Paths in Online Games”, *Proceedings of 5<sup>th</sup> ACM SIGCOMM Workshop on Network and System Support for Games*, 2006
- Thawonmas, Ruck, Masayoshi Kurashige, and Kuan-Ta Chen, “Detection of Landmarks for Clustering of Online-Game Players”, *The International Journal of Virtual Reality*, 2007c, 6(3) pp11-16
- The City Paper, “NY Man Pleads Guilty to Trafficking Credit Card Numbers”, (30 Apr 2008) 8 Sept 2009, <http://www.nashvillecitypaper.com/content/city-news/ny-man-pleads-guilty-trafficking-credit-card-numbers>

- Their, David, "World of Warcraft Shines Light on Terror Tactics", *Wired*, (20 Mar 2008) 12 Feb 2009, [http://www.wired.com/gaming/virtualworlds/news/2008/03/wow\\_terror](http://www.wired.com/gaming/virtualworlds/news/2008/03/wow_terror)
- Tveit, Amund, Oyvind Rein, Jorgen V. Iversen, Mihhail Matskin, "Scalable Agent-Based Simulation of Players in Massively Multiplayer Online Games", *Eighth Scandinavian conference on Artificial Intelligence*, 2003, pp153 – 162
- Van Rijsbergen, C. J., *Information Retrieval*, (1979) 18 October 2010, <http://www.dcs.gla.ac.uk/Keith/Preface.html>
- Wasserman, Philip D, *Neural Computing: theory and Practice*, New York: Van Nostrand Reinhold, 1989, pp7
- William H. Wolberg and O.L. Mangasarian: "Multisurface method pattern separation for medical diagnosis applied to breast cytology", *Proceedings of the National Academy of Sciences, U.S.A.*, Volume 87, December 1990, pp 9193-9196.
- Yan, Jianxin Jeff and Hyun-Jin Choi, "Security Issues in Online Games", *The Electronic Library: International Journal of the Application of Technology in Information Environments*, vol 20, no 2, 2002



<b>REPORT DOCUMENTATION PAGE</b>			<i>Form Approved</i> <i>OMB No. 074-0188</i>		
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p><b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b></p>					
<b>1. REPORT DATE (DD-MM-YYYY)</b> 03-08-2011		<b>2. REPORT TYPE</b> Ph.D. Dissertation		<b>3. DATES COVERED (From – To)</b> Sep 2007 – Aug 2011	
<b>4. TITLE AND SUBTITLE</b>  IDENTIFICATION AND CLASSIFICATION OF PLAYER TYPES IN MASSIVE MULTIPLAYER ONLINE GAMES USING AVATAR BEHAVIOR			<b>5a. CONTRACT NUMBER</b>		
			<b>5b. GRANT NUMBER</b>		
			<b>5c. PROGRAM ELEMENT NUMBER</b>		
<b>6. AUTHOR(S)</b>  Bednar, Earl M., Major, USAF			<b>5d. PROJECT NUMBER</b>		
			<b>5e. TASK NUMBER</b>		
			<b>5f. WORK UNIT NUMBER</b>		
<b>7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)</b> Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Street, Building 642 WPAFB OH 45433-7765			<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>  AFIT/DS/ENS/11S-01		
<b>9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b> HQ AF/A9I ATTN: Tim Booher 1570 Air Force Pentagon Washington DC 20330-1570			<b>10. SPONSOR/MONITOR'S ACRONYM(S)</b>  HQ AF/A9I		
			<b>11. SPONSOR/MONITOR'S REPORT NUMBER(S)</b>		
<b>12. DISTRIBUTION/AVAILABILITY STATEMENT</b> APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.					
<b>13. SUPPLEMENTARY NOTES</b>					
<b>14. ABSTRACT</b> The purpose of our research is to develop an improved methodology for classifying players (identifying deviant players such as terrorists) through multivariate analysis of data from avatar characteristics and behaviors in massive multiplayer online games (MMOGs). To build our classification models, we developed three significant enhancements to the standard Generalized Regression Neural Networks (GRNN) modeling method. The first enhancement is a feature selection technique based on GRNNs, allowing us to tailor our feature set to be best modeled by GRNNs. The second enhancement is a hybrid GRNN which allows each feature to be modeled by a GRNN tailored to its data type. The third enhancement is a spread estimation technique for large data sets that is faster than exhaustive searches, yet more accurate than a standard heuristic. We applied our new techniques to a set of data from the MMOG, <i>Everquest II</i> , to identify deviant players ('gold farmers'). The identification of gold farmers is similar to labeling terrorists in that the ratio of gold farmer to standard player is extremely small, and the in-game behaviors for a gold farmer have detectable differences from a standard player. Our results were promising given the difficulty of the classification process, primarily the extremely unbalanced data set with a small number of observations from the class of interest. As a screening tool our method identifies a significantly reduced set of avatars and associated players with a much improved probability of containing a number of players displaying deviant behaviors. With further efforts at improving computing efficiencies to allow inclusion of additional features and observations with our framework, we expect even better results.					
<b>15. SUBJECT TERMS</b> Modeling and Simulation, Multivariate Analysis, Neural Networks,					
<b>16. SECURITY CLASSIFICATION OF:</b>			<b>17. LIMITATION OF ABSTRACT</b>	<b>18. NUMBER OF PAGES</b>	<b>19a. NAME OF RESPONSIBLE PERSON</b>
<b>a. REPORT</b>	<b>b. ABSTRACT</b>	<b>c. THIS PAGE</b>			Dr. John O. Miller (ENS)
U	U	U	UU	136	<b>19b. TELEPHONE NUMBER (Include area code)</b> (937) 255-6565, ext 4326; e-mail: John.Miller@afit.edu