MACHINE LEARNING IN EUCHRE:
A COMPARISON OF TECHNIQUES

A Thesis

Submitted

in Partial Fulfillment

of the Requirements for the Degree

Master of Science

Michael J. Holmes

University of Northern Iowa

May 2001

This Study by: Michael J. Holmes
Entitled: Machine Learning in Euchre: A Comparison of Techniques

has been approved as meeting the thesis requirement for the

Degree of Master of Science.

Date	Dr. Eugene Wallingford, Chair, Thesis Committee
Date	Dr. Mark Fienup, Thesis Committee Member
Date	Dr. Jack Yates, Thesis Committee Member
Date	Dr. John W. Somervill, Dean, Graduate College

ACKNOWLEDGMENTS

There are many people who have given me help and support as I progressed through my education and with this endeavor in particular. I wish to offer them my appreciation.

First, I would like to thank Dr. Mark Fienup and Dr. Jack Yates for assisting me with this work, and their instruction throughout my graduate courses. I'd also like to offer a special thanks to my advisor Dr. Eugene Wallingford. He has been my instructor and mentor from my first undergrad course through my graduate curriculum.

Next, I offer my appreciation to my friends who graciously offered to be test subjects for this research: Brian, Chris, Jeff, Justin, Nate and Vanessa. I couldn't have finished this without their help.

Finally, I wish to thank my parents for setting me on my course, and to my wife Misty for always being there to help me with this work.

Thanks to all!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE PAGE

CHAPTER 1

INTRODUCTION

Many problems in computer science are best viewed in terms of searching. To solve a problem using search, the correct solution must be found from a set of possibilities. Many problems have a high time complexity because the search space is so large. One example of a problem with a such a time complexity is the Traveling Salesman Problem (TSP). The goal in the TSP is to find the shortest path that visits each of the cities on the salesman's route once and returns to the beginning city. In the case where there are *n* cities to be visited, there are *n!* different possible solution paths.

While the TSP is difficult in many ways, it is simpler than the complex problems humans solve each day. The complexities of these problems come from the many sub-problems created when finding an overall solution. To further complicate the matter, it will never be known for certain what possible solutions exist in the search space since the set of input variables cannot be defined precisely.

Throughout the history of computer science and artificial intelligence research, game playing has served as a way to test the performance of different approaches to search. These "toy problems" are not necessarily chosen because the problem itself is interesting, but because games have clearly defined limits and rules, and the results can be applied to a larger set of more significant problems. Moreover, they still offer a complex search space.

The measure of success for this line of research is the accuracy and speed with which the system is able to find a solution. The simplest way to find a solution is by performing a blind search. This method “blindly” looks at each possible solution, one at a time, until the correct solution is found. While blind search is guaranteed to find the solution if one exists, the time required to do so relies on the speed of the machine.

Machine learning offers a different approach to blind search. The idea behind machine learning is to use knowledge of the domain to aid in solving the problem. The goal is to use domain knowledge and training data to learn how to solve the problem efficiently. Learning techniques attempt to trade off some up front computation with a reduction in the size of the search space. Differing learning techniques vary in the amount of overhead computation required.

The purpose of this study was to compare and contrast the effectiveness of different learning approaches in the card game Euchre. Four approaches of varying levels of processing were examined by how beneficial the extra cost of processing was to overall success.

While the specific game of Euchre was studied, the results are applicable to a range of problems. One such group of problems is all card games. Many card games have slightly different rules, but are very similar in design. These similarities lead to similar shaped search spaces; therefore, approaches that work well in Euchre ought to work well in learning other card games.

A larger group of problems are those that have similar shaped search spaces across all domains. The results of this study can be applied to this group of problems as well. In machine learning the learner is attempting to simplify the search space. Search spaces of similar shape and proportion may be simplified in parallel ways.

Euchre is played with a deck of 24 cards consisting of the 9, 10, J, Q, K and A of each suit. The goal is for each player and his partner to win a majority of the five tricks in each hand. A trick consists of one cycle of each player playing one card and the is won by the player who lays the highest ranking card. Each player must follow the lead suit if he is able. If he cannot follow suit, he may play trump to win the trick.

The ranking of the non-trump suits is conventional, with the Ace being high. The trump suit differs in that the Jack is the highest card, while the Jack of matching color is second. If, for example, Diamonds were trump the rank order of the cards would be the Jack of Diamonds, Jack of Hearts, Ace of Diamonds, King of Diamonds and so on. Also in this situation the Jack of Hearts is considered a Diamond, and need not be played if a Heart is led.

In the game of Euchre there are many important problems that must be solved in each hand, as well as in each game: deciding whether or not to pass the trump card, call trump, or play one card over another. I will focus on solving the latter problem.

At first glance, it may seem that there is, at most, a search space of five since that is the number of cards each hand begins with. No card player, however, decides which card to play based only on the cards in his or her hand. The complexity comes from the nearly infinite combination of factors that led to the hand. Certain combinations of factors may lead to choosing one card over another, while a different set of circumstances would point to an alternate solution. These parameters include the score, number of tricks taken, who dealt, the initial card up, the trump suit, who called trump, the suit led, which player led, the cards played, past tricks and the cards in hand. The total number of possible combinations of these parameters is approximately 9 x 10^{40}, and the total number of cards to choose from is 24. The problem is not in finding which card to play, but in finding an algorithm for doing so.

As in the TSP problem, the search space is very large and the time complexity, the time it takes to solve a problem of size *n*, is great. Therefore, it is implausible to search the entire search space. Learning offers a means to reduce the search space and the time to search for the solution.

Research Hypothesis

Four learners were programmed: two using induction and two using analytical approaches. The first used a decision tree, while the second was based on a consensus vote of a group of decision trees. The two analytical learners used rules to encode the domain knowledge. While the first used the examples to build new rules, the second learner first learned target concepts and applied those to the examples as well. Each approach used adds a higher level of processing. It is hypothesized that the higher the level of processing, the better the learner will correctly predict which card to play, and it will win more simulated games.

Statement of Problem

My program will learn one specific action in the game: deciding which card to play. This clearly defines what the system needs to learn, as well as separates different cognitive processes. In this study, my focus will not be on getting a perfect solution, but on observing what approaches are effective in getting a good solution.

Significance of the Study

This research serves three purposes:

1. It examines the effectiveness of inductive and analytical learning approaches in a specific domain.
2. The results can be applied to a range of similar card games.
3. The results can be applied in general to a larger set of problems with similar attributes.

CHAPTER 2

REVIEW OF RELATED LITERATURE

One of the first papers published that explored the possibility of a learning, thinking machine was written by Alan Turing in 1950. This was the same year that the U.S. Navy purchased the first commercially produced computer from Engineering Research Associates (The Computer Museum History Center). This fact makes it all the more impressive that his work is still relevant with today's modern technology. Turing's argument, which began simultaneously with the birth of the digital computer, is still used for discussion in current machine learning research.

Turing (1950) replaced the question, "Can machines think?" with a test. The test was to demonstrate two human characteristics in the machine: the ability to learn and the ability to act intelligently. In Turing's estimation, the correct approach would not be to program a machine that could think intelligently, like an adult, but create a child-like machine instead. It would mimic a human child's development by learning and obtaining intelligence.

Problem Solving by Search

In concluding his work, Turing (1950) states, "The learning process may be regarded as a search for a form of behavior which will satisfy some criterion" (p. 460). This concept developed into an approach for developing intelligent systems.

To view a problem as a form of search, three things are required. First, a goal must be defined, along with a way to test for its presence. Next, possible states must be determined, including the starting state. Finally, the actions required to move from one state to another must be decided (Ginsberg, 1993). A search must be done to find the sequence of states that begin in the starting state, and end in a goal state. This path is then the solution. In this fashion, it is possible to view any problem in terms of a search problem.

While it *is* possible to solve any problem with a search, doing so in complex problems would require too much time. The search space is simply too great for every possibility to be examined. So, instead of blindly searching the entire search space, domain knowledge can be used to focus the search. Using knowledge as a way to improve search provides two benefits. It allows one to change the order the states are considered, and serves as a guide to prune away portions of the search space (Wallingford, 1998).

Knowledge-Based Reasoning

In order to use knowledge to improve search, a way to represent that knowledge must first be devised. Ginsberg (1993, p. 107) outlines the intelligent problem solving process in four steps:

"1. Identify the knowledge needed to solve the problem.

2. Select a language in which that knowledge can be represented.
3. Write down the knowledge in that language.
4. Use the consequences of the knowledge to solve the problem."

Ginsberg's problem solving process requires a way to encode the knowledge in a format that can be utilized by the learner. Logic is one such language for describing knowledge. It consists of specified syntax and semantics. Syntax is a set of rules for constructing logic sentences, and semantics is tying meaning to the sentence connecting the knowledge. The domain knowledge can then be written using logic structures, which is then utilized by the machine learner.

The final step is for the learner to apply the knowledge. This only requires applying the rules to our current understanding of the environment. The knowledge rules link together and form a chain of rules. By applying the rules in sequence the knowledge base grows, thus adding more information each time a rule is applied.

In the example of Euchre there is a rule:

MustPlayDiamond ← (DiamondLed and DiamondInHand)

This rule states that if a diamond was led, and there is a diamond in hand, a diamond must be played. Therefore, if *DiamondLed* and *DiamondInHand* are in the knowledge base, *MustPlayDiamond* can be added to it. Figure 1 shows an example of rules chaining together. The initial knowledge base contains: *SuitLedDiamiond*, *RankLedA*, *SuitCard1Diamond*, *RankCard1K*. As the rules are processed, *DiamondLed* and

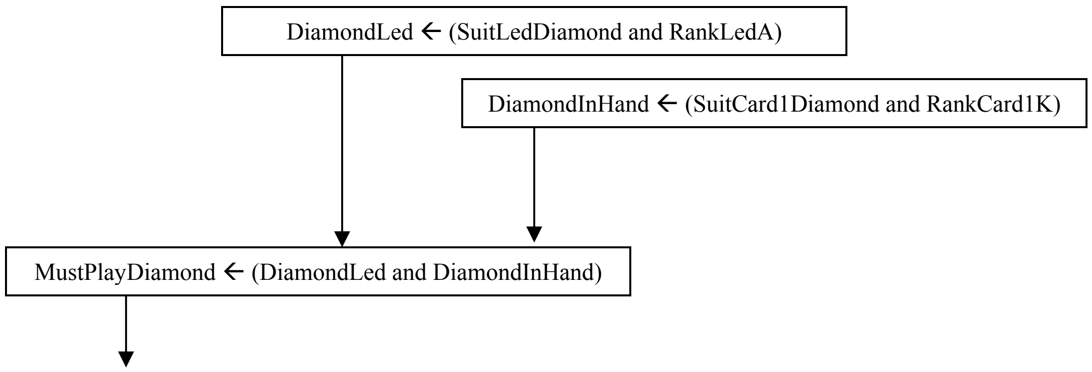
DiamondLed ← (SuitLedDiamond and RankLedA)
DiamondInHand ← (SuitCard1Diamond and RankCard1K)
MustPlayDiamond ← (DiamondLed and DiamondInHand)

Figure 1. An example of chaining rules.

DiamondInHand are added to the knowledge base. This in turn, allows *MustPlayDiamond* to be added.

While rule base systems are powerful systems, writing and organizing the rules is a daunting task. Although the syntax and semantics of the language are well defined, it is unclear how to properly encode domain knowledge or even what information should be included (Ginsberg, 1993). Due to this fact rule bases can become quite large and cumbersome, with rules interacting in unexpected ways.

Inductive Learning

One way around the difficulties of building and maintaining a rule-base is to allow the system to create its own. With inductive learning, the system is given a set of training examples, and the system induces a general rule from the inputted examples. This type of learning is widely used for classification problems (Mitchell, 1997).

Most research with induction involves the building of a decision tree. The tree contains attributes at each node, with a sub-tree for each possible value. The leaf nodes of the tree represent the corresponding classification. Figure 2 shows a partial decision tree for the concept *MustPlayDiamond*.

While inductive learning removes the complication of creating a rule-base ahead of time, it introduces new problems. For complex domains, it may take a large number of example cases to produce a reliable decision tree. These may not always be possible to gather. In addition, some of the example data may be incorrect, or inconsistent, causing noise that must somehow be accounted for.

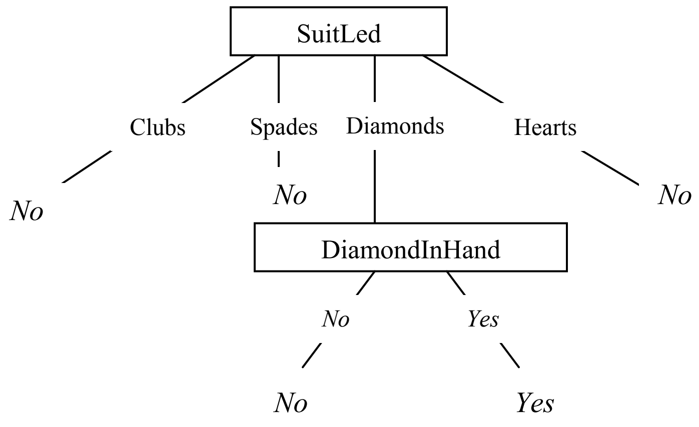

Figure 2. A partial decision tree for the concept *MustPlayDiamond.*

Analytical Learning

Analytical learning takes an explanatory approach. Along with the training data, the learner is given a domain theory. The domain theory contains background knowledge in the form of rules that can be used to form an explanation for the training cases. In the case of Euchre, the domain theory is the set of rules that define legal plays. This additional knowledge is then incorporated with the training data to create a rule that is consistent with both (Mitchell, 1997).

The benefit of using the domain theory is that it generally reduces the amount of example cases needed to produce reliable output. Fewer cases are required, because the learner begins with a basis of knowledge to build on. Without the domain theory provided, the learner must be given enough example cases to demonstrate all of the knowledge it contains.

In any set of training cases there is information that is inconsistent or irrelevant. These imperfections effect the results of the learning. The domain theory helps alleviate this “noisy” data problem by resolving inconsistencies in the data.

There is, however, a trade-off to be made. For some domains, it may be difficult, or impossible, to create a domain theory; the domain constraints may be ambiguous. Yet for some problems, a complete domain theory may not be required. In that case, it need only cover a sub-set of the domain. This may occur when certain knowledge is not relevant to the specific problem being solved. For example, rules pertaining to the selection of the trump suit were not included in the domain theory. This knowledge can be excluded because it has know relevance in choosing what card to play.

Conclusion of Literature Review

In conclusion, the literature demonstrates differing approaches to intelligent systems, specifically machine learning. Each approach has benefits and drawbacks. Moreover, each lends itself better to different domains and sets of problems. For some problems time is crucial, while in others, accuracy is more important. Therefore, the question is not which technique attains the best results, but instead, which technique best applies to the problem.

CHAPTER 3

METHODS AND PROCEDURES

The purpose of this study was to compare and contrast different learning approaches to the problem of selecting which card to play in the game Euchre. A total of four approaches were used, and each learned from a set of sample data collected from human players. The results were first measured according to the percentage of cards the learners correctly choose from a set of test cases. Secondly, results were measured by the percentage of hands won when the learners were pitted against each other.

Data Collection

The data used for learning was collected from nine Euchre players. These players had a variety of skill levels, ranging from novice to very experienced. In addition a set of random cases was included. This data offers inconsistent and "noisy" data. While this is not a scientifically chosen sample, it mirrors the situations that human learners have: poor, or noisy, data is often mixed in with good data. The novice players may actually confuse the learner if their choices are random, or not as deliberate as an expert's might be.

Three basic types of cases were given to the players:

1. Cases in which a certain card is required to be played. This occurs when the player must follow suit.

2. Constructed cases that were designed with the same cards and game situation, but differ in who led.

3. Random cases that were taken from an actual game.

Each person was given the same game facts, including previous tricks, the suit named trump, the player who called trump, etc. With this information, the tested player chose which card to play.

Learning Approaches

Inductive Learning

The first learning approach used was inductive learning. In this case, the system was given sample cases as input. The training cases included the game environment information as well as the card the human teacher chose to play. A decision tree was built from each player's training data to see how learning from a specific individual affects the results. These decision trees were then used to search for which card to play when given a new situation in a game.

This process is similar to a human learning to play the game by only observing one player's decisions. A good test to see how well the system has learned is to see if it can follow the rules in given situations where the move is forced. The algorithm for generating the decision trees is shown in Figure 3.

Consensus Voting

The next level of processing put the individual decision trees together. The decision was made by getting a solution card from each of the trees and then taking a vote. This approach alleviates the problem of noisy data by using the most prominent choice.

Inputs:

- A set of training examples, consisting of the values of the environment attributes, and the card chosen to be played.
- A default card to be played.

Algorithm:

- If the set of examples is empty, return the default card.
- If all examples have the same card chosen to be played, return that card.
- If there are no attributes left, return the most commonly played card.
- Otherwise choose the attribute *a* that best discriminates the examples.
    - Create a tree with *a* as its root.
    - Create a sub-tree for each possible value of *a.*
        - Recursively call the algorithm with the subset of examples where the value of *a* = *x,* removing *a* from the attribute set. Using the most commonly played card as the default.

Output:

- A decision tree based on the attributes from the training set.

Figure 3. Algorithm for inducing a decision tree.

Analytical Learning

Another line of attack was analytical learning. This approach used the example data and, in addition, employed the domain theory. The domain theory is a set of background information that defines the domain. In this case, the domain theory is the set of Euchre rules that determines which card can be played in a specific situation.

Each rule contains the knowledge to be added to the environment data, followed by a list of facts that must be present for the rule to apply. In this manner, the domain rules form chains of rules that link together. During each pass through the rules, the next “linking” rule may be applied and so on. This process is continued until no further facts can be added. A new rule is then added to connect the final link to the card to play, as shown in Figure 4.

This strategy benefits from already knowing the difference between a legal and illegal move. In induction, the system had to learn the rules, as well as the strategy. This analytical approach is similar to how a human might learn: first having the rules explained and then observing a player. By knowing the rules ahead of time, the learner is able to analyze why each card was played.

Inputs:

- A set of training examples, consisting of the values of the environment attributes, and the card chosen to be played.
- The domain theory, the rules of the game.

Algorithm:

- Apply the domain theory to each training example.
  - Create a new rule to play the corresponding card for the matching analysis of the input.
- Analyze the set of rules that apply to each card.
  - Create a general rule for the case that a specific suit must be played.
  - Create a general rule for when any card may be played.

Output:

- A set of rules to be used to choose which card to play.

Figure 4. Algorithm for analyzing training examples and creating rules.

Target Concept Analysis

The final method took the analytical technique one step further. First, certain target concepts were learned. This allows the system to learn how to further analyze the situational data. It can then base its decision on this deeper analysis.

The target concept algorithm in Figure 5 was run for each concept that was learned. Then, these new concept rules were added to the domain theory, and the analytical learning algorithm is run. This would be similar to a human learner being told how to analyze the situational data and identify the presence of certain concepts.

The first concept learned was when the player's partner currently was winning the trick. This concept benefits the learner by allowing it to recognize when it may be better to lose the trick so its partner will win. Next, the concept of recognizing the situation when it was legal to trump was learned. This enables the learner to recognize cases in which it can play trump to win the trick. The final concept was trump being led. This is a simple concept, but important to recognize that if the Jack matching the trump color is led, that it is considered trump.

Inputs:

- The name of the target concept to learn.
- A set of training examples, where the target concept to be learned is true.
- The domain theory, the rules of the game.

Algorithm:

- Apply the domain theory to each training example.
    - Create a new rule for the target concept for the matching analysis of the input.
- Analyze the set of rules generated and create a generalized rule.

Output:

- Rule(s) that define when the target concept is true.

Figure 5. Algorithm for learning a target concept.

CHAPTER IV

RESULTS

In general, the results gathered fit the hypothesis: each additional level of processing facilitated better learning. This analysis poses new questions, as well. On the surface, the results correlate well with the theory. However, upon further inspection, some other factors offer an alternate explanation.

The first measure of success was to test how well each learner correctly matched the trainer. This was done by giving each learner a set of test cases and comparing the card the learner chose with the one the human trainer played. Each training set comprised thirty sample cases. Sets one through nine were collected by the human players, while training set ten was a set of randomly selected cards. Table 1 shows the total number of cards chosen correctly by each algorithm for each training set. The consensus voter has a single score based on a vote among the decision tree learners.

One interesting result is that training sets one and nine produced a higher total correct in decision tree learning. Furthermore, the set of random data proved to be a better training set than that of some actual players. These results can be explained by the fact that decision tree learning attempts to “think” like the trainer. In this way, the learning system is not learning “what card to play based on the situation,” instead, it learns “what card would the trainer play in this situation.” The varying results of the different training sets demonstrate the importance of the quality of data in inductive learning.

Table 1.

The Number of Cards "Correct" for Each Set of Training Data

	Training Set												
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Mean**	**Standard Deviation**	**Correct**
Decision Tree	12	5	3	2	5	4	4	5	10	5	5.50	3.10	*18.33%*
Consensus Voting					9						9.00	0.00	*30.00%*
Analytic Learning	23	19	16	18	18	16	16	18	21	16	18.10	2.38	*60.33%*
Target Concept Analysis	24	20	17	18	19	17	17	19	23	16	19.00	2.67	*63.33%*

Note. The totals correct are out of a possible 30.

The most noticeable fact shown from the results is that the analytical techniques (Analytic Learning and Target Concept Analysis) had a significantly higher percentage correct than the inductive ones (Decision Tree and Consensus Voting). The cause of this disparity is the fact that the analytical approaches were given the domain theory as input. Therefore, these systems already knew the rules for legal moves. Half of the test cases used were forced decisions, which is a situation where there was only one legal move. Table 2 shows the totals correct for the fifteen non-forced situations in the testing set.

These numbers show a different picture. The last two techniques benefited from getting all fifteen of the forced situations correct, which skewed the overall numbers in

Table 1. When a comparison is made of only those situations where a decision between multiple legal moves is made, the Consensus Voting method achieved the highest success.

Table 2.

<u>The Number of Cards "Correct" for Each Set of Non-Forced Situation Training Data</u>

	Training Set												
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Mean**	**Standard Deviation**	**Correct**
Decision Tree	9	2	1	2	3	1	1	2	7	2	3.00	2.75	*20.00%*
Consensus Voting	6										6.00	0.00	*40.00%*
Analytic Learning	8	4	1	3	3	1	1	3	6	1	3.10	2.38	*20.67%*
Target Concept Analysis	9	5	2	3	4	2	2	4	8	2	4.10	2.56	*27.33%*

<u>Note</u>. The totals correct are out of a possible 15.

The programs based on the first and ninth training sets achieved a better learning rate than the others for all approaches again. This suggests that the effect of training data is consistent across the techniques. Data that achieves a higher success rate in one type of learning likewise achieves a higher success rate in another. Moreover, the data sets that had a low success rate were lower for every technique.

The fact that the success levels correlate among the learning approaches demonstrates the importance of the training data and its affect on results. Additionally it shows that the effectiveness of the example cases varies across the individual set of answers. This suggests it would be possible to construct a specific set of examples that best demonstrates a certain pattern.

The second measure of the systems was to pit them against each other and see which won the most hands. This was done with a set of fifteen random game situations. Each system was partnered with itself and played against a partnership of each of the other systems. Then each of the games was played a second time with the table positions switched so that a fair judgment could be made. Each row in Table 3 shows the total hands won for each algorithm.

Table 3.

The Number of Hands Won by Each Technique Against Each Other Technique

VS	**Decision Tree**	**Consensus Voting**	**Analytic Learning**	**Target Concept Analysis**	**Percent Won**
Decision Tree		14 *(46.67%)*	9 *(30.00%)*	9 *(30.00%)*	*35.55%*
Consensus Voting	16 *(53.33%)*		13 *(43.33%)*	12 *(40.00%)*	*45.55%*
Analytic Learning	21 *(70.00%)*	17 *(56.67%)*		14 *(46.67%)*	*57.78 %*
Target Concept Analysis	21 *(70.00%)*	18 *(60.00%)*	16 *(53.33%)*		*61.11%*

Note. The totals won are out of a possible 30.

As shown in Table 3, the decision tree learner won a total of 14 out of the 30 hands against the consensus voter, while it won only 9 against each of the analytical learners. This is winning percentage of 35.55% of the 90 total hands played.

The consensus voter had a slightly higher total winning percentage of 45.55%. While the consensus voter won 16 of the 30 hands against the decision tree learner, it only had totals of 12 and 13 against the two analytical learners.

In this testing, the two analytical approaches played very similarly. In fact, there were only a few cases in which they chose different cards to play. This small amount of variance is what allowed the fourth technique to win a larger percentage against the third.

The last two approaches benefited again from the domain theory. The inductive approaches scored lower winning percentages because in some cases they played illegally. This caused them to renege, and thus, a hand that otherwise may have been won was lost.

This same test was conducted a second time (Table 4), but the inductive approaches were forced to only make legal plays. This alleviated the problem of the inductive approaches reneging. The percentages won by each approach in this testing were very close. When the rules were enforced on the inductive approaches, they were as effective as the analytical ones. This indicates that the extra processing of encoding the domain theory required by the analytical approaches may not have been cost effective.

Table 4.

The Number of Hands Won by Each Technique Against Each Other Technique With the Game Rules Enforced

VS	**Decision Tree**	**Consensus Voting**	**Analytic Learning**	**Target Concept Analysis**	**Percent Won**
Decision Tree		14 *(46.67%)*	15 *(50.00%)*	14 *(46.67%)*	*47.78%*
Consensus Voting	16 *(53.33%)*		15 *(50.00%)*	16 *(53.33%)*	*52.22%*
Analytic Learning	15 *(50.00%)*	15 *(50.00%)*		14 *(46.67%)*	*48.89%*
Target Concept Analysis	16 *(53.33%)*	14 *(46.67%)*	16 *(53.33%)*		*51.11%*

Note. The totals won are out of a possible 30.

In this second test, the totals won by each approach are very close. Table 5 shows the total each won as well as the standard deviation of the totals for each test. The deviation among the totals dropped significantly in the second test. This demonstrates that all learning techniques perform relatively the same, when forced to make only legal moves.

Table 5.

The Total Number of Hands Won by Each Technique

	Decision Tree	**Consensus Voting**	**Analytic Learning**	**Target Concept Analysis**	**Standard Deviation**
Test 1: Without Rules Enforced	32	41	52	53	*10.55*
Test 2: With Rules Enforced	43	47	44	46	*1.83*

Note. The totals won are out of a possible 90.

These totals also demonstrate that the inductive and analytical approaches both realize similar results once illegal moves are excluded. The more complex of each learning approach, the consensus vote and target concept analysis, performed only slightly better than the simpler ones. Both the consensus voter and target concept learners required additional data collection and encoding of knowledge. This analysis indicates that this extra work required by these approaches may not be of great benefit.

CHAPTER V

DISCUSSION AND FURTHER WORK

This research demonstrates that, when forced to make legal moves, inductive and analytical approaches achieve similar levels of success for learning to play Euchre. The decision tree and analytic learners averaged nearly the same accuracy when choosing between multiple legal moves, as shown in Table 2. The total number of hands won by each approach in Table 5 demonstrates this similarity in playing hands as well.

One factor that affected the results of the second testing was the fact that the inductive approaches were not allowed to make illegal moves. While the player must follow the rules to win, knowledge of the rules did not prove to be the most important factor in success.

When the learners were pitted against each other they played fifteen hands. Then the positions were switch and these same fifteen hands were played again. This ensured that each player had equal opportunity to win the thirty total hands. If the abilities of the learners were totally equalized by knowing the rules, then they should each win half of the hands.

Another factor in the closeness of the results is that the number of decisions to make in one hand is small. On average each player would have to choose between two legal cards to play. While this choice is very important to strategy and winning tricks, some such cases do not affect the outcome of the hand, because the cards are of essentially equal power. For example, if the player must choose between playing the 9 or 10 of a suit, there is no advantage to play either card over the other. With many more hands played, the advantage of the consensus voter may grow as the number of "meaningful" decisions increases. With the consensus voter correctly choosing at a higher percentage than the other learners in these non-trivial cases, it will win more games.

This study also demonstrated the importance of the training data on how well the learner performed. While one would expect the training sets from stronger players would produce better learners, the results do not demonstrate that fact. Of the training cases selected, 1, 3, 4, 6, 8 are those of stronger players, and the others of novice players. Learners from both sets of players averaged the same totals correct shown in Table 2.

The reason it would be expected that the training sets of better players would produce better learners, is that the algorithms learn to play consistent with the training set. In this way they are learning a pattern in the behavior. Some players' strategies are more easily learned than others. For example, one player may always play a specific card when able. This means that it is likely that there may be many training cases in which that card is played. This is a shallow strategy and can be quickly learned from a small set of cases. On the other hand, another player may choose a card to play on many factors. This is a more complex strategy, and therefore may take a much larger training set to recognize the pattern.

All of the learning algorithms had higher totals correct for the training sets one and nine. Similarly, some training sets show poor performance across all approaches. This demonstrates that effectively building the set of training cases used in the learning has a great effect on the outcome. Additionally, since the learning approaches correlated with the training sets a conclusion can be made that the difference in training sets were more important than the differences in the algorithms.

Significance of the Results

The first goal of this research was to demonstrate machine learning as applied to a specific domain of Euchre. To do this, two differing approaches were used: inductive and analytical learning. Machine learning requires a way to represent knowledge about the problem. With the information encoded in a usable format, the learner is able to apply the knowledge and learn from example cases. Decision trees and rule bases are two different ways to r
epresent the same knowledge. The results of this research demonstrate that both models may apply and perform well in this specific problem.

Another goal was to be able to apply the results from this example to a range of similar card games. A common characteristic of all "trick" based card games is that they have a hierarchy of cards. This hierarchy is a part of the domain rules, and is irrelevant to the learning of strategy. For example, in Euchre the jack of the trump suit is the highest card, while in Pitch the ace is high. This information is not important to learn the strategy of playing the highest valued card. Since the underlying goal for this type of card game is similar, one may predict similar success by applying the same learning approaches.

This research also indicates the importance of the training data. It demonstrates that, in applying machine learning to other card games, it may be most effective to focus time on developing the training cases. Quality of training cases will dominate differences in learning techniques with this type of search.

The final goal was to apply the results to a larger, more generalized set of problems. In Euchre, there are 24 possible cards to play. As calculated before, there are approximately 9×10^{40} possible combinations of input. There are many problems across many domains that have a relatively small set of output options and a very large amount of possible input sets.

Figure 6 shows shapes of different types of search spaces. The size of the input data is indicated by the width at the top of the diagram, while the base represents the number of possible output solutions. The first search space represents problems similar to Euchre. In this search space the possible solutions are greatly outnumbered by the possible combinations of input. This search space contrasts with the others in which the number of possible solutions is greater than the input variables, or where the size of the input and output are proportional in magnitude.

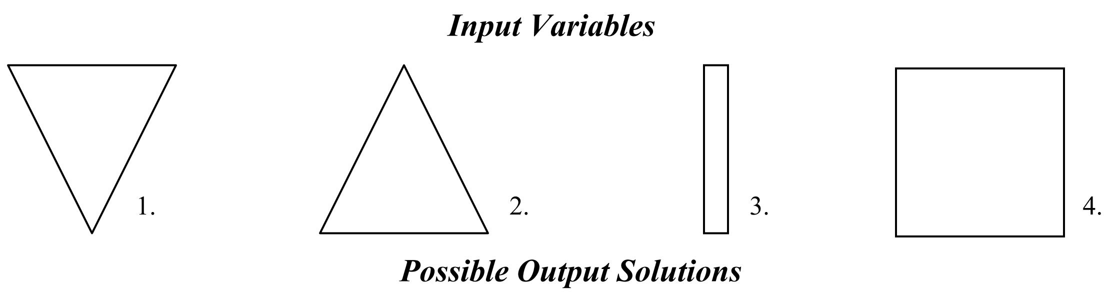

Figure 6. Shapes of different search spaces.

One particular domain where there are many problems that have a similar structured search space is meteorology. The combination of inputs required to predict the likelihood of precipitation, temperature range and other weather patterns is nearly infinite. While the range of possible output values in these examples is small in comparison.

It is the proportion of the search space that influences how the learning techniques respond in that domain. The learning algorithm is designed independently from the type of knowledge that is being learned. The "shape" of the problem determines what approaches are efficient and cost effective. This means that an effective approach to learning which card to play in Euchre can be used to learn how to predict tomorrow's high temperature.

Machine learning reduces the amount of the search space examined. It is reasonable to assume that similarly shaped search spaces can be reduced in a parallel fashion. Demonstrating the effectiveness of different approaches using a small example like Euchre may give an indication of what approach may achieve the best results in a larger problem with similar constraints.

Limitations of the Study

While the game of Euchre is a good problem to demonstrate the effectiveness of machine learning and what approaches to use in similar problem types, it has limitations. This example posses aspects that other problems may lack.

One of these limitations is the learning ceiling. In many problems there is a range of skill levels. In the game of chess, players are ranked and the best players require years of practice and study. While in chess the skill level of a master differs greatly from that of a novice, in Euchre there is only a small difference. This is because by the nature of the game of Euchre, the skill of a player quickly reaches a plateau. This learning ceiling means that the machine learner may reach a high level of success quickly, and not progress from there. This must be taken into account when extrapolating these results to a more complex problem.

Another matter that should be considered is the structure of the training sets. In this study random game situations were used. This was done to study how well the learners would learn under similar conditions to a human learner. Additional research could be performed using sets of data structured to focus the learning in a specific direction. In addition, different sizes of training data could be examined to find how the size of the data set affects the results.

Conclusion

The goal of machine learning is to use knowledge to aid in solving complex problems. Using knowledge to focus the search can reduce the amount of the search space that must be examined. This concept is an attempt at modeling a human's cognitive processes, rather than simply relying on the computer's vast computational power to blindly search for the solution. Even while the speed and power of today's machines is increasing rapidly, some problems have a time complexity too great to be overcome by speed alone.

This study demonstrates that machines, like humans, can learn in multiple ways. The inductive approaches learn simply on observation, in the same manner humans are able to watch examples and internalize what they see. The analytical technique corresponds to a person first learning and understanding the domain, and then observing examples. One difference between human learners and the machine learners in this study is that the human learners are able to ask questions of the instructor. This form of validation assists learning, and serves another approach to pursue in machine learning.

The successes and failures of different knowledge representations and learning approaches in smaller problems may help direct research on larger problems with similar constraints. This study shows that the outcomes of inductive and analytical approaches are closely equivalent, which may give insight in how to approach similar type problems.

REFERENCES

Computer Museum History Center, The. (Unknown). Timeline of computer history. Retrieved August 14, 2000 from the World Wide Web: http://www.computerhistory.org/timeline/

Ginsberg, M. L. (1993). Essentials of artificial intelligence. San Francisco: Morgan Kaufmann Publishers.

Mitchell, T. M. (1997). Machine learning. Boston: WCB/McGraw-Hill.

Turing, A. M. (1950). Computing machinery and intelligence. Mind, 59 (236), 433-460.

Wallingford, V. E. (1998). Satisfying goals through heuristic search. Retrieved September 16, 1998 from the World Wide Web: http://www.cs.uni.edu/~wallingf/teaching/161/sessions /session06.html