Applications of Decision Tree Learning

Nick Biddle – 0185799

Mike Schulenberg – 1409566

Joshua Strain – 1332695

**Part I. Survey**

In his book *Machine Learning*, Tom Mitchell states that decision tree learning methods “are among the most popular of inductive inference algorithms and have been successfully applied to a broad range of tasks” (Mitchell 52). What might explain their popularity, and what sorts of tasks might benefit from their capabilities? Often, decision trees are desirable because the knowledge they represent conforms to an “if-then” rule structure easily understood by humans, and their ability to handle both text and numerical attributes increases their utility (Ruey-Shiang, Tsung-Chieh and Shao-Ping 4439). Because of their flexibility, these algorithms have been successfully employed in a variety of tasks such as freeway incident detection, medical diagnostics, and image recognition.

Although many neural network algorithms have been developed to detect incidents on freeways, the nature of their connection weight-based rules result in knowledge that is hard to understand. Additionally, neural networks maintain a complex structure with many connections, as opposed to the simpler, branching structures demonstrated by decision trees. To address these shortcomings, one study examines the application of decision tree learning to the problem of incident detection. (Shuyen and Wei 4101-4102)

The study essentially views incident detection as a classification task with “incident” and “non-incident” classes as the possible output. Using Ross Quinlan’s C4.5 algorithm, the study constructs a decision tree that learns from data collected from a total of 70 simulation runs, with 40 acting as training runs, and the remaining 30 reserved for the testing phase. Nodes in the resulting decision tree represent threshold values, and when these values are exceeded, the tree outputs an incident. (Shuyen and Wei 4102-4103)

To evaluate performance, the study relies on three measures commonly used to evaluate incident detection algorithms: detection rate, false alarm rate, and mean time to detect. It also examines classification rate, a measure of the percentage of correct classifications, regardless of whether they belong to the incident or non-incident class. The results are then compared to both back-propagation neural network (BPNN) and radial basic function neural network (RBFNN) architectures for a final evaluation (Shuyen and Wei 4102-4013).

Ultimately, the study finds that the decision tree algorithm offers similar performance to the BPNN and RBFNN algorithms, while producing acceptable results in detection rate, false alarm rate, and mean time to detect. However, because the decision tree model is easier to understand, its shows promise when applied to automatic incident detection. (Shuyen and Wei 4105)

In the sphere of medical diagnostics, a different study examines the use of decision tree learning to help physicians predict the outcome of in vitro fertilization (IVF), as well as determine how best to tailor the IVF treatment to a patient’s individual needs and improve the pregnancy success rate. Because the physician must normally consider a huge number of variables, the decision-making process develops an intuitive component, where accuracy might provide a better outcome. (Ruey-Shiang, Tsung-Chieh and Shao-Ping 4437)

As with the study involving automatic incident detection conducted by Shuyan and Wei, the IVF researchers choose the decision tree model because of the clarity provided by its “if-then” rules structure. However, the IVF researchers must contend with the additional difficulty of selecting the right attributes from a vast array of diverse medical data. To extract a near-optimal set of attributes from which to construct a decision tree, the study employs a genetic algorithm search technique, where chromosomes represent attributes taken from IVF data records. (Ruey-Shiang, Tsung-Chieh and Shao-Ping 4438)

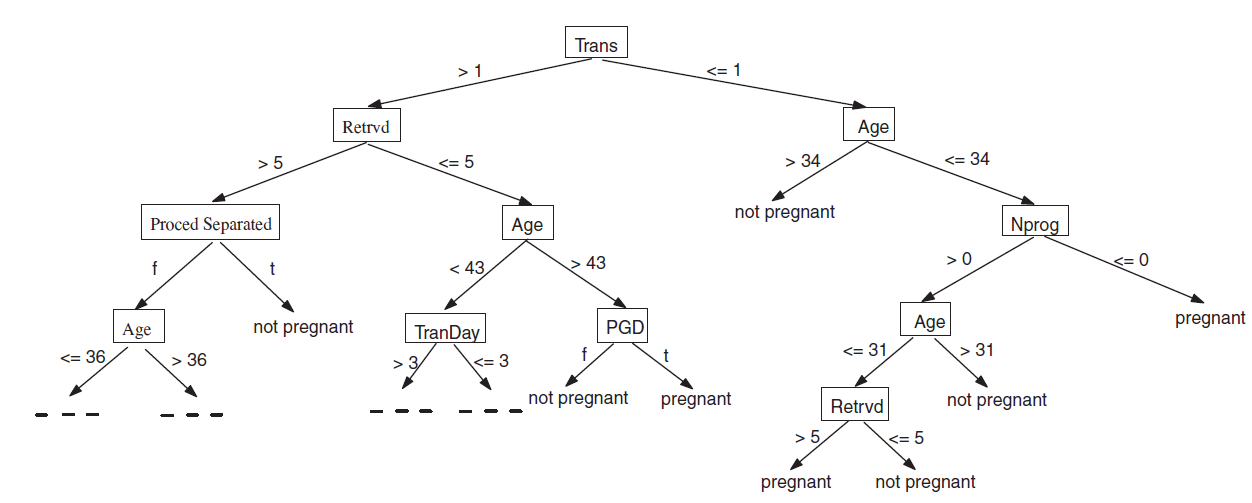


Figure 1. A sample DT for predicting the IVF outcome with IVF patient clinical data. Source: Integrating genetic algorithm and decision tree learning for assistance in predicting in vitro fertilization outcomes (4438)

This decision tree model allows for two possible results, pregnant and non-pregnant, but also tracks the number of false predictions (Ruey-Shiang, Tsung-Chieh and Shao-Ping 4443).

While the study results in a slightly worse predictive accuracy when compared to data mining attempts using other models, this hybrid of genetic and decision tree algorithms results in “a set of if-then rules that can lead to more insights into the combinative relationships among the relevant IVF attributes” (Ruey-Shiang, Tsung-Chieh and Shao-Ping 4448).

Decision tree learning also shows promise as a component in content-based image retrieval systems. In conventional systems, performance is negatively impacted by what one study refers to as a “[semantic gap] between low-level image features and the richness of user semantics” (Ying, Dengsheng and Guojun 2554). To increase retrieval accuracy and improve performance, the study examines the use of decision tree learning to bridge the gap between image features and semantics. (Ying, Dengsheng and Guojun 2554)

However, standard decision tree algorithms present some issues when applied to an image retrieval system: (1) they lack a pruning technique optimized for image semantic learning, and (2) they have difficulty in discretizing the features of an image. While algorithms such as C4.5 can process continuous attributes, they do not perform as well as when handling discrete attributes. (Ying, Dengsheng and Guojun 2555)

To address these issues, the study proposes a new learning algorithm called DT-ST that includes methods for simplifying the decision tree and discretizing image features. DT-ST solves the first issue by applying “a hybrid of pre-pruning and post-pruning techniques in order to resolve the noise and tree fragmentation problems. As a result, the tree grows in a [well-controlled] manner and the classification performance is improved” (Ying, Dengsheng and Guojun 2555). For the second issue, “DT-ST converts low-level color/texture features into color/texture labels, thus avoiding the difficult image discretization problem” (Ying, Dengsheng and Guojun 2555). These custom methods improve the classification accuracy of DT-ST compared to popular decision tree algorithms such as ID3 and C4.5, resulting in better performance. (Ying, Dengsheng and Guojun 2555)

How well does DT-ST perform overall? When compared to conventional content-based image retrieval systems, the system based on DT-ST demonstrates a performance improvement exceeding 10%, a significant decrease in the semantic gap between image features and semantic features. The study further notes “that in addition to being useful as a stand-alone image retrieval tool to search image databases, the proposed system can also be integrated with other text-based image search systems to improve their retrieval performance” (Ying, Dengsheng and Guojun 2569).

If decision tree learning techniques can produce successful outcomes when applied to automatic incident detection, medical diagnostics, and image recognition, can they do the same for casino games such as blackjack by maximizing the expected utility of each hand? A paper released in 2010 describes an intelligent agent that “uses decision tree learning to determine the optimal way to play a hand. It uses the same method of determining optimal actions to also calculate aggregate probabilities of winning and uses this information in the placing of a bet based on its utility function” (Goyal, Kjeldergaard and Deshmukh 13).

**Part II. Application**

Our goal now shifts from understanding a decision tree in concept to attempting to simulate one using standard programming techniques. We chose to experiment with a game of blackjack. The decisions made during a game of blackjack “conform to an ‘if-then’ rule structure,” meaning it is possible to model them with a decision tree. This experiment required implementing a simple two player blackjack game. In this game, one of the players is human and the other is the computer. The computer can make choices randomly or based on a simulated decision tree. It is expected that the computer will play the game much better when using the decision tree to make informed decisions.

The game designed for this experiment uses slightly simplified casino blackjack rules. In each round, a player and a computer player are dealt two cards from a standard deck of 52. The values of these cards are added to give the hand score. Each card has a value equal to the number on it, except for Aces, Kings, Queens, and Jacks. Aces are worth 1 or 11, depending on which is more beneficial. Kings, Queens, and Jacks are all worth 10. The goal of the game is for the player to reach a total score 21 without “busting”. Busting refers to when a player’s score exceeds 21. If this happens, the player loses.

After the initial cards are dealt, the players are each given a turn where they have 3 options: hit, stand, or split. The “split” option can only be selected if the initial hand is a pair of cards with the same value. If a player decides to split, their cards are split into two new hands, and they are dealt two more cards. The player wins if either hand beats their opponent. Players cannot split after the first turn. The “hit” option means the player is dealt another card for their current hand. The player’s score is increased by the value of the card. Players can hit as many times as they want or until they bust. The “stand” option indicates the player is comfortable with their hand and does not want any more cards. The computer player in this game is treated the same as the human player, and is given the same choices for each turn. The game ends when both players stand with scores less than or equal to 21 or when either player busts. If both players bust, the game is considered a “push” or tie. If either player splits, that player can win if either hand beats their opponent. If both players split, the player with the hand closest to 21 wins.

For the players to win a round in blackjack they must beat the dealer, to accomplish this the player must score higher than the dealer without busting. When a player can see 1 or more of the opponent’s cards, complex strategies can arise using only the 3 options hit, stand, or split. In general, experienced players play by a set of pre-determined rules designed to maximize the probability of a win. These rules were modelled with the decision tree designed for this experiment. Figure 2 shows a representation of the tree. When playing the game designed for this experiment, the player is given the option of having the computer make decisions randomly or by using the decision tree. If the random option is selected, the computer will not consider any of its opponent’s cards, and will choose to hit, stand, or split at random. If the tree is selected, the computer will play based on the strategy outlined in Figure 2 and attempt to maximize the probability of a win based on the current cards on the table.

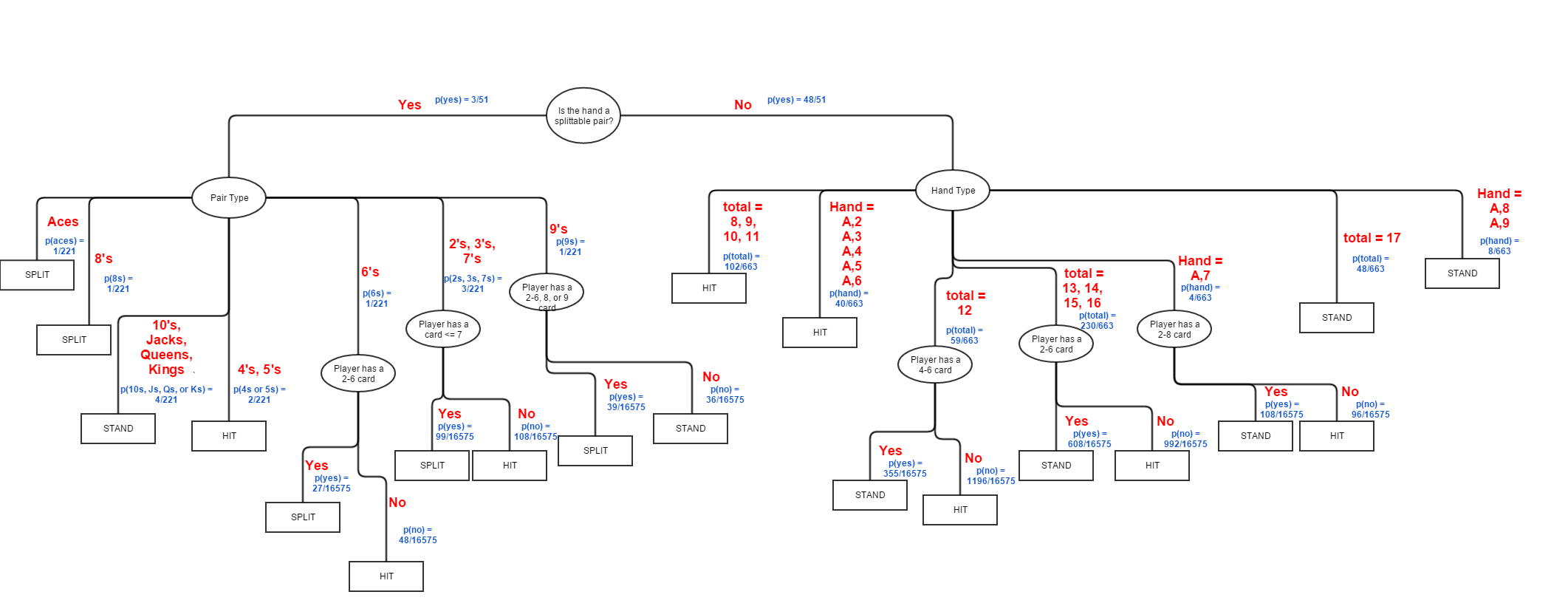


Figure 2: The decision tree that was designed and implemented for this experiment. The internal nodes are ovals, the terminal nodes are rectangles. Attributes for each branch are shown in red and their corresponding probabilities are shown in blue.

The tree shown in Figure 2 has two types of nodes: internal nodes and terminal or “leaf” nodes. Internal nodes split into different branches according to the different values of corresponding attributes. Each branch terminates at a leaf node. The terminal nodes correspond to the class of the choice made by the computer. These classes are the available choices each player has during their turn: hit, stand, or split. To use this tree, a user would start at the root node and proceed down the branches specified by both players’ hand attributes, ending at a terminal node that indicates the best choice (hit, stand, or split) for the current set of card attributes.

The probabilities shown in Figure 2 are the probabilities of each branch being taken. They were calculated based on a single deck of 52 cards. The first branches from the root node show the probabilities of getting or not getting a pair of cards with the same value. The next level of internal nodes corresponds to hand type. If a pair was dealt, the pair is then classified. If two cards with different values are dealt, then the hand is classified by total score or cards present. The probability of a branch from one of these nodes corresponds to the probability of the first attribute (the cards being a pair or not) AND the attributes listed for that branch. For example, the probability of being dealt pair of aces is the probability of being dealt an ace and the conditional probability of being dealt another ace given that the first card was an ace.

Figure 2 can be used to calculate the entropy of the choices as follows:

Entropy(S) =

The entropy associated with splitting:

5 split leaves, 15 non-split leaves, 20 total leaves.

Entropy(split) = - ) = **0.81**

The entropy associated with hitting:

8 hit leaves, 13 non-split leaves, 20 total leaves.

Entropy(hit) = - ) = **0.93**

The entropy associated with standing:

7 hit leaves, 14 non-split leaves, 20 total leaves.

Entropy(stand) = - ) = **0.89**

Data was generated by playing the game 100 times. Half of the games were played with the computer making random choices when asked to hit, stand, or split. The other half were played with the computer using the decision tree to assess each situation and make informed decisions. The number of computer wins, player wins, pushes (ties), computer busts, and player busts were collected and plotted in Figure 3 and Figure 4.

Figure 3: The number of computer wins, player wins, pushes, computer busts, and player busts after 50 games played with the computer making random decisions.

It is clear from Figure 3 that the computer did not perform very well when making random choices. The number of computer busts exceeds the number of computer wins, and the number of player wins exceeds the number of computer wins by a wide margin. The number of player busts was lower than the number of computer busts as well. This result was expected as some strategy is required to win frequently.

Figure 4: The number of computer wins, player wins, pushes, computer busts, and player busts after 50 games played with the computer making informed decisions.

Figure 4 shows a much better performance by the computer player. The number of computer wins is greatly increased from the random trials. The number of computer busts is also lower than the number of player busts. Fewer computer busts led to fewer pushes as well since pushes happen when both players bust. Overall, using the decision tree led to the computer performing competitively rather than making what human players would consider to be foolish mistakes. This is due to the computer choosing the option that would maximize its probability of winning by traversing the decision tree at every turn.

**Part III. Conclusion**

Although different requirements have resulted in decision tree learning algorithms of varying capabilities, according to *Machine Learning*,these methods are mainly effective for problems that share a certain set of characteristics:

* “Instances are represented by attribute-value pairs […]
* The target function has discrete output values […]
* Disjunctive descriptions may be required […]
* The training data may contain errors […]
* The training data may contain missing attribute values” (Mitchell 54)

Many studies have demonstrated that a considerable number of problems conform to these characteristics, making decision tree learning useful in a broad range of applications from automatic incident detection, to medical diagnostics and prediction, to image recognition and retrieval. As the blackjack simulation described in Part II has shown, visualizing a problem in terms of how a decision tree might break it down, even when not employing a decision tree algorithm directly, can assist in the implementation of a solution, thanks to the ease with which its “if-else” rules structure can be understood. With increases in computing power, coupled with the further refinement of decision tree learning, perhaps one day these algorithms can break free of their current limitations, propelling human knowledge to heights as yet unimagined.

# References

Goyal, Sanchit, et al. "Intelligent Agent for Playing Casino Card Games." 2010. *Midwest Instruction and Computing Symposium.* Web. 21 April 2017. <http://micsymposium.org/mics\_2010\_proceedings/mics2010\_submission\_49.pdf>.

Mitchell, Tom. "Decision Tree Learning." *Machine Learning*. McGraw Hill, 1997. 52-77.

Ruey-Shiang, Guh, Jackson Wu Tsung-Chieh and Weng Shao-Ping. "Integrating genetic algorithm and decision tree learning for assistance in predicting in vitro fertilization outcomes." *Expert Systems with Applications* 38.4 (2011): 4437-4449.

Shuyen, Chen and Wang Wei. "Decision tree learning for freeway automatic incident detection." *Expert Systems with Applications* 6.7 (2009): 4101-4105.

Ying Liu, Dengsheng Zhang, Guojun Lu. "Region-based image retrieval with high-level semantics using decision tree learning." *Pattern Recognition* 41.8 (2008): 2554-2570.