

ABSTRACT OF THESIS

POKERFACE: EMOTION BASED GAME-PLAY TECHNIQUES FOR COMPUTER POKER PLAYERS

Numerous algorithms/methods exist for creating computer poker players. This thesis compares and contrasts them. A set of poker agents for the system PokerFace are then introduced. A survey of the problem of facial expression recognition is included in the hopes it may be used to build a better computer poker player.

KEYWORDS: Poker, Neural Networks, Artificial Intelligence, Emotion Recognition, Facial Action Coding System

Lucas Cockerham

7/30/2004

POKERFACE: EMOTION BASED GAME-PLAY
TECHNIQUES FOR COMPUTER POKER PLAYERS

By

Lucas Cockerham

Dr. Judy Goldsmith

Dr. Grzegorz Wasilkowski

7/30/2004

RULES FOR THE USE OF THESES

Unpublished theses submitted for the Master's degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgments.

Extensive copying or publication of the thesis in whole or in part requires also the consent of the Dean of The Graduate School of the University of Kentucky.

THESIS

Lucas Cockerham

The Graduate School
University of Kentucky
2004

POKERFACE: EMOTION BASED GAME-PLAY
TECHNIQUES FOR COMPUTER POKER PLAYERS

THESIS

A thesis submitted in partial fulfillment of the
requirements for the degree of Masters of Science
in the College of Engineering
at the University of Kentucky

By

Lucas Cockerham
Lexington, Kentucky

Director: Dr. Judy Goldsmith, Associate Professor of Computer Science

Lexington, Kentucky

2004

©Lucas Cockerham, 2004

Acknowledgments

I wish to thank the following people:

- Dr. Judy Goldsmith, for all her help, expertise, and guidance;
- Caryl Goldsmith, for supplying her experience with poker in constructing the emotion tables;
- Emily Hendren, for supplying an article from the New Yorker with information on Paul Ekman's emotion research;
- Aydin Gurel, for allowing me to use his neural network Java package;
- all my friends who helped me test the poker game.

Table of Contents

Acknowledgments	iii
List of Tables	v
List of Files	vi
1 Introduction	1
2 Existing Work	3
2.1 Static Players	3
2.2 Game-State Modeling Players	5
2.3 Evolutionary/Learning Players	7
2.4 Comparisons of Existing Work	10
2.5 Additional Input for Evaluating Human Opponents	11
3 Methods	15
3.1 Overview	15
3.2 Poker Game	15
3.3 Neural Network	16
3.3.1 Neural Network Design	16
3.3.2 Neural Network Training	17
3.4 Emotion Inputs	18
3.5 Data Collection and Analysis	20
3.6 Configuration	21
4 Results	22
5 Future Work	25
6 Conclusion	28
Bibliography	30
Vita	32

List of Tables

3.1	Mappings of Emotions to Hands	19
4.1	Poker Player Performance, in terms of dollar amount winnings.	23
4.2	Computer game winnings per round.	24

List of Files

thesis.pdf

Chapter 1

Introduction

The game of poker is a classic, multi-player, betting card game. As with most well-known games, there have been many efforts to create a computer player that is highly competitive with human opponents. Many of these attempts are compared and contrasted here.

Poker is a game with many variations. The existing literature does not necessarily focus on one particular variation of poker. In general, the game of poker is played using a standard deck of cards. Each player has a certain number of cards at his or her disposal, called a hand. Play consists of a set of rounds, involving the acquisition of cards and contributions to the betting pool. The most assured way to win is to have the best hand in the game. Each set of cards has a particular ranking, and the highest ranking hand wins at the end of a game. Each round, players may call (meet a current bet), raise (increase the current bet), or fold (withdraw from the current round). However, poker is a rare game in that much of the strategy lies in guessing your opponents' hands and tricking your opponents into incorrectly guessing yours. This latter practice is known as bluffing, and plays an integral role in the game.

Poker is a game of imperfect information. A player does not know what cards his or her opponent has, nor does the player know what future cards will be dealt. The player also does not know what actions the opponents will take after the player's turn, and cannot discern this due to the number of possibilities. Therefore, it is computationally infeasible for a computer player to build a winning strategy by determining the entire game tree. Instead, the computer must make guesses based on what information it does have. According to Findler [8], a player is restricted to the following knowledge:

- the player's hand;

- the player's betting behavior in current and previous games;
- the opponent's betting behavior in current and previous games;
- the number of cards drawn by the player and the player's opponents.

Using this knowledge, the player can make decisions about what to do in hope of improving the chances of winning. Thus, the computer must process this limited information in order to develop a winning strategy. It should also be noted that certain information, such as opponent's facial expression, body language, tone of voice, etc., is not readily accessible by the computer.

In Chapter 2, a survey of existing work in the development of poker player artificial intelligence, as well as image processing for emotion analysis, are presented. In Chapter 3, the methods used for the original work in this thesis are presented. Chapter 4 details the results of these methods. Chapter 5 discusses future work related to the thesis, while Chapter 6 contains the conclusion.

Chapter 2

Existing Work

In this chapter, the existing strategies for developing poker-playing agents will be examined. The strategies have been roughly grouped into three classes: static players, game-modeling based players, and evolutionary/learning players. In the first section, players that use a static rules-based system for determining actions are studied. In the second section, methods for modeling the game state and/or opponents are considered. Finally, the third section consists of an overview of players that use evolutionary strategies.

2.1 Static Players

The simple solution to developing a poker player is to provide it with a set of guiding principles to follow and then releasing it to play. At least one poker text that teaches humans the intricacies of poker takes a similar approach. This book, *Win at Poker* [14], provides a simple set of guidelines for beginning players to use when deciding a course of action to take in particular games of poker. For example, the author states that, as a general rule, a player should fold if their initial hand is valued less than a hand with a pair of jacks in draw poker. This idea of a rules-based approach leads to some of the earliest work in developing computer poker agents, and is still used in more recent work, either as a piece of a solution or a indicator of how well another solution performs.

One of the first people to study poker applications in computer science was Findler [8]. Findler designed a number of systems that used heuristics to play the game of poker. Most of the players were based on rigid memory structures (decision trees, in this case) that model the game process but did not develop any sort of strategy. The decision tree indicated what move to make for any situation based on a set of rules. Findler called these decision tree-based players *static players*.

Findler also developed a player that used a “mathematically fair strategy.” In this case, the player calculated the difference between expected winnings and expected losses, based on probabilities of the success of a particular hand, and the previous bets made in that round by the player and the player’s opponents. Findler pointed out that this strategy “does not consider second-order effects of game situational variables.” This means the player did not keep track of the opponent’s playing style from hand to hand, how many players folded, or seating arrangements. It also means the player had no mechanism for handling bluffing. Thus, Findler indicated the player could consistently be beaten by another player that outbluffs it.

Another strategy was developed by Findler that uses five factors:

- The pot on the table (*TABLEPOT*);
- The number of people still alive in the game (*LIVE*);
- The number of raises (*RAISECOUNT*);
- The number of opponents with actions after the player (*FOLLOWERS*);
- The amount that must be put into the pot to stay in the game (*RAISE*).

The player then used this information to determine the probability of winning with the following formula:

$$RH = TABLEPOT / (LIVE * (RAISECOUNT + 1) * FOLLOWERS * RAISE) \quad (2.1)$$

Each of the strategies developed by Findler were rules-based players, and none enjoyed measurable success against human opposition. However, this does not mean they were not useful. Numerous authors since Findler have used similar static players in their research, often as a foil to their true poker agent [1, 10, 15]. Other authors have used a partially static system to solve small pieces of the entire game situation, as in [4, 6], where the initial betting round action was always determined using a rules system.

Billings, Davidson, et al.’s player, *Poki*, begins by using a rule-based system to determine its action during the pre-flop, the initial round of the poker variant Texas Hold’em. During the pre-flop, each player receives two private cards and a betting round takes place. Since there is a minimal amount of information available at this stage, the authors argued that a simple rule-based system is sufficient.

Additionally, Barone and While [1] provided a useful classification of poker players, based on their usual actions during a game of poker. Often static players are modelled to imitate these prototype poker players. The four players are:

- Loose Passive: Players who overvalue their hands, but rarely raise, being fearful of large pots;
- Tight Passive: Players who play few hands, usually with a high probability of winning, but rarely raise, being fearful of large pots;
- Loose Aggressive: Players who overvalue their hands, and who raise often to increase potential winnings;
- Tight Aggressive: Players who play few hands, usually with a high probability of winning, and who raise often to increase potential winnings.

2.2 Game-State Modeling Players

Another useful method for developing poker players is to try to model the game state as much as possible. As discussed earlier, poker is a game of vast, imperfect information that cannot be represented in a feasible amount of time or space by a computer. However, a few authors have tried to use methods to model as much of the game state as possible.

For poker, the game state consists of all of the information that a player (human or computer) uses to determine a course of action. Poker agent authors generally define what constitutes the game state in different ways depending on their application. In general terms, the game state consists of such things as the player's hand, the opponent's hand, the pot, the rules of the game, and all of the previous actions of the player and his or her opponents, both in the current game and previous games.

Shi and Littman [15] used a game theoretic approach to create a competent poker player through game modeling. In this case, the authors sought to maximize the utility of a particular problem using the assumption that one's opponent will seek to minimize the utility. The authors discussed the classical solution to such problems as building a tree of the different possible moves and then "minimaxing" through the tree to determine the best possible choice. Using a linear programming approach, the authors argued that solution times were polynomial in the size of the game tree. However, the game tree for poker is quite large. Too large, in fact, to make ordinary

solutions feasible. The authors then presented abstraction methods to decrease the size of the game tree and thus made it possible to solve large game tree instances.

The initial abstraction employed by the authors was to take all of the possible hands and place them into bins. A bin is a grouping of hands with similar values. In theory, there is not much difference in terms of the ranking of the hand between a pair of threes and a pair of fours, so these two hands can be placed together, therefore decreasing the size of the game tree.

While this works for variations of poker where one deals with complete hands, many popular versions of poker use partial hands, where a player must use his or her hand together with a set of known community cards. Here, the authors suggested ranking hands by taking the average of the strengths of all possible hands that may be created using that partial hand.

In order to experimentally validate their work, the authors developed a much smaller and simpler variation of Texas Hold'em they called Rhode Island Hold'em. The authors reduced some of the basic factors of the game (number of cards and betting rounds), thus making the game tree much smaller than in Texas Hold'em. The authors then generated a rules-based poker player to compete against their abstraction-based player. The authors developed a number of variations on the rules-based player, and used the original and the variations to compete against the abstraction-based player. Their results showed the abstraction-based player to be superior to the rules-based players.

Findler [8] also developed a few players that take the approach of modeling the state of the game, as opposed to his static players that were discussed earlier. Findler implemented four different players using different Bayesian networks to determine what actions to take. The first player kept very general information about its hand strength. The second player kept the same information, but for all opponents (at least as much as was made available). The third and fourth players kept track of a bit more information about the current game state. Essentially, each Bayesian player used one more piece of information than its predecessor. The Bayesian players were some of Findler's least successful.

Findler also described five more theoretical player types, though none were implemented at the time. The two most promising of these were the EP player and the Statistically Fair Player. The former builds discriminator trees with nodes that described the opponent's personalities and game situations. The latter is a variation of the previously defined mathematically fair player without the original's weakness to bluffing.

Davidson, Billings, et al. [4, 6] also used modeling of the game state through a set of probabilities that were used to determine actions and keep opponents guessing. Once the initial round

is finished, their player, *Poki*, used a three step system to determine its actions:

1. Determine the EHS (Effective Hand Strength)
2. Translate the EHS into a probability triple $\text{Pr}(\text{fold})$, $\text{Pr}(\text{call})$, $\text{Pr}(\text{raise})$ using game context, a set of betting rules, and formulas.
3. Generate a random number to help choose the action to add unpredictability.

Using this, *Poki* then attempted to determine the estimated value (EV) of a particular decision. It did this by playing the game out to completion based on different results. Each different version of the finished game was called a trial. Each trial was ran two times, once to consider the consequences of a check or call, and once for a bet or raise. In each trial, the opponent was given a random hand. The future actions for the opponent and *Poki* were determined using the probability triple above. A call EV and a raise EV were generated from these trials, and the action with the greatest value is taken. This simulation was not a full game tree search (that other authors, such as [15], have found infeasible). Instead, only part of the game tree was expanded; only the most likely nodes were considered. This is determined via the probability triple that was an indicator of the likely actions of both *Poki* and its opponents. The authors argued that using the above strategy helped them to develop excellent strategies without relying on expert knowledge.

2.3 Evolutionary/Learning Players

Another possible approach to building better poker players is to create computer poker agents that can learn about themselves, their opponents, or both. A number of poker players utilized these capabilities to some degree.

Findler [8] developed a number of players he described as Learning Players. The first was called the Adaptive Evaluator of Opponents. This player guessed its opponent's initial hand (roughly — hands were partitioned into groups of similar value, and the player guessed what group the hand belongs to), then developed a statistical library of actions taken by the opponent. As more actions were taken by the opponent, the statistical database was updated, and the player learned how the opponent plays. The next learning player was called the Adaptive Aspiration Player, which adjusted the player's style of play based on whether or not the player wished to play aggressively and take bigger risks for greater gains, or conservatively, to try to ensure minimal loss. The preferred style of play was determined by the situational variables, such as position and opponent's style, while

the player was active. Findler's last learning player is called Selling and Buying Player's Images. In this case, the player tried to model the probable actions of its opponents. Findler indicated that a measure of two values was enough to model a player's style: the player's level of consistency and level of cautiousness. In terms of quality of play against average opposition, the Adaptive Evaluator of Opponents and Adaptive Aspiration Player were the strongest players, in the above order.

Barone and While [1] used a different approach to building a computer poker player by utilizing genetic algorithms. They began with a calculation of hand strength by comparing the strength of the hand to every other possible combination of the opponents' hands. This heuristic alone did not suffice, however, as it did not indicate how to maximize winnings by raising the pot on a good hand. Thus, the authors then incorporated a set of functions that defined when the player should fold, raise, or call. A probability was generated for each of these actions, and this three-tuple of action probabilities is called a candidate.

In order to put in place the evolutionary structure, a hypercube populated by candidates was created. The hypercube was a three by four cube, with the width dimension representing when the player went in the game (early, middle, or late). The length dimension indicates how many times the ante had been raised. The authors suggested a limit of three raises per round, so the length dimension indicated how many raises had occurred (zero, one, two, or three). The authors indicated that the extra dimensions of the hypercube could be used to add more poker principles. Each element of the cube was populated by a set of possible candidates.

At each point in the game, a candidate population was consulted to determine which action to take. For each hand, the results were reported back to the population. Evolution then occurred after some fixed number of responses to the population. The best half of the candidate results were kept. This half is then mutated to replace the discarded half, and the new candidate function was recalculated. The authors did not specify how the mutation occurs.

In the first experiment, a randomly generated player played against two tables (groups of poker players). One table consisted of nine opponents of the "loose" type, and another nine opponents of the "tight" type. Over time, the evolving player became better and better against its opponents. However, its winnings were much greater against loose players as opposed to tight players. The second experiment categorized the type of play the evolving player used against a competent tight opponent as proposed by Findler. It showed that the player makes the more intelligent moves against a particular type of player once it is evolved. The third experiment showed that a player evolved at a particular table performs better at that table than a player evolved at another table. Lastly, their fourth experiment demonstrates the increase in proficiency of this new evolutionary

poker player as opposed to their randomly generated player.

Kendall and Willdig [10] developed another evolving poker player. Their evolving player used three pieces of information to determine whether to call, fold or raise:

- hand strength;
- number of players left at the table;
- amount of money in the pot.

The information that determined a particular hand's strength allowed the evolution to occur. The possible values for hand strength range from 1 to 10, with 10 meaning that the hand is the best possible. Initially, each hand was given a strength of 10. Such a high hand strength meant that the player would raise every time. Each time a player played a hand, the hand's strength value was modified by .1. If the player won, the hand strength increased by .1 (though not above 10), and vice versa if the player lost.

The evolving player was then pitted against rules-based versions of the four standard poker players discussed previously. Not surprisingly, the performance of the player was very poor initially. But as it learned that contesting every pot was not wise, and began to contest only on good hands, its performance increased. Like the previous experimental results for other authors, the evolving players learned much more quickly against loose players, and won more often as well.

Lastly, Davidson, Billings, et al. [4, 6] used neural networks to develop an opponent-modeling system that they combined with their aforementioned techniques to round out their poker player. This allowed them to adapt play to individual opponents dynamically. The authors wanted to obtain two pieces of information about the opponent from the model: the general strength of the opponent's hand, as well as their likely action in a given situation. To do this, the authors used a standard feed forward artificial neural network. The neural network consisted of a set of nineteen inputs that contain information about the game state. It produced three outputs, each of which corresponded to one of the values from the probability triple.

The authors first experimentally validated their work by having it compete against generic computer poker players, similar to the methods used by previous authors. However, the authors found that the results generated by these tests were incomplete, since they did not compare their poker player's skills to a human's. They found testing against human opponents was indispensable in determining the strength of *Poki*. The authors tested their players against numerous human opponents using an Internet Relay Chat channel. The authors created a general channel on which

any person could play. Once an individual accumulates enough points to show they were a proficient player, they were allowed to move on to another channel for better players. Thus, the authors were able to judge the level of competition for *Poki*. They found that against real human opponents, *Poki* was an intermediate poker player.

2.4 Comparisons of Existing Work

In this section, the previous work is compared and contrasted based upon the author's experimental results as given in their papers.

Findler [8], who first proposed the rules-based players, argued that an agent capable of learning was needed to play poker. His two most successful players both learned information from play against their opponents (see above, Adaptive Evaluator of Opponents and Adaptive Aspiration Player). Similarly, all of the players discussed here, with the exception of Shi and Littman's, used a learning approach to improve the play of their poker players. Two different approaches are used. Barone and While [1] and Kendall and Willdig [10] both used evolutionary programming to develop a learning poker player. Billings, Davidson, et al. used a neural network to do opponent modeling. Neither approach seemed to particularly stand out from the other in terms of results, although the results given by Billings, Davidson, et al. [4] have been tested extensively by human players, which lends extra credence to their results. Also, the work by Billings et al. diverged from the other two in that it uses a rules-based system to determine initial play and a partial decision tree to help make decisions about what actions to take.

Lastly, the final approach is to use search algorithms on a game tree, as done by Shi and Littman [15]. By the authors' own admission, this approach did not seem to be very effective due to the size of the tree, even with abstractions to decrease its size. The authors were able to get decent results using a simplified game of poker, but their approach would break down under the size of a game like Texas Hold'em. The fact they had to design their own simplified version of poker to obtain decent results for their method made it weak, at least in comparison to learning-based methods.

Of the three approaches given above, only Billings, Davidson, et al. [4] sought to integrate the approaches into one poker player. Findler [8] developed players that used learning processes, static rules, and classical searches, but not together at once. Billings, et al. used an expert system to determine simple pieces of game-play, such as the initial betting round. Further, they use a classical search to help evaluate the expected value of a particular hand. Finally, they used

learning techniques to develop an opponent modeling system based on neural networks. Thus, Billings, Davidson, et al. seem to have created the most robust system as compared to the other authors' work, and this seems to be supported by the results they have gotten from extensive human testing.

Further, Billings, Davidson, et al. [4] extensively tested their poker agent against human players, including players who demonstrated skill at playing poker. The other papers generally tested against computer generated rules-based players, even though the inferiority of such players had been shown by Findler and acknowledged by the other authors [1, 10, 15]. This leads one to believe that Billings, Davidson, et al. have built the most advanced player, as the authors claimed it is on an intermediate level against human opposition, and the other players were only tested against rules-based players.

2.5 Additional Input for Evaluating Human Opponents

As shown in [4, 8], bluffing (both the ability to bluff and the ability to recognize a bluff) is one of the most important aspects of developing a strong poker player. Since the ultimate goal is to build a poker player that can compete well against humans at a high level of skill, a possible research topic for computer poker agents is to better improve their ability to recognize an opponent's bluff.

When humans play each other at poker, they can (and probably do) use much of the same information the previously mentioned computer poker agents use to develop an idea of whether or not an opponent is bluffing. However, humans have a clear advantage over the computer agents in that they may see the body language and facial expressions of their human opponents. This is not simple input for a computer to receive, nor is it simple to determine how to process this information once received. This part of the paper will survey possible methods to allow computers to receive and process this input.

If one could develop a way for a computer poker agent to be able to receive information about the facial expressions and/or body language of a human opponent, new information could be gleaned that would allow the computer poker agent to perform better in many instances against the human player. One idea is to provide the computer poker agent with an additional input in the form of digital images of the players and use this information to help deduce the opponent's strategy. This section will therefore look at various technologies for the video capture of facial expressions, and ways these expressions could be processed to help the computer poker agent determine a human opponent's strategy.

The recognition of facial expression has been an actively studied topic in psychology for some time now. Paul Ekman [9] developed a system called FACS (Facial Action Encoding System) that encodes the muscles of the face to indicate what emotion is currently being shown. FACS uses Action Units (AUs) that refer to positions of the specific muscles in the face. AUs are combined to define facial expressions. Therefore, these AUs can be used to help a computer agent identify facial expressions. This survey will concentrate on the use of neural networks to identify expressions via AUs.

A paper by Donato et al. [7] reviews a number of different techniques for classifying of facial images. Only those techniques specifically related to facial expression recognition and that performed the best in the comparison will be considered here. Interested readers should see [7] for details on other methodologies. Here, only Gabor wavelet decompositions and independent component analysis are considered.

Gabor filters are filters that can remove variation in images, such as from lighting and contrast. A Gabor filter is a two-dimensional sine wave with a Gaussian envelope. By looking at the output of differing families of Gabor filters applied to human faces, Zhang [16] has been successful at determining facial expressions.

Independent component analysis [7] is a generalization of Principal component analysis that seeks to find high-order dependencies in an image set. Examples of high-order dependencies include “nonlinear relationships among pixel grayvalues such as edges, in which there is phase alignment across multiple spatial scales, and elements of shape and curvature. In a task such as facial expression analysis, much of the relevant information may be contained in the high-order relationships among the image pixels.”

Dailey, et al. [5] used both Gabor wavelet decompositions and principal component analysis to pre-process facial images obtained from the FACS database and actors. The pre-processed information was then used as an input to a six-unit neural network. Using the preprocessed information, the neural network could identify six prototypical emotional expressions: happiness, sadness, fear, anger, surprise, and disgust. To train the network, the authors used images of fourteen actors. They trained the network on thirteen of the actors, and kept one actor out as a holdout set. After each epoch of training, the network was tested on the holdout set. After six epochs passed with no improvement in performance on the holdout set, the network was reverted to the first previous network where an improvement had occurred. The authors did this to prevent over-training. The authors trained 182 networks on different partitions of the actors and achieved 85.9 percent accuracy in identifying the six emotions previously specified.

Pantic, et al. [13] developed a more fine-grained system that could recognize 30 action units, their combinations, and their intensities from video images. The system that they developed had four parts. The first, the Facial Data Generator, acquired images and performed pre-processing on them to detect certain features (such as the eyes). The second stage, the Facial Data Evaluator, used this data to pick out the best of the images from the previous stage. The evaluator then determined a set of face-model points that indicate the positioning of important features about the face to be analyzed. These points were then compared with the points from the neutral face image for consistency. Images that were highly inconsistent were thrown out; those that were consistent were given a value referred to as a certainty factor that generalized how well the points made sense in context of the neutral face image. The certainty factors and face-model points for those images with the highest certainty factors were then passed to the next stage. The data analyzer is an expert system that looks at the neutral images of the person, as well as the images in which affect is shown. The data analyzer compared the differences between the face-model points of two images and then determined what AU was being shown by that difference. Lastly, a post-processor called HERCULES was used. If an image showed one expressionless feature (such as the eyes) coupled with another feature that showed an expression (such as smiling with the mouth), the image was passed into HERCULES. HERCULES used an expert system to determine if an emotion was being shown despite its absence in one of the areas of the face in which the analyzer looks for data. For instance, if the mouth was smiling but the eyes did not indicate happiness. The system agreed with expert human FACS encoders 86% of the time.

Tian, et al. [11] developed a system to recognize individual AUs from facial expressions. Their system addressed two problems: facial feature extraction and facial expression classification. To do this, the authors first extracted individual facial features from initial still images. Two types of features, permanent and transient, were extracted. Permanent features are those expressive features of the face that are always present, such as the nose, eyes, and mouth. Transient features are things like furrows and brows that are not always present. To extract these features and determine changes in them, each feature was mapped to sets of points. As the points move, the facial feature being modeled could be extracted.

Once the features had been extracted, they were separated into upper facial features and lower facial features. This separation was useful because upper facial features have little interaction with lower facial features. Features of each were represented by some number of parameters. These parameters were then given to a three layer neural network with six hidden units. This neural network then had outputs for each of the AUs in that section of the face. When multiple AUs were

present, more than one output node was excited. The output node with the highest value was used to determine the correct AU. Using a neural network with twelve hidden units, the authors were also able to recognize combinations of AUs. Here, the output nodes remained the same, but any output node with a value above a certain threshold was considered excited, and thus combinations of AUs could be found. The network achieved 96.4 and 96.7 percent recognition for upper and lower face AUs, respectively.

In [12, 2, 3], Littlewort, Bartlett, et al. described a system they developed for the automatic recognition of spontaneous facial emotions. Their system used multiple cameras to capture facial images from a variety of human subjects. These images were then mapped into a single plane (so the faces were always oriented straight ahead regardless of how the subject had their head tilted). Information about the expressions presented on the face were estimated using Gabor filters combined with Adaboost and/or Gentleboost. These expressions were then classified into one of seven emotions: neutral, anger, disgust, fear, joy, sadness, and surprise. In order to recognize these emotions once they have been extracted from the images, Littlewort et al. trained the filters on a set of digital images of emotions that had been hand coded by trained human emotion encoders. Littlewort et. al. were able to achieve a 93 percent success rate in identifying emotions on well-lit, posed training data, and 80 percent on images collected from the world wide web. The researchers have begun using their research in a number of applications, including an emotional mirror that reflects a user's emotions back at them, and evaluating psychiatric patients using data on their emotions collected via the software.

Chapter 3

Methods

3.1 Overview

In this section, an overview of the original research and work performed for this thesis is provided. The goal of the research is to develop a poker game that makes use of some type of emotion data to help build a better computer poker player. To this end, a simple poker game is developed that uses one human opponent against several computer opponents. This poker game makes use of an artificial neural network to determine its actions. In some instances, this neural network is given emotion inputs in addition to its other data inputs. A number of human players then play this game, both with and without the emotion inputs, and the computer poker players' performance is compared.

The rest of this section proceeds as follows. First, the poker game is discussed. Second, the neural network that forms the basis for the computer poker player's move selection is described. Third, the type of emotion inputs that are given to the neural network will be covered. Fourth, a discussion of the way the data for the research is collected and analyzed is given.

3.2 Poker Game

For this project, a simple, interactive poker game is developed. The game is straight poker, with no variant rules. The human player can see his cards, and is given a listing of each action taken by his four computer opponents. The player begins with \$100 of "play money" that he or she can use for betting. The computer players have no limit on their money so that the human player may continue to play indefinitely, up until he or she runs out of money.

Play begins with each player being dealt five cards and placing an ante. Each player can only see his or her cards. The ante is automatically taken from each player, and is set at \$1 for simplicity. Once this happens, each player takes one of three actions: raise, call or fold. Each computer player always raises by a set amount (\$2); the human player can raise as much money as he or she desires. The difference is due to the difficulty in training the computer player to raise by differing amounts of money, which may be fixed in future versions of the program. The betting continues until all players but one fold, or all of the player's bets are equal.

Once the stopping conditions occur, one of two things happen, depending on how the game ended. If only one player did not fold, that player automatically wins the pot, or all of the money that has been bet. If more than one player remains, a showdown occurs. In a showdown, all players who have not folded their cards reveal their cards. The player with the highest poker hand (see [14] for a list of poker hands and their precedence) wins the pot.

In each round, one player is considered the dealer. This is important, as the player to the left of the dealer bets first, and the dealer bets last. Strategically, being the dealer is an advantage in poker. In the first game, the human player is the dealer. As play progresses, each computer player becomes the dealer, then back to the human player, and so on until play ends. Each of these rounds follows the steps outlined above.

The game presented in this thesis is very simple. It uses the simplest possible poker rules, and makes other concessions to simplify game-play. This was done for several reasons. First, it allows a poker player to be developed in a reasonable amount of time. Second, since the game is simple, it is straightforward to analyze. Third, there is no obvious ultimate version of poker. By creating a simple game, the research can be later expanded to other versions of poker as the results dictate.

3.3 Neural Network

The poker agent used by the computer players in this thesis relies mainly upon an artificial neural network to make its decisions. Neural networks are well-known and commonly used algorithms for machine learning. Many of the poker researchers covered in Chapter 1 used neural networks to develop their players. For this reason, they are relied upon heavily in this research.

3.3.1 Neural Network Design

The decisions the computer poker players make are based on a neural network. The network is a standard feed forward neural network with eight or nine input units (depending on the presence

or absence of an input for an emotion), two hidden layers with ten units each, and one output unit. What follows is a description of the neural network in terms of its inputs and outputs.

The neural network takes the following inputs:

Position The position of the player at the table. The first player is in position 1, and so on. The higher the position, the more advantage the player has due to knowledge of the actions that have already been taken by other players.

Hand Strength A numeric value that indicates the relative strength of the player's hand. The value of the hand is determined by taking all of the possible poker hand types and assigning them consecutive values, with the lowest valued hand starting at 1. This metric only takes into consideration the hand ranking; thus two hands that are both full houses have the same hand strength even if the cards are different.

p1Action The action taken by the first computer player. The possible computer player actions are fold, raise, and check. Each action is mapped to a numeric value.

p2Action The action taken by the second computer player.

p3Action The action taken by the third computer player.

p4Action The action taken by the fourth computer player.

Pot The amount of money a player would win by winning the hand.

Ante The amount of money the player must ante (place into the pot) to remain in the hand. Note that this does not include any money the player previously placed in the pot.

The neural network generates three outputs, one each for the possible actions (raise, check, fold). The action has the highest value in the output layer is the one that is taken. In the event of a tie, precedence is given in this order: check, fold, raise.

3.3.2 Neural Network Training

The neural network is trained by the author via play of the poker game. The actions of the computer players are chosen randomly amongst the three possible options. The choices made by the author are meant to represent sound decisions by a poker player in the real world, and did not take into consideration the random nature of the poker player (which would have vastly changed

the author’s playing style against the computer). Thus the poker player’s training is skewed to the author’s understanding of the game and sound strategies described in [14]. It is also limited by the amount of time the author spent training it. Due to the vast number of situations in poker, it is impossible to train the player for a large subset of all possibilities.

The data for the given inputs above is gathered and recorded each time a player takes an action within the game. Thus, if the betting round passes through the human player three times, three training entries are recorded. This data is written to a file that is then used to train the neural network. The network is trained using mini-batch training using the data collected from the author’s play sessions. Mini-batch training is performed by first initializing the weights of the network and then processing a subset (but not all) of the training cases, and then updating the weights.

Approximately 450 training entries (consisting of an action taken by the author during a round of betting) are used for training the neural network. The error ratio of the networks that are used in the game that the actual research testing is conducted on is fairly high (30%). This is probably due to the limited number of training cases in conjunction with the large number of possible inputs that can occur in the actual game. The author was able to achieve lower error ratios, but not on a consistent basis across the neural networks with the differing inputs (see below). Therefore, the network with the higher error ratio that can be achieved with both networks is used.

3.4 Emotion Inputs

Being able to characterize emotions exhibited by the human player could prove useful in determining a course of action for the computer players to take. However, there are several obstacles to this. As shown in the survey of existing work, while many groups have had success in capturing emotions from video or images, the technology is still new, expensive, and difficult. This combination of obstacles is insurmountable at present, so a place holder was developed to randomly generate emotions one might expect from a human playing the poker game. The idea is that while the actual emotions cannot be directly captured, emotions can be generated that fit a given situation, and the neural net can use them to its advantage.

To do this, a biased random number generator is created to assign random emotions based on the hand. That is, a hand that ranks as two pair would generate a certain set of possible emotions, and a straight flush would generate another set of possible emotions. Refer to table 3.1 to see the emotions generated by each hand, and their bias within the random number generator.

Table 3.1: Mappings of Emotions to Hands

Hand	anger	disgust	enjoyment	fear	sadness	surprise
No Pair	30%	30%	0%	0%	30%	10%
One Pair	20%	10%	10%	30%	20%	10%
Two Pair	0%	0%	50%	20%	0%	30%
Three of a Kind	0%	0%	60%	10%	0%	30%
Four of a Kind	0%	0%	60%	10%	0%	30%
Straight	0%	0%	60%	10%	0%	30%
Flush	0%	0%	60%	10%	0%	30%
Full House	0%	0%	60%	0%	0%	40%
Full House	0%	0%	40%	0%	0%	60%

The values assigned within the table above were reached by considering the author’s typical emotional responses to certain poker hands, particularly during the training phase. The emotions indicated here are taken from the set of seven emotion dimensions that are recognized by the software developed in [3]. Each hand ranking is assigned emotions that may occur regularly when that hand was dealt to a human player. Probability was then assigned to each emotion based on an approximation of how likely it occurs for that hand. The emotions are assigned atomically; no consideration is given to what emotions would occur if certain hands were dealt consecutively. For example, if a player is dealt a hand with only a high card several times in a row, the likelihood of the player being sad or disgusted is much higher than if the player only received one such hand.

For each hand, the computer player is assigned an emotion via the biased emotion generator. This emotion is then given to the neural network as an extra input. The neural network is trained by randomly generating emotions and adding them to the existing training data based on the hands. Thus, the emotion can then be used as an input to the computer player’s neural network, and help the computer poker player determine its action. It is important to note that while the computer looks at the human player’s card to determine what emotion to assign, the computer players do not have access to this information when making their decisions.

When the neural network was trained with the additional random emotion information, 4500 training cases were used. This is a much larger number than the original training set. This is because it was deemed desirable to ensure that a variety of emotions were represented in the training data. If only one emotion was generated for each training case, there was a danger the neural network would be trained to believe that was the only possible emotion. To circumvent this, each training case from the set used to train the initial neural network was randomly assigned an emotion based on table 3.1 ten times. This meant that each case in the original training set was

used to create ten new training cases for the new network. This was done to make sure that a number of emotions should appear for each hand ranking in the training data.

3.5 Data Collection and Analysis

The research methodology used for this thesis is described in this section. It covers the data that is collected, how it is collected, and how it is interpreted. For this research, the measure of a poker player is how much money the player wins (or loses). Thus the main metric for performance is the positive or negative difference between the amount of money a player ends with and the amount of money the player begins with. Each human player begins with \$100, an amount that is large enough to allow human play to continue for some time based on the minimum bet. Once all of that money is gone, the game is over for the human player. The poker players have limitless resources; no matter how much money they lose, they may still play. However, this is not taken into account in the computer player's strategy, and therefore has negligible impact on the game. Regardless, the worst score a human player can generate is 0 (that is, the player loses all of his or her money); a computer player has no limit on how much money it can lose or win, except on how long the human player continues to play. To ease analysis, each computer player is assumed to start with \$100. Therefore, at the end of the game, the computer players could end up with a negative amount of money.

Each game consists of five players: four computer players and one human player. Five to ten individuals are recruited to play the game a total of 4 times. Each session lasts for fifty hands or until the human player loses all of his/her money. Once the game ends, the program tallies the amount of money won or lost by each player (both human and computer).

Each individual plays half of his or her games against a poker player that does not use any additional emotional input, and half against a poker player that uses the emotional inputs outlined above. The subject does not know the type of game he or she is playing against. The games are always played in the same order; the first game does not use emotion, the second does, the third does not, and the fourth does. Once play is complete, the average winnings of the computer poker players with the non-emotion-based input is compared to the average winnings of the emotion-based computer poker players. These is also compared against the winnings of the human player.

The human participants are not coached by the author except to provide the rules to facilitate game-play. The author does not discuss strategy with the participants, nor are they permitted to discuss strategy with one another. Participants are not treated differently based on ability or

experience. Prior to playing, the participants are asked to rate themselves as novice, intermediate, or expert poker players.

3.6 Configuration

Some testing and all development for this thesis was performed on a Athlon XP based Desktop PC with 512 MB of RAM, running Windows XP. Development was done in Borland JBuilder 9 Personal. Additional testing was performed using a Dell laptop with a Pentium 3 processor and 512 MB of RAM, also running Windows XP. Java Runtime 2 was used to interpret the program when it was ran from the laptop.

Chapter 4

Results

As shown previously, it is difficult to create a poker player that performs well against human competition. The results of this research do nothing to refute this supposition. The poker players developed here, both with and without pseudo-emotion input, performed poorly against human players. This section evaluates their performance, and discusses some of the flaws that human players exposed. It also compares the different types of neural nets that were used.

Table 4.1 succinctly summarizes the results. The column labeled “Human” shows the average ending money of all the human players who played the game with each input. This was done by taking how much money each player had at the end of a round of fifty games. All of these values were then averaged to give the average winnings of a human player after fifty hands. The computer player’s winnings were calculated similarly, save that the average shown is the average of each of four computer poker player’s winnings combined, with their averages calculated just like the human’s (including negative values if the player lost more than \$100). The average of all four players was used, since they all used the same decision making mechanism in determining their actions. The column with this information is labeled “Computer.”

Clearly, the human players were superior to the computer players. Interestingly, the humans performed significantly better against the computer players that used a neural network with no additional pseudo-emotion information. The humans nearly doubled their winnings against such players. Additionally, the computer was generally able to at least break even when no emotion input was provided. However, the computer players’ losses were significant with the emotion input, losing much more money than the human player actually started out with. In effect, if all of the players had set down with \$100, the computer players would almost all run out of money while the human player would have significantly increased his or her money.

Table 4.1: Poker Player Performance, in terms of dollar amount winnings.

Player	No Emotion Input	Pseudo-Emotion Input
Human	+\$478	+\$915
Computer	+\$5	-\$104

Looking at the demographics of the players sheds little light on the subject. All considered themselves novice players, but had played poker before. So despite being novices, they were able to intuit good methods to consistently defeat the computer poker player. The only other relevant pieces of demographic information is that three of the five are Information Technology professionals with varied and extensive experience with computer programs. Three of the five (though not the same three) were also very experienced with video game AI due to extensive time spent playing video games. All five have at least bachelor’s degrees.

In total, five players played 200 hands each (in rounds of 50), for a total of 1000 games. That means the original neural network and the network using pseudo-emotion inputs each played 500 hands with four players, or 2000 hands all together. Thus the sample space for the testing was sufficiently large.

Looking at the results, it is apparent that the computer poker player is inferior to even novice human poker players. Besides the aforementioned pratfalls to creating a good artificial poker player, every single human player found a single, large weakness in the author’s poker agent. Some exploited it better than others, but all realized the advantage of placing very large raises against the computer when they had a very good hand. Most of the time, at least one computer player would meet the large raise, and generally lose. Conversely, the computer poker player was timid about raising (most players claimed they saw raises rarely), as well as being limited to only raising \$2 at a time. Thus, the computer would have needed to raise several times to meet some of the raises the human players made.

There are a number of reasons this flaw exists in the game. First, due to the amount of training that occurred, good hands were rare, so the computer players were not well versed in how to react to them. Second, the computer players were trained to react to raises in general, not how much was raised. Thus, a \$2 raise was the same to them as a \$500 raise, even though they are very different. Third, the computer players’ money was endless, so they continued to meet large raises with no regard for how much money they were losing.

Past the computer player’s generally poor performance lies the information that is most relevant to this research: what effect the pseudo-emotion information had on game play. As the table

Table 4.2: Computer game winnings per round.

round	Pseudo-Emotion Input?	Winnings
1	No	+\$72
2	Yes	-\$120
3	No	-\$62
4	Yes	-\$216

shows, the poker players with pseudo-emotion information lost more money than those without the information. The difference is significant, as it is more than \$100. Thus, it seems that the pseudo-emotion information does not make for a better player. In fact, it makes for a worse player.

However, there are at least two caveats that indicate this data may be misleading. First, the order the rounds of fifty hands were played in may be significant. Rounds using the non-emotion based computer players were played first and third; emotion based computer players participated in the second and fourth rounds. As the human subjects played the games, they learned how the computer players played and adjusted their playing style. It is difficult to determine if this truly had an impact based on the way the tests were set up, but table 4.2 shows that the computer players posted their best performance in the first round against humans, and each subsequent round they did not perform as well, though the worst performances were still posted when the pseudo-emotion input was used.

It is also possible that the additional pseudo-emotion input served as an extra, obfuscating input to the neural network. It is difficult to prove this in any way, but it is reasonable to think that an extra input that was added on to the existing training data would confuse the training. In other words, as the network is trained, the extra input may lead the network to believe a different output is correct, rather than the one for which it is trained. The argument particularly make sense in light of the fact that the player that used the extra pseudo-emotion information was clearly the worst of the two players.

Chapter 5

Future Work

The possibilities for future work or expansion of the research material presented here are seemingly limitless. This discussion of future work will look at two possible directions where expansion is possible. First, the ways the poker player itself can be improved (with no relation to emotion-based input) will be examined. Second, ways in which the poker player's gathering of emotion-based inputs can be improved will be discussed.

The poker player used in this research, as explained and justified previously, is quite simple. The most obvious improvements that can be made to the poker player involve changing the game itself or tweaking the way the poker player makes its decisions. However, any modification of the former automatically means a modification of the latter, as it must be adjusted to a new type of play.

The game of poker currently used is the simplest common game of poker. In general, it requires less strategy than more common poker games, and depends more on luck, particularly in a computer-based environment where the effectiveness of bluffing is minimized. Therefore it would likely be more interesting to change the game to variant such as five-card stud or five-card draw. Other variations, such as the very popular Texas Hold'em, are possibilities. Currently, the game was designed to accommodate the game of five-card draw. This would be the most likely route to expand the game, and would require adding the ability to remove one to three cards from any player's hand for new cards, and also introduces a second betting round.

Besides the extra betting round, this adds the complication of choosing cards to exchange. Previous researchers have used rules-based systems [4] to determine the cards to exchange; this could also be done by an additional neural network, again trained on inputs from human users. Regardless, such a modification would require some sort of different AI component be added to

determine the cards to exchange.

While there are many other ways to expand the poker game itself, the most interesting addition is the collection and use of actual emotion data gathered from human players. Emotions, and the ability to read them, are an integral part of playing poker. They allow players to determine whether or not a player is bluffing, for example, which can lead to a significant edge when playing the game. The player presented here uses randomized emotion inputs, which is a novel but naive way of representing emotions that could be displayed by the human player. The more interesting approach would be to try to gather and analyze actual emotion data from the player.

Advances in image processing and expression recognition, as shown in the survey section, make such an addition to the presented system feasible, with the appropriate equipment and expertise. It has been shown that cameras can capture images of human faces and determine the expressions that are being displayed in terms of the six types of expressions/emotions that are currently used by the biased random emotion generator. Thus, if the technology and image-capture software were readily available to gather the expression information, this could replace the data generated by the biased random emotion generator. Then the computer poker player's would have a true and most likely correct insight into an emotion displayed by the human opponent while they play the game.

Given the generally poor quality of all computer poker players, it is possible that the additional input could improve their general performance. Certainly, one of the most important parts of poker is reading an opponents' reaction, and using image-captured emotions would allow the computer player to evaluate its opponents emotions. Most likely, the computer player would be better at reading emotions than the average human poker player!

However, without implementing this, it is difficult to say if the extra information would prove useful. It seems likely that some additional processing would be required to help the computer player identify emotions that are often displayed when the opponent has certain hands. This would mean that the computer player could need history information for its opponents, or it could generalize about the normal response for certain situations from most human players (though the variability is unknown at this point) and use that to further help it guess the strength of the human opponent's hand and determine its best course of action.

Additionally, based on the poor performance of the poker player in general, it would seem likely that the AI for the base poker player would need to be improved for the more precise emotion information to actually improve its performance. There are a number of ways to possibly do this; approaches could be borrowed from those suggested in the survey chapter. Further, the approach to training could be modified and refined. There are several possibilities: more training could be

performed, a more adept poker player could be used to perform the training, and, of course, the actual emotion information could be used in the training.

Regardless, the possibility of capturing image information directly from a human player offers many avenues of research. Simply finding ways to gather emotion information and analyze it for its own sake has been a very active topic of research in literature over the last few years. Then actually applying this information to a computer poker player and analyzing its impact on performance would be an additional path of possibly rewarding study.

Chapter 6

Conclusion

A number of interesting conclusions can be arrived at by studying the work in this thesis. Based on the survey and the research performed here, the most difficult part of making a good emotion-based poker player may be to make a good poker player, with no emotion information. The player presented here performed poorly against human competition. Based on its performance, it is likely that major changes would be needed to make this poker player competitive with the average human. Probably, a single approach (such as modeling with a neural network) is not sufficient to create a competitive poker player, even excepting any other circumstances (such as lack of comprehensive training) that diminished the player's performance. That being the case, the focus for future research should first head towards creating a competitive poker player, and then adding emotion-based information.

That being said, it was interesting to note the apparent negative effect that the pseudo-emotion information had on the poker player's performance. The hope was that at worst it would have no impact, and at best cause a nominal improvement in performance. Instead, the poker player was noticeably worse when it used the extra pseudo-emotion information. However, it is difficult to pinpoint exactly why it was worse. Human improvement and learning could have accounted for part of it, as could the addition of an input that obfuscated the real goal of the training. These are not conclusive results that emotion inputs would be detrimental to a good poker player; instead, it shows that much care and consideration must be taken in trying to integrate this new dimension of information.

In the end, this research shows that competitive computer poker players are ripe for advancement. It is difficult to create a very good player, and creating a player that is competitive against another human will require thinking outside the obvious realms of solutions. This lends credence

to the possibility that emotion information will be a possible path to improving computer poker players. Using emotion information will require sufficient advancements in the fields of image processing to easily and cheaply capture emotion information in such away that it can be provided in a straight-forward manner to a skilled computer poker player. It is possible such information may not help, and may even hinder, such a player, as occurred in this research. However, this research does not closely approximate the interaction between a moderately good poker player and a good emotion detection system. It only shows such interaction is possible, and will likely lead to interesting, if not necessarily fruitful, results.

Bibliography

- [1] Luigi Barone and Lyndon While. An adaptive learning model for simplified poker using evolutionary algorithms. In *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 153–160, 1999.
- [2] M. S. Bartlett, G. Littlewort, B. Braathen, T. J. Sejnowski, and J. R. Movellan. A prototype for automatic recognition of spontaneous facial actions. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*. MIT Press.
- [3] Marian Stewart Bartlett, Javier R. Movellan, Gwen Littlewort, Bjorn Braathen, Mark G. Frank, and Terrence J. Sejnowski. Towards automatic recognition of spontaneous facial actions. In Paul Ekman, editor, *What the Face Reveals, 2nd Edition*. Oxford University Press, 2003.
- [4] Darse Billings, Aaron Davidson, Jonathan Schaeffer, and Duane Szafron. The challenge of poker. *Artificial Intelligence*, 134(1-2):201–240, 2002.
- [5] M. Dailey, G. Cottrell, and R. Adolphs. A six-unit network is all you need to discover happiness. In *Twentysecond Annual Conference of the Cognitive Science Society*, pages 101–106, 2000.
- [6] Aaron Davidson, Darse Billings, Jonathan Schaeffer, and Duane Szafron. Improved opponent modeling in poker. In *International Conference on Artificial Intelligence*, pages 1467–1473, 2000.
- [7] Gianluca Donato, Marian Stewart Bartlett, Joseph C. Hager, Paul Ekman, and Terrence J. Sejnowski. Classifying facial actions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):974–989, 1999.
- [8] Nicholas V. Findler. Studies in machine cognition using the game of poker. *Communications of the ACM*, 20(4):230–245, 1977.

- [9] Malcolm Gladwell. The naked face. *The New Yorker*, pages 38–49, August 2002.
- [10] Graham Kendall and Mark Willdig. An investigation of an adaptive poker player. In *Australian Joint Conference on Artificial Intelligence*, pages 189–200, 2001.
- [11] Ying li Tian, Takeo Kanade, and Jeffrey F. Cohn. Recognizing action units for facial expression analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):97–115, 2001.
- [12] Gwen Littlewort, Marian S. Bartlett, Ian Fasel, Joel Chenu, Takayuki Kanda, Hiroshi Ishiguro, and Javier R. Movellan. Towards social robots: Automatic evaluation of human-robot interaction by face detection and expression classification. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.
- [13] Maja Pantic and Leon J. M. Rothkrantz. An expert system for recognition of facial actions and their intensity. In *AAAI/IAAI*, pages 1026–1033, 2000.
- [14] Jeff Rubens. *Win At Poker*. Dover, 1968.
- [15] Jiefu Shi and Michael L. Littman. Abstraction methods for game theoretic poker. In *Computers and Games 2000*, pages 333–345, 2000.
- [16] Zhengyou Zhang. Feature-based facial expression recognition: sensitivity analysis and experiments with a multi-layer perceptron. *International Journal of Pattern Recognition and Artificial Intelligence*, 13(6):893–911, 1999.

Vita

Date and Place of Birth: Lexington, Kentucky, January 29, 1979

Education: Bachelor of Science in Computer Science,
Eastern Kentucky University, 2001

Professional Positions: Graduate Teaching Assistant
University of Kentucky
Lexington, Kentucky, 2001–2003

Senior Software Quality Analyst
ACS
Lexington, Kentucky, 2003–present

Honors: Honors Program Graduate
Eastern Kentucky University, 2001