IT-3105 Lecture: AI Poker Players

1 AI and Games

Although most worthwhile games are complex enough to warrant player classifications such as novice, amateur, expert, grand-master, etc., they are not necessarily equally difficult for AI programmers, especially the hackers of the 21st century. Chess, checkers and backgammon, for example, clearly challenge the human brain, but AI programs now routinely defeat the biological world champions in these pursuits.

A simple 4-part categorization of games can partially explain these differences in their *AI vulnerability*:

- **Perfect Information Game** - All players can assess the complete state of the game at all times. That includes both the state of the playing arena (e.g. the board) and the states of all opponents' holdings (e.g. cards, tokens, chess/checkers pieces, etc.).
- **Imperfect Information Game** - Some information about the state of the playing arena or the opponents' holdings are not available to all players at all times.
- **Deterministic Game** - The immediate outcome of any basic move is not susceptible to chance and is thus completely under the control of the players themselves. The state of the game can only be modified by these basic moves.
- **Stochastic Game** - Chance plays a role in the determination of possible states, whether starting states or other states. Any games involving dealt cards or rolled dice, for example, are stochastic.

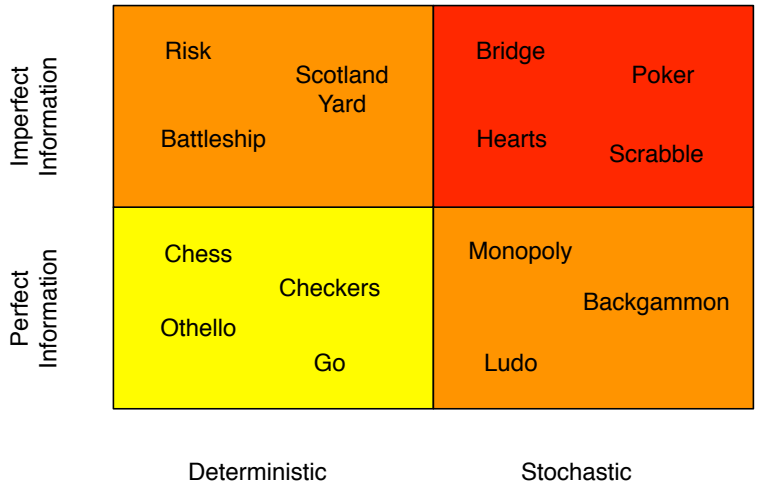

Figure 1: Game Classifications

In general, those games in the bottom left corner of Figure 1 should be easiest for a computer program with significant available memory to investigate deep search trees. The game of Go is a notable exception, due to it's enormous space of possible moves and significant patterns (of stones). Some of the games in the upper left and lower right may, despite hidden information or stochastic actions (respectively), sprout search trees (that incorporate this uncertainty) of manageable size. Backgammon, for example, has admitted world-class AI players for over a decade.

However, most games in the upper right present huge challenges to AI via their combination of hidden information and random events. It should therefore come as no surprise that championship-level AI poker players have taken much longer to invent than grand-master- slaying chess programs.

2 AI Poker Players

In the early days of AI, poker-playing systems sprang up along with chess and checkers players, but with much less success. The two most classic systems are described in these articles:

- Nicholas Findler, *Studies in Machine Cognition Using the Game of Poker*, Communications of the ACM, 20(4), 1977.
- Donald Waterman, *A generalization learning technique for automating the learning of heuristics.* Artificial Intelligence, 1, 1970.

More recently, AI researchers at the University of Alberta have crafted impressive poker players, as discussed in these and many other articles from that institution:

- Darse Billings, Denis Papp, Jonathan Schaeffer, Duane Szafron, *Opponent Modeling in Poker.* Proceedings of AAAI-98, 1998.
- Darse Billings, Aaron Davidson, Jonathan Schaeffer, Duane Szafron, *The Challenge of Poker.* Artificial Intelligence, 134(1-2), 2002.
- Michael Johanson, *Robust Strategies and Counter-Strategies: Building a Champion Level Computer Poker Player.* Masters thesis, University of Alberta, 2007.

Many of the methods described in this document stem from the Alberta research.

2.1 Basic Components

As depicted in Figure 2, a successful automated poker player consists of (at least) the following three components:

1. Hand evaluator - determines the general winning potential of a player's cards (a.k.a. her *hand*)
2. Action Selector - Given the current situation, this determines the next action: Fold, Check/Call or Bet/Raise.
3. Opponent Modeler - Uses playing histories to make assumptions about the types of opponents one is facing.

Detailed examples of each component appear in later sections of this document.

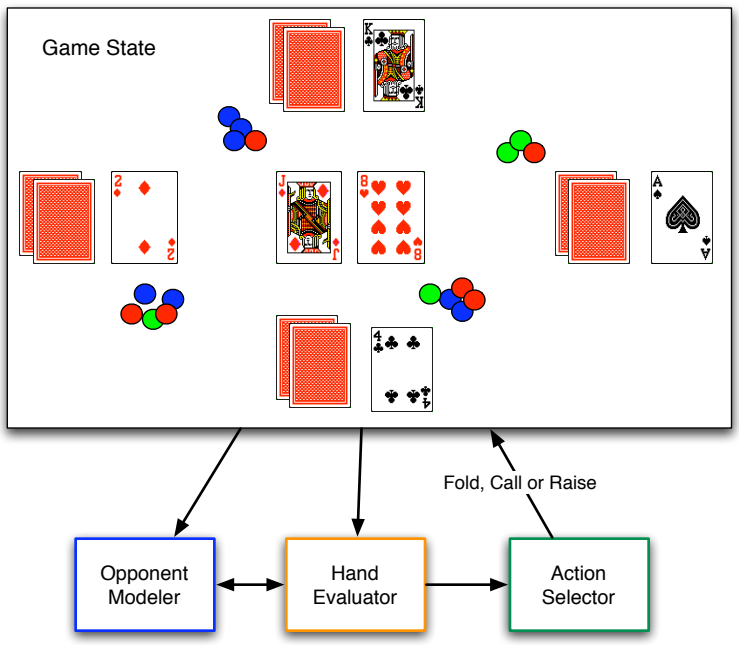

Figure 2: Basic components of a generic AI poker player.

2.2 Waterman's Poker Player

In 1968, Donald Waterman wrote a system to play 5-card draw using basic principles of expert systems. Figure 3 summarizes his system.

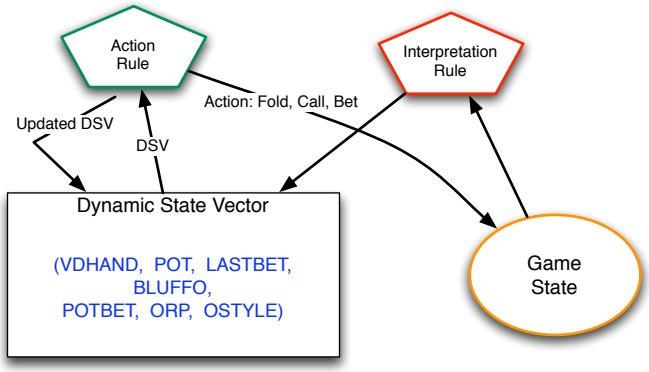

Figure 3: Basic elements of Waterman's Poker Player from 1968.

Interpretation rules take basic data as input and produce higher-level concepts. For example, the following rule simply classifies the amount of resource in the pot:

If POT $>$ 50 $\Rightarrow$ POT $\leftarrow$ BIGPOT

Action rules are more complicated. They produce both an updated dynamic state vector (DSV) and an

action (that will affect the game state). For example:

IF: VDHAND = SUREWIN
and POT = BIGPOT
and LASTBET = POSITIVEBET
THEN: POT ← POT + (2*LASTBET)
and LASTBET ← 0
and ACTION = CALL

These rules were organized into a **production system**, which fires the first rule whose left-hand side matches the DSV, in much the same way as the expert systems of the 1980's, 1990's, and present. Waterman's main contribution was actually to Machine Learning, since these rules were learned by his system. His system did not have a sophisticated method for evaluating hands nor for opponent modeling. Still, it was one of the first half-decent poker-playing systems, and thus an AI classic.

Since human game-playing experts probably do use (hundreds or thousands of) rules, the rule-based expert-system approach seems, at least in theory, quite reasonable. However, the decision-making heuristic rules used by human brains are not always easy to formalize. The human brain is a pattern-matching system far superior to modern machine perception systems, thus enabling us to detect patterns of such intricacy and complexity that they defy our abilities to cognitively contemplate or describe them. Hence, we cannot assist an AI knowledge engineer in formalizing them in a computer program, and thus, the left-hand sides (i.e. preconditions) of most expert-system rules will express far simpler sensory patterns than those (effortlessly) detected by our own brains. The computer is at a serious disadvantage when trying to use human-like rules to play against humans.

As illustrated in later sections, today's computers can turn the tables on humans by exploiting their perfect memories and systematic search capabilities. For example, in 5-card draw, a contemporary laptop can quickly compute all possible hands of a single opponent and their relative strength against a focal player's (known) hand. But in Waterman's day, these brute-force searches were only possible on a relatively small scale, and thus the heuristic rule-based approach presented a better alternative.

2.3 Findler's Poker Players

In 1977, Nicholas Findler published results summarizing the performance of a host of automated, 5-card-draw strategies. His system included:

1. Hand evaluations based on monte-carlo simulations.
2. Betting decisions based on a dynamic-state vector.
3. Sophisticated opponent models

Notice that Findler uses monte-carlo simulations instead of heuristics to evaluate hand strength. This was a move that capitalizes on the computer's strengths.

The following equation was used to determine betting behavior for some of Findler's agents:

$$RH = \frac{TABLEPOT}{LIVE * (RAISECOUNT + 1) * FOLLOWERS * RAISE}$$

where

- TABLEPOT = the size of the pot.
- LIVE = number of active players.
- RAISECOUNT = the number of raises in this betting round.
- FOLLOWERS = the number of players that come after the agent (and thus can raise it).
- RAISE = the amount that the agent needs to add just to stay in the game.

Findler's main contribution was not the poker strategies themselves but the diverse types of poker players and their abilities to model opponents. He also built one of the first game environments, which enabled humans to design and employ automated poker players.

3 Texas Hold 'Em

The origins of poker are much debated, but it has been a popular card game for at least the last century. Today, Texas Hold 'Em has become an extremely popular game, with large, lucrative tournaments played all over the world. A complete description of the game can be found at any number of web sites. As a brief summary, a Hold 'Em game consists of many *hands*, and each hand involves the following steps:

1. The two players to the immediate left of the dealer put in small wagers, known as *blinds*, to insure that each player contributes some money to the pot, even when they have bad cards. **We will not put a lot of emphasis on this aspect of the game in this document, but it is quite important to include it in a simulator**.
2. Two *hole cards* are dealt to each player. These are not visible to the other players. See Figure 4 for an example.
3. A round of betting is performed, with the player to the immediate left of the dealer beginning.
4. 3 shared cards, *the flop*, are dealt face up in the middle of the table. All players can use these, in combination with their hole cards, to make a 5-card poker hand, as shown in Figure 5.
5. A second round of betting is performed.
6. The 4th shared card, *the turn* is dealt face-up and added to the flop. Each player now makes a 5-card poker hand from these 6 cards: the flop, turn and the two hole cards. This is illustrated in Figure 6.
7. A third round of betting is performed.
8. The 5th shared card, *the river* is dealt face-up and added to the flop and turn. Each player now makes a 5-card poker hand from these 7 cards: the flop, turn, river and the two hole cards. Figure 7 illustrates this stage.

9. A final round of betting is performed.

10. If 2 or more players have still not folded, they must show their hole cards. This is often called a *showdown*. The player with the highest-rated hand wins the entire pot. On ties, the pot is split among the winners. If all players except one fold, then that player is **not** required to reveal her hole cards before collecting the pot.

3.1 Betting

During betting, each player has the option to *fold*, *call* or *raise* the current bet. By folding, the player forfeits the current hand, thus losing any money that she has contributed to the pot so far. To call, the player matches the current bet. For example, if the bet for the current betting round is 100, and a player has already contributed 70 during that round, then 30 is needed to call.

When raising, a player contributes **more** than the current bet, thus increasing that bet. This forces all other players to contribute even more money to the pot if they wish to remain in the hand.

At the beginning of each betting round, the slate is wiped clean, the current bet is set to 0, and previous contributions to the pot have no effect upon the new round.

In simulation, it is wise to include a *maximum bet* parameter such that the betting in any given round does not escalate to unreasonable amounts.

The act of *bluffing* in poker involves aggressive betting when one has relatively bad cards. A bluffer will normally raise the bet in hopes of getting all other players to fold. The bluffer knows that if anyone continues to call her through all the betting rounds, she will probably lose. If everyone does fold, the bluffer need not show her hole cards, and thus nobody knows for sure if they have been bluffed. Bluffers are only exposed when they get caught in a showdown.

Good poker players will a) bluff occasionally, or even quite often, b) take note of how often other players bluff, and c) proceed to showdowns (largely by calling, not raising) even with relatively bad cards in order to force opponents to show their hole cards. Good poker players are experts at attaining, remembering, and using information about their opponents playing styles.

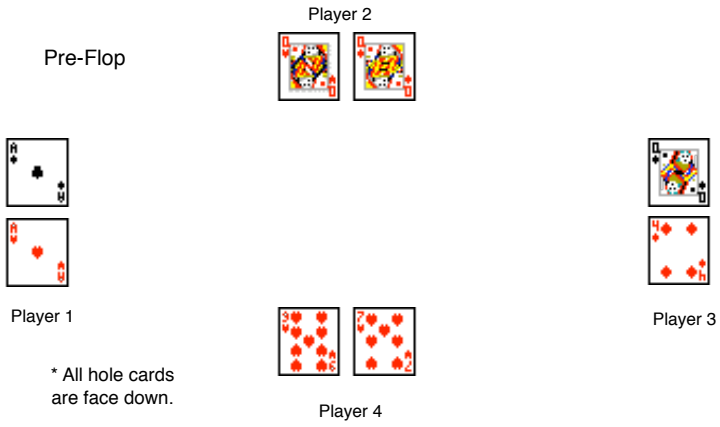

Figure 4: A hand of Texas Hold 'Em prior to the flop. Each player has two private *hole* cards.

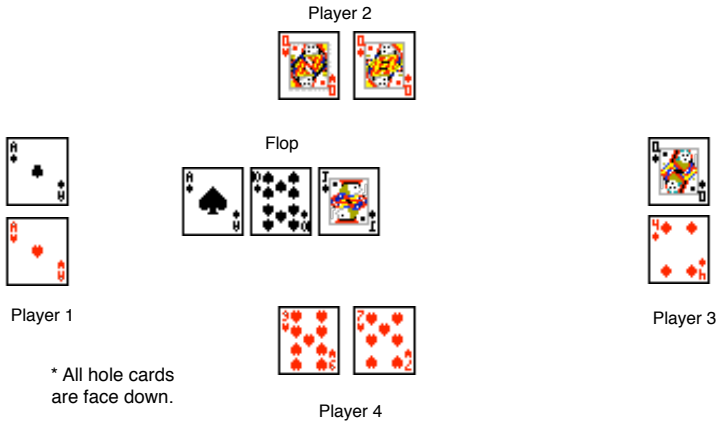

Figure 5: After the *flop* in Texas Hold 'Em. The 3 middle cards (a.k.a. the *flop*) are available to all players.

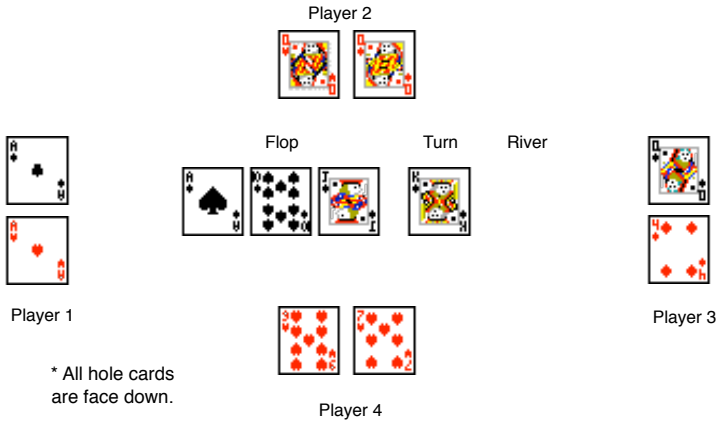

Figure 6: After the *turn* card in Texas Hold 'Em.

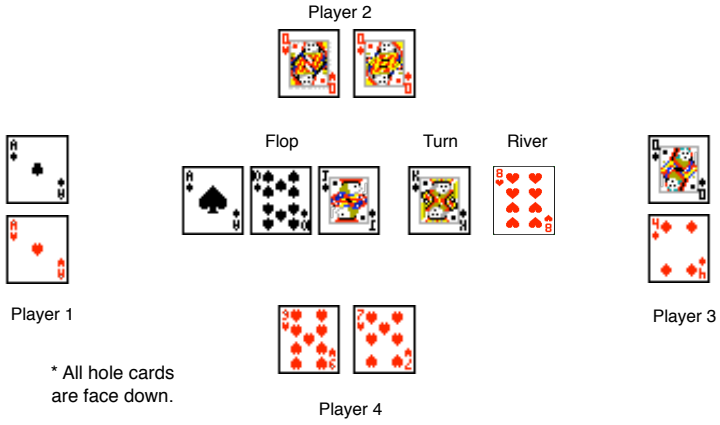

Figure 7: After the *river* card in Texas Hold 'Em. Each player now makes the best 5-card hand out of their hole cards plus the five community cards: flop, turn, river.

3.2 Power Ratings of Poker Hands

There are hundreds, if not thousands, of variations on the card game of poker. However, most share a common ranking of hands:

1. Highest card
2. One pair
3. Two pair
4. 3 of a kind
5. Straight
6. Flush
7. Full House
8. 4 of a kind
9. Straight Flush - with Royal Flush being the highest of these

Many web sites, such as Wikipedia, explain these rankings in detail. Although many poker games, such as Texas Hold 'Em and 7-card stud, give players access to 6, 7 or more cards, it is only the best 5 of those cards that compose the actual **hand**. Hence, all of the standard poker rankings are based on a 5-card hand.

For any automated poker system, a function is needed which takes a group of cards, normally 5, 6 or 7, and finds the best 5-card poker hand within them. It then returns a power rating of some sort.

One suggestion for the power rating is a list consisting of the power index, which is 1 for a hand with nothing (but a high card), 2 for a pair, 3 for 2 pairs,...and 9 for a straight flush. The remaining elements of the list are values used to break ties in cases where, for example, two players have a full house, one with aces and 10's, the other with queens and jacks. The complete power rating should reflect all potential tie-breaking information in the 5-card hand.

Consider the following 2 full-house hands:

1. $(A\diamondsuit, A\heartsuit, A\spadesuit, 10\clubsuit, 10\spadesuit)$
2. $(Q\clubsuit, Q\heartsuit, Q\spadesuit, J\diamondsuit, J\heartsuit)$

For these hands, a rating that covers all 5 cards would be (7 14 10) and (7 12 11), respectively, since the 3 aces (14) and 2 10's (10) are the only 5 cards used. However, for a weaker hand, such as a pair of kings with the next 3 highest cards being 8 6 and 5, the power rating would be longer: (2 13 8 6 5). Conversely, a royal flush would have a simple, but dominating, power rating of (9 14), where the 14 indicates the value of the highest card in the straight, the ace.

The power-rating procedure can make basic assumptions, such as the fact that a single 52-card deck is being used. Thus, for example, no 5-of-a-kinds are possible. If the power rating is used for games without shared cards, then if one player has 3 aces, it's guaranteed that no other player will have 3 aces. Hence, a 3-of-a-kind power rating would only need two numbers: 1) the code for 3-of-a-kind (e.g. 4) plus 2) the value of the triple

(e.g. 14 for an ace). Since 3-of-a-kind aces couldn't possibly be tied (assuming all hands were generated from the same 52-card deck), there is no need for additional numbers. However, in a game with shared cards, such as Texas Hold 'Em, two players could have the same 3-of-a-kind hand, and thus, the tie-breaker information would need to appear in the power rating.

Alternate rating systems are clearly possible, but all should incorporate the standard rules for comparing poker hands.

3.3 Texas Hold 'Em Rollout Simulations

Power ratings give a local classification of a poker hand, but they completely disregard the larger context of the game, such as the cards that opponents possess. At the end of a poker game, we use power ratings to determine who actually wins the pot, but during a poker game, we need to *evaluate* a hand: we need to estimate the probability that it will beat all other opponent' hands.

By the time a hand consists of 5 cards (or more), evaluation is quite straightforward, since the basic power rating of the cards often provides a good indicator of it's strength. However, the power rating does not a) account for the opponents and their possible hands, nor b) give much useful information when a hand consists of only 2 or 3 cards.

Poker books written by experts are chock full of hand evaluations for 2-card hands, many of which are based on their years of poker-playing experience. One of the most influential poker authors is David Sklansky, whose *The Theory of Poker* is considered a classic. In *Hold 'Em Poker*, Sklansky presents his 8 groups for 2-card hands, with group 1 including super hands such as a pair of aces and group 8 capturing relatively weak hands that nonetheless have some potential, such as a 2 and 3 of the same suit. Hands that fail to make a category are those that a player would normally fold, unless they are dead set on bluffing.

With the advent of increased computing power, it is now feasible to test the winning potential of a hand, H*, by simply fixing its 2 cards and dealing out the remaining 50 cards (to fictitious players) many times, and thus in many ways. The system can then simply count the number of times that H* wins to get a reasonable estimate of it's potential. This is called *rollout simulation*, and it can be performed from any point in a poker game. For example, in Figure 8, the flop and turn have already appeared, but the river has not. Thus, from the perspective of player 1, each card with a question mark denotes one that is unknown and can thus be instantiated in many ways, using all but the 6 cards visible to player 1.

3.4 Pre-Flop Rollouts

The possible number of hole-card pairs is:

$$\binom{52}{2} = 1326$$

For each of these combinations, we can do many (R) rollout simulations to get an idea of how often that combination turns out to be the best hand in a K-player game. For these simulations, **we assume that every player remains until the showdown**, which is highly unrealistic, but nonetheless, a useful starting point.

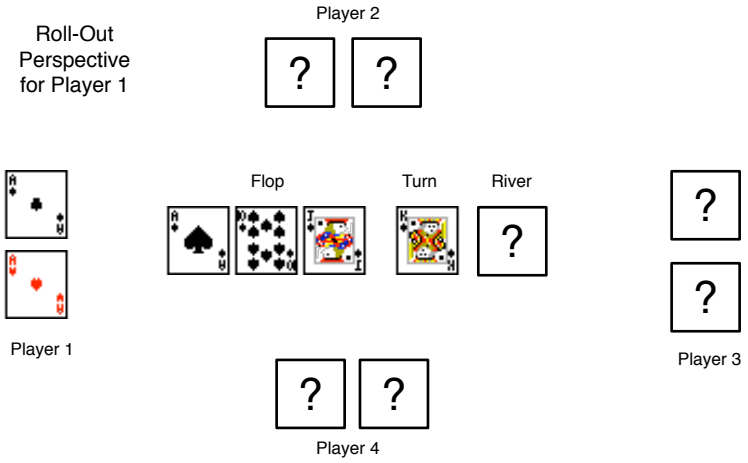

Figure 8: Rollout simulation from the perspective of Player 1. Cards with question marks are those that need to be randomly filled (many times in many ways) during the rollout simulation for player 1.

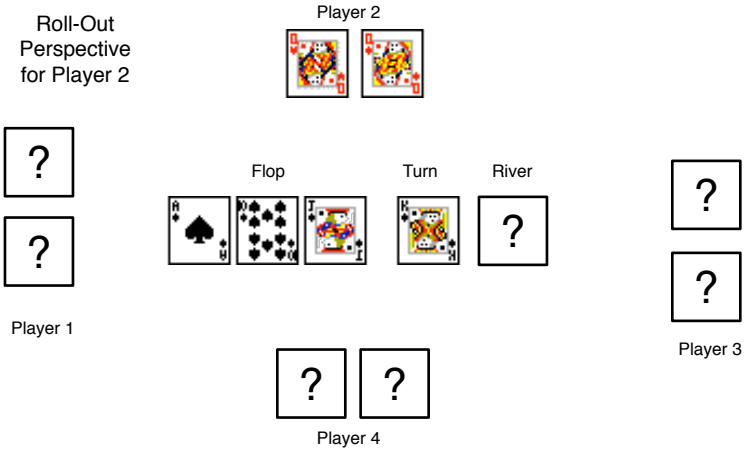

Figure 9: Rollout simulation from the perspective of Player 2. Cards with question marks are those that need to be randomly filled (many times in many ways) during the rollout simulations for player 2.

That entails $1326 * R$ rollout simulations, and it should be done for all realistic game sizes: from 2 to 10 players. Since R needs to be in the thousands or tens (or hundreds) of thousands to get useful estimates, this can be too computationally demanding to perform on-line (i.e. during a poker game).

3.4.1 Simplifying Pre-Flop Rollouts via Equivalence Classes

We can reduce the computational load of pre-flop rollout simulation by noticing that many 2-card combinations are equivalent in terms of their potential strength. For example, if the cards do not share the same value nor suit, we get a 12-member equivalence class (for the 3-and-queen pair) with combinations such as:

$$(3\spadesuit, Q\clubsuit) = (3\clubsuit, Q\diamondsuit) = (3\diamondsuit, Q\clubsuit) = \ldots$$

A similar set of 12 combinations exists for any pair of cards with unequal suit and value.

Figure 10 shows the basis for a rollout simulation for a member of this type of equivalence class.

If the cards are of different value but the same suit, then an equivalence class of size 4 is formed for each pair of values. For example:

$$(9\spadesuit, 10\spadesuit) = (9\clubsuit, 10\clubsuit) = (9\diamondsuit, 10\diamondsuit) = (9\heartsuit, 10\heartsuit)$$

since no suit dominates another in most poker games. Figure 11 shows the basis for a rollout simulation for a member of this type of equivalence class.

Finally, for any pair of cards of the same type, such as kings:

$$(K\spadesuit, K\clubsuit) = (K\clubsuit, K\heartsuit) = (K\diamondsuit, K\heartsuit) = \ldots$$

Thus, each pair is a member of a 6-element equivalence class. Figure 11 shows the basis for a rollout simulation for a member of this type of equivalence class.

Now, we only need to perform a rollout simulation for one member of each equivalence class. All other members will inherit the same winning probability. So instead of performing 1326 rollout simulations (for each of the K= 2 to 10 table sizes), we only need to perform:

$$\binom{13}{2} + \binom{13}{2} + \binom{13}{1} = 78 + 78 + 13 = 169$$

where each of the 3 terms, from left to right, denotes the number of equivalence classes for:

1. unpaired, mixed-suit (a.k.a. *unsuited*) combinations.
2. unpaired, same-suit (a.k.a. *suited*) combinations.
3. pairs.

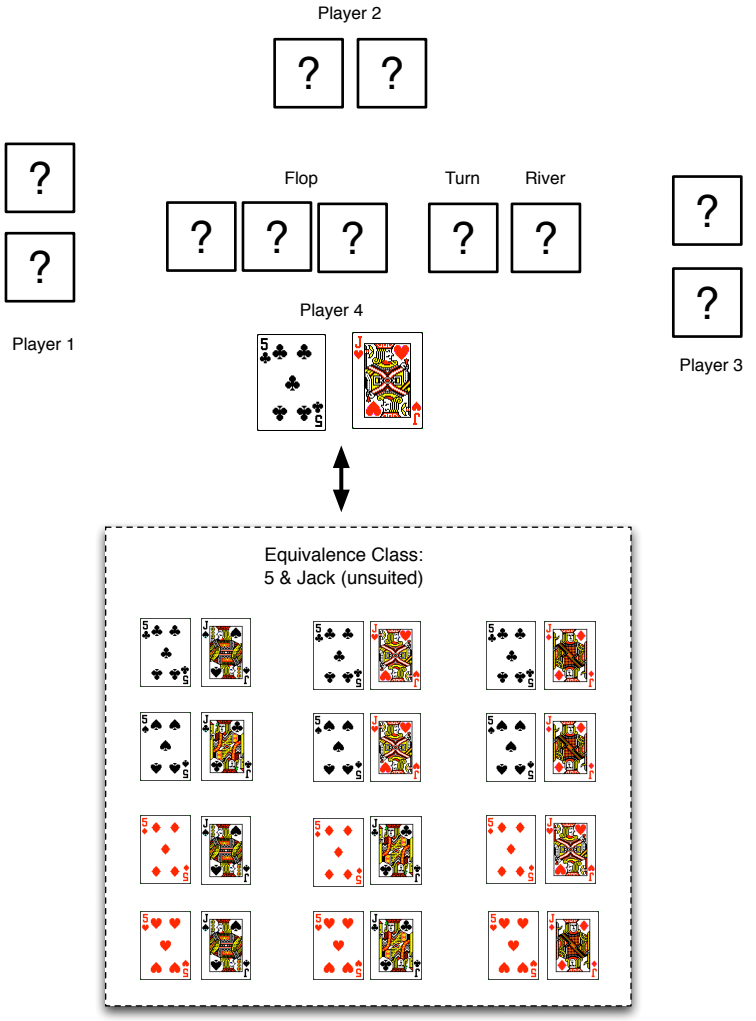

Figure 10: Example of an unsuited, unpaired set of hole cards and its equivalence class of size 12 for pre-flop rollout simulation.

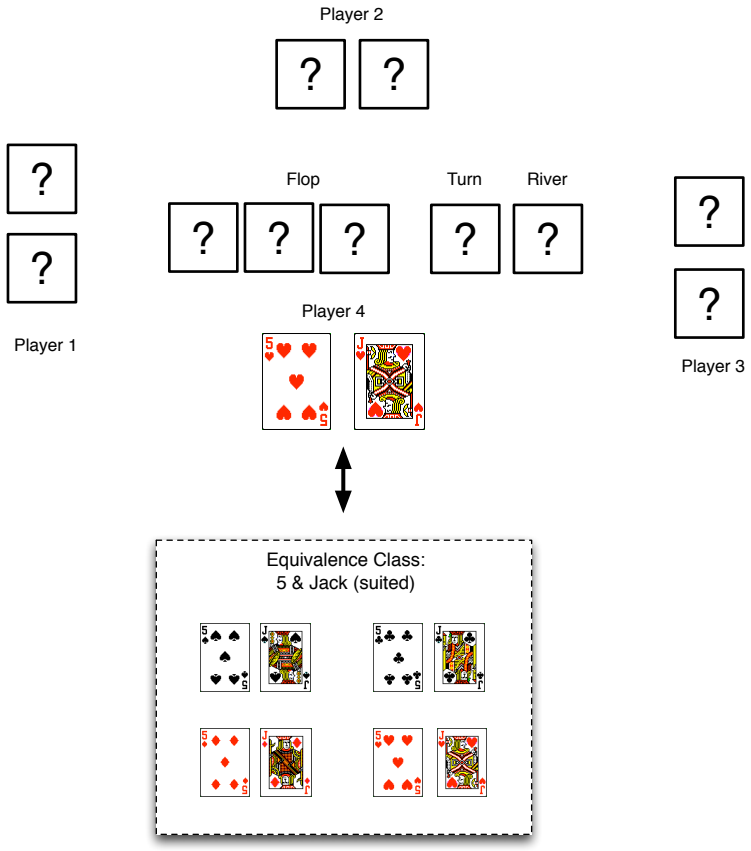

Figure 11: Example of a suited set of hole cards and its equivalence class of size 4 for pre-flop rollout simulation.

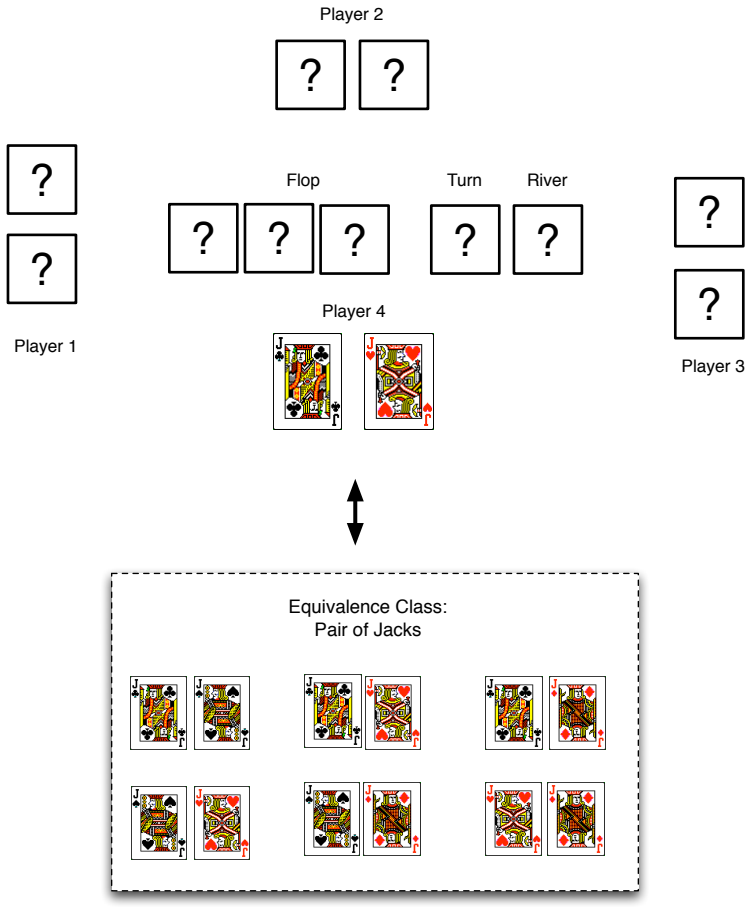

Figure 12: Example of a paired set of hole cards and its equivalence class of size 6 for pre-flop rollout simulation.

3.4.2 Pre-Flop Probability Tables

For each of the 169 equivalence classes, we can perform R rollout simulations and record the results in a table. This can be done off-line such that the complete table is available to any automated (or human) Hold 'Em player. The production of a table entry for one equivalence class in a 4-player game is illustrated in Figure 13. The values in these tables represent probabilities that hole-card pairs in the equivalence class will win a k-player game.

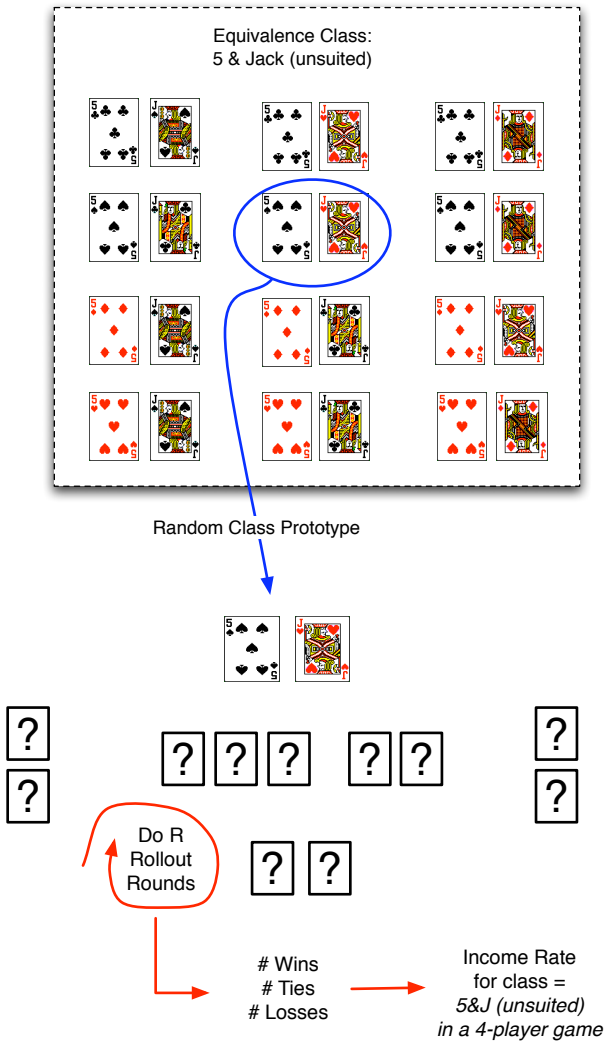

Figure 13: The general procedure for producing an entry in the pre-flop probability table via R rollout simulations. It does not matter which member of the equivalence class is chosen, nor whether the same or different members are used during the R rollouts.

The table can be a data structure such as that depicted in Figure 14.

Once generated, the table can be used by any player, human or artificial, by simply converting her (its) hole cards into the appropriate equivalence class and then finding that class's entry in the table. For example, the hole cards $(3\diamondsuit, 4\diamondsuit)$ represents the equivalence class *3-4 suited*, whose entry the system then finds for the appropriate table size (i.e. number of players).

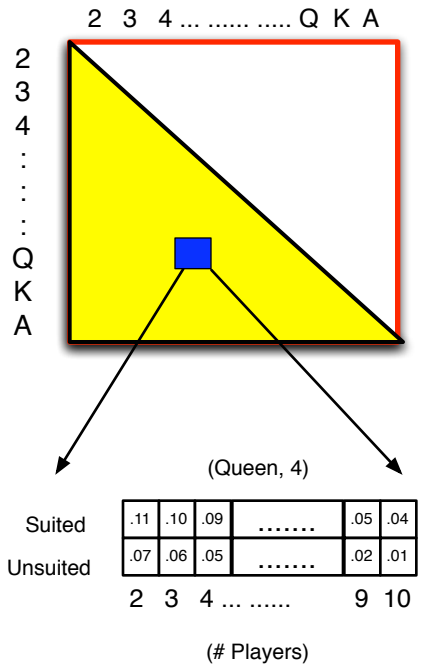

Figure 14: Basic data structure for pre-flop, rollout simulation results, which are to be used as winning probabilities during the pre-flop stage of a Texas Hold 'Em round.

3.5 Hand Strength

Hand strength is a basic measure of the winning potential of the focal player's hole cards, H, in combination with the currently available shared (a.k.a. *community*) cards, S. As depicted in Figure 15, to compute hand-strength, we merely remove H and S from the available cards. After the flop, that leaves 47 possible cards, and thus $\binom{47}{2} = 1081$ hole-card pairs. Each such pair is combined with S to see if it yields a better 5-card hand than H+S. The numbers of wins, ties, and losses for H versus each of these 1081 pairs are recorded, and the hand strength of H is calculated as:

$$\left(\frac{Wins + \frac{Ties}{2}}{Wins + Ties + Losses} \right)^k$$

where k is the number of opponents who have not folded, and the denominator is simply 1081 for the post-flop hand strength calculation. The number of active opponents needs to be considered in this probability calculation, since **each** opponent needs to be beaten in order for the focal player to win.

For post-turn and post-river calculations of hand strength, the extra shared cards are simply removed from the set of available cards such that $\binom{46}{2} = 1035$ hole-card combinations are used to evaluate post-turn hand strength, and $\binom{45}{2} = 990$ are generated to compute the post-river value.

For this simple measure of hand strength, the unknown Turn and/or River cards are merely ignored. In more computationally-intensive versions, possible values for these missing shared cards are also considered. But this pushes the number of computations to over a million for the post-flop stage. We ignore these advanced

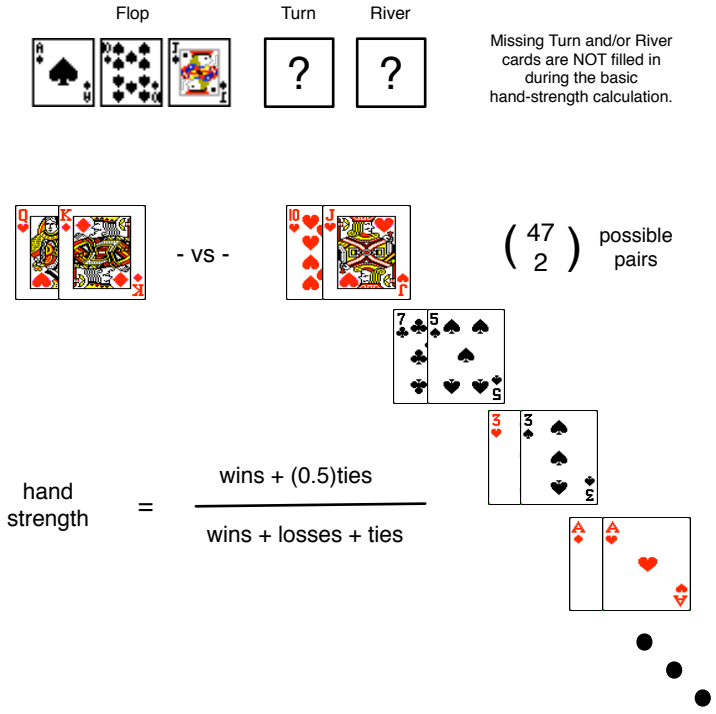

Figure 15: Basic calculation of hand strength for $(Q\heartsuit, K\diamondsuit)$ when the flop is $(A\spadesuit, 10\spadesuit, J\clubsuit)$. A similar calculation, with $\binom{46}{2}$ and $\binom{45}{2}$ possible hole-card pairs can be performed after the Turn and River stages, respectively.

hand-strength estimates in this discussion.

Figure 16 provides 5 examples of hole cards and their hand strengths relative to the flop ($A\spadesuit, 2\spadesuit, 3\spadesuit$). Notice that the first 4 pairs are very strong, but only the second, ($4\spadesuit, 5\spadesuit$), is unbeatable.

However, if the turn card comes up $A\clubsuit$, notice how the hand strengths change (as shown in Figure 17. Now, the first pair yields 4 aces, which is only beaten by the second hand (a straight flush) but is no longer beaten by the normal straight produced by the third hole-card pair (whose hand strength decreased after the turn card appeared). Note also that the 4th hole-card pair increased in strength, since the turn card gave it a full house (aces and 3's).

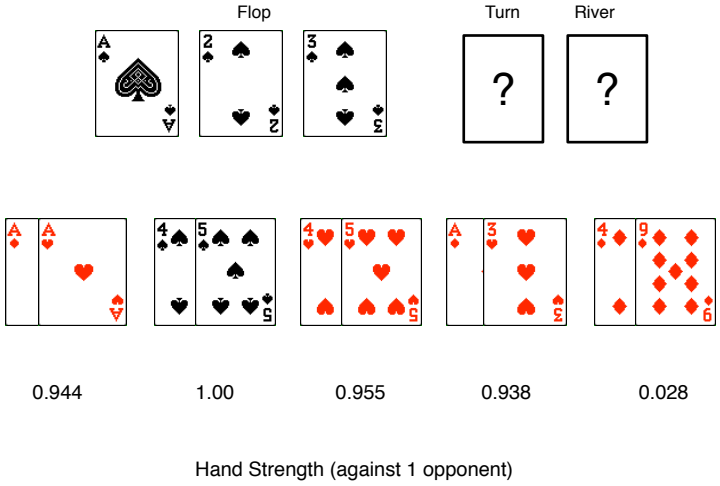

Figure 16: Several examples of hole cards and their post-flop hand strengths for a given flop, assuming that only one opponent has not folded.

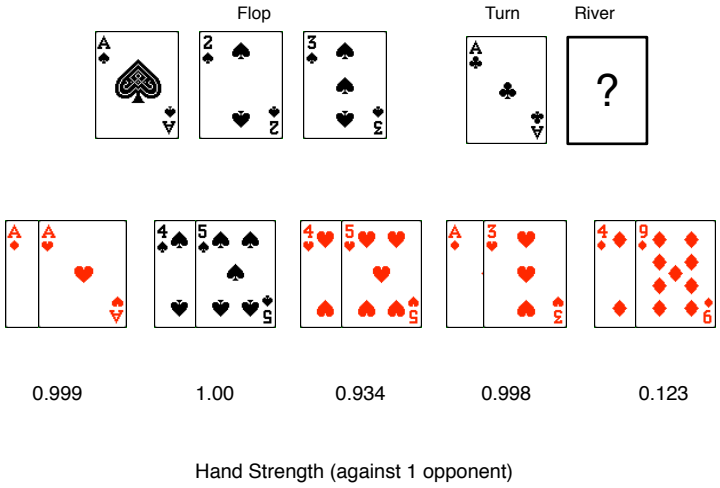

Figure 17: Several examples of hole cards and their post-turn hand strengths for a given flop and turn, assuming that only one opponent has not folded.

The combination of pre-computed pre-flop winning probabilities and on-the-fly hand-strength calculations enable a poker-bot to evaluate its hand at any point in a Hold 'Em contest. Prior to the flop, the hand

contains only 2 cards and is thus best evaluated by the pre-flop tables. After the flop, the hand consists of 5 cards and can be assessed for hand strength, which can also be calculated after the Turn and River cards. Since the basic hand-strength calculations outlined above only involve a comparison of approximately 1000 hands, they are easily done on-line on most modern laptops.

3.6 Opponent Modeling

By observing opponents over the course of a long game (i.e. many hands), a player can compile statistics that might form the basis for general rules of the form:

> **Context:** When in the post-flop betting round with 1 raise in the current round, 3 or more players remaining, and pot odds of 0.25. **Action:** player 3 often calls even when his hand is very good.

These rules are potentially useful for human players, but the statistics needed to form them probably exceed the memory capacity of most adult brains. However, an automated poker player can log reams of relevant data about its opponents and then use it to form high-utility behavioral rules.

One approach, described below, links the context and action to the hand strengths of an opponent's cards such that, in the future, the system can compute an expected hand strength for that opponent based on a recently-observed context and action.

So far, we have used the following criteria for assessing a player's standing in a hand of poker:

1. Power rating - provides a simple, local quantification of the value of a poker hand.
2. Pre-flop winning probability - estimates the chances that a 2-card hand will be filled out to become the best hand in a k-player game.
3. Hand strength - computes the probability that a hole-card pair combined with 3, 4 or 5 shared cards will beat any other hole-card pair (combined with the same shared cards).

These are all useful, but they ignore some of the most important information in a poker game: the betting behaviors of opponents. More specifically, the relationships between an opponent's hand strength, the context of the game, and his betting behavior. While it's useful to know that player X raises a lot, it's much more useful to know that player X raises a lot even when he has poor cards (i.e., that player X bluffs a lot). And its even more useful to know that player X only does so after the flop. In short, the more that I know about what player X does *when he has poor cards*, the better I am able to predict **when** he has poor cards. Opponent models consist of structured information that allow us to make such predictions with reasonable accuracy. The models considered below link game contexts, player actions, and player hand strengths such that the latter can be estimated from the former two.

Consider the game scenario depicted in Figure 18. At this point in the game, player 2 has already folded, while players 1, 3 and 4 remain. The system is playing from the perspective of player 1, so his cards are visible.

Player 1 begins the round by betting. This bet happens in a particular game *context* (denoted by C1), which captures various salient aspects of the current game situation (as discussed more thoroughly below). Since

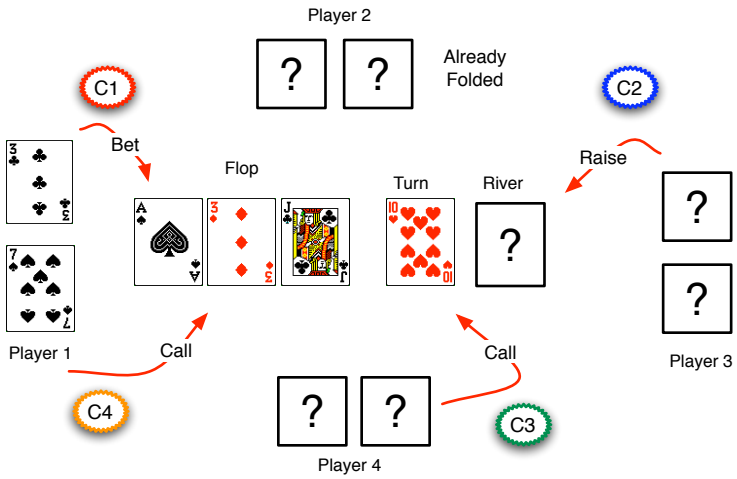

Figure 18: C1-C4 denote action contexts.

each player's action can potentially change the context - depending upon how a context is defined - player 3 makes his decision (to raise) in context 2 (C2). Player 4 then calls while in context C3. Player 1, now in context C4, decides to call, and the post-turn betting round ends.

At this point in the game, the computer has logged context-action pairs for each player. For those players who make it to the showdown, this information will prove useful (in the models of those players).

Figure 19 portrays the post-river betting round, which begins with player 1 folding in context C5. Player 3 then bets out of context C6, and player 4 calls out of C7. The final betting round ends with players 3 and 4 still active, and thus a showdown is needed to determine the winner. These 2 players then reveal their hole cards, as shown in Figure 20, and player 4's 3-of-a-kind 10's beats player 3's 2-pair (aces and jacks).

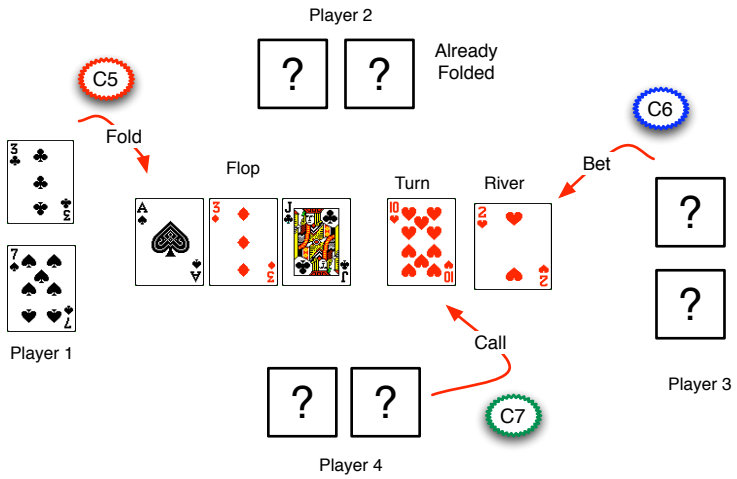

Figure 19: C5-C7 denote action contexts.

At this point, the important information for the opponent modeler is not the winner, but the hole cards of players 3 and 4. Knowing these cards, the system can go back in time to the pre-flop, flop, turn and

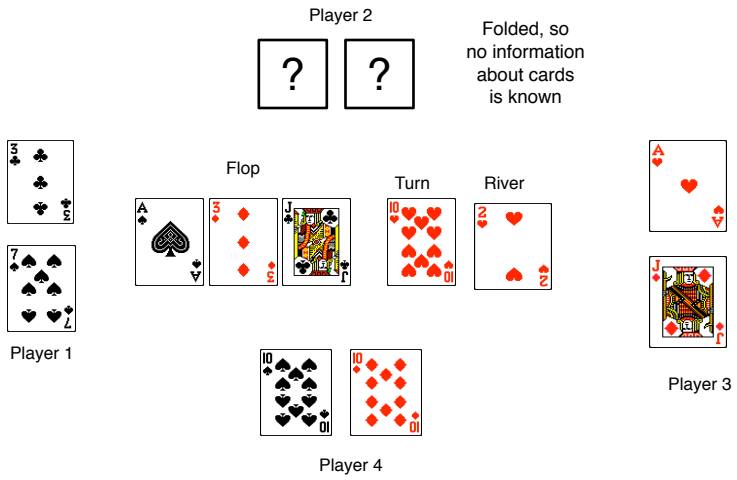

Figure 20: At hand's end, player 1 sees the cards of players 3 and 4 and can thus update her model of each.

river stages of the game. At each point, it can calculate the hand strength with which players 3 and 4 were operating. For the pre-flop stage, the hand-strength is equated with the winning probability from the pre-flop rollout simulations, while in the other 3 stages (post-flop, post-turn, and post-river), the standard hand-strength calculations apply. Since it already knows the contexts and actions for those time points, it can now link contexts, actions and hand strengths into useful associations, as shown in the tables of Figures 21 and 22.

Context	Action	Hole Cards	Shared Cards	Hand Strength
C2	Bet/Raise	$(A\heartsuit, J\diamondsuit)$	$(A\spadesuit, 3\diamondsuit, J\clubsuit, 10\heartsuit)$	**0.950**
C6	Bet/Raise	$(A\heartsuit, J\diamondsuit)$	$(A\spadesuit, 3\diamondsuit, J\clubsuit, 10\heartsuit, 2\heartsuit)$	**0.911**

Figure 21: Information for the model of player 3.

Context	Action	Hole Cards	Shared Cards	Hand Strength
C3	Call	$(10\spadesuit, 10\diamondsuit)$	$(A\spadesuit, 3\diamondsuit, J\clubsuit, 10\heartsuit)$	**0.958**
C7	Call	$(10\spadesuit, 10\diamondsuit)$	$(A\spadesuit, 3\diamondsuit, J\clubsuit, 10\heartsuit, 2\heartsuit)$	**0. 925**

Figure 22: Information for the model of player 4.

As long as the contexts are properly defined (as described below), the context-action pairs for a given player, P, will reoccur in the course of several hundred (or thousand) games. Each time they do, if P makes it to a showdown, then his cards will be revealed and more card-strength information will become associated with the context-action pair (C,A). Thus, over time, table entries such as those in Figure 24 will arise, with many hand strengths associated with (C,A). The average of those strengths will then give an indication of the hand-strength to **expect** of P when he performs action A in context C.

As shown in Figure 25, model tables can serve as useful predictors of opponent hand-strength. In this example, player 1 uses the contexts and associated actions for players 2, 3 and 4 as inputs to their respective player models. Each model outputs an estimated hand strength for that player, and these allow player 1 to make an informed decision as whether to fold, call or raise.

Context	Action	Hole Cards	Shared Cards	Hand Strength
C1	Bet/Raise	(3♣, 7♠)	(A♠, 3♢, J♣, 10♡)	**0.294**
C4	Call	(3♣, 7♠)	(A♠, 3♢, J♣, 10♡)	**0.294**
C5	Fold	(3♣, 7♠)	(A♠, 3♢, J♣, 10♡, 2♡)	**0. 259**

Figure 23: Information about player 1's behavior that nobody (except player 1) will know, since player 1 folded and thus never revealed his cards in the game of Figures 18 - 20.

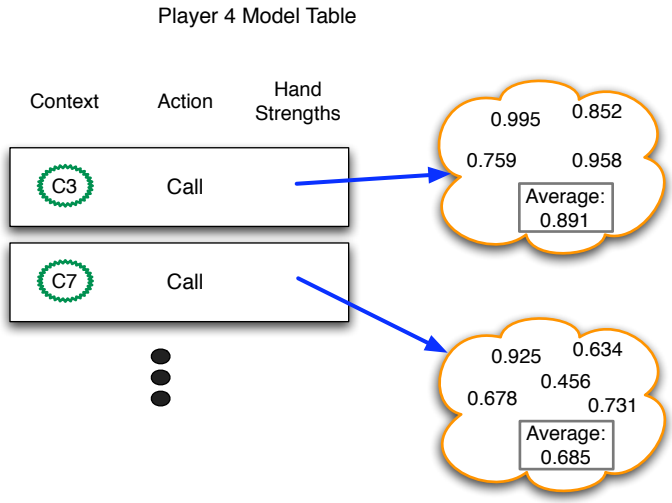

Figure 24: The sketch of a hypothetical model table for player 4 after many rounds of play.

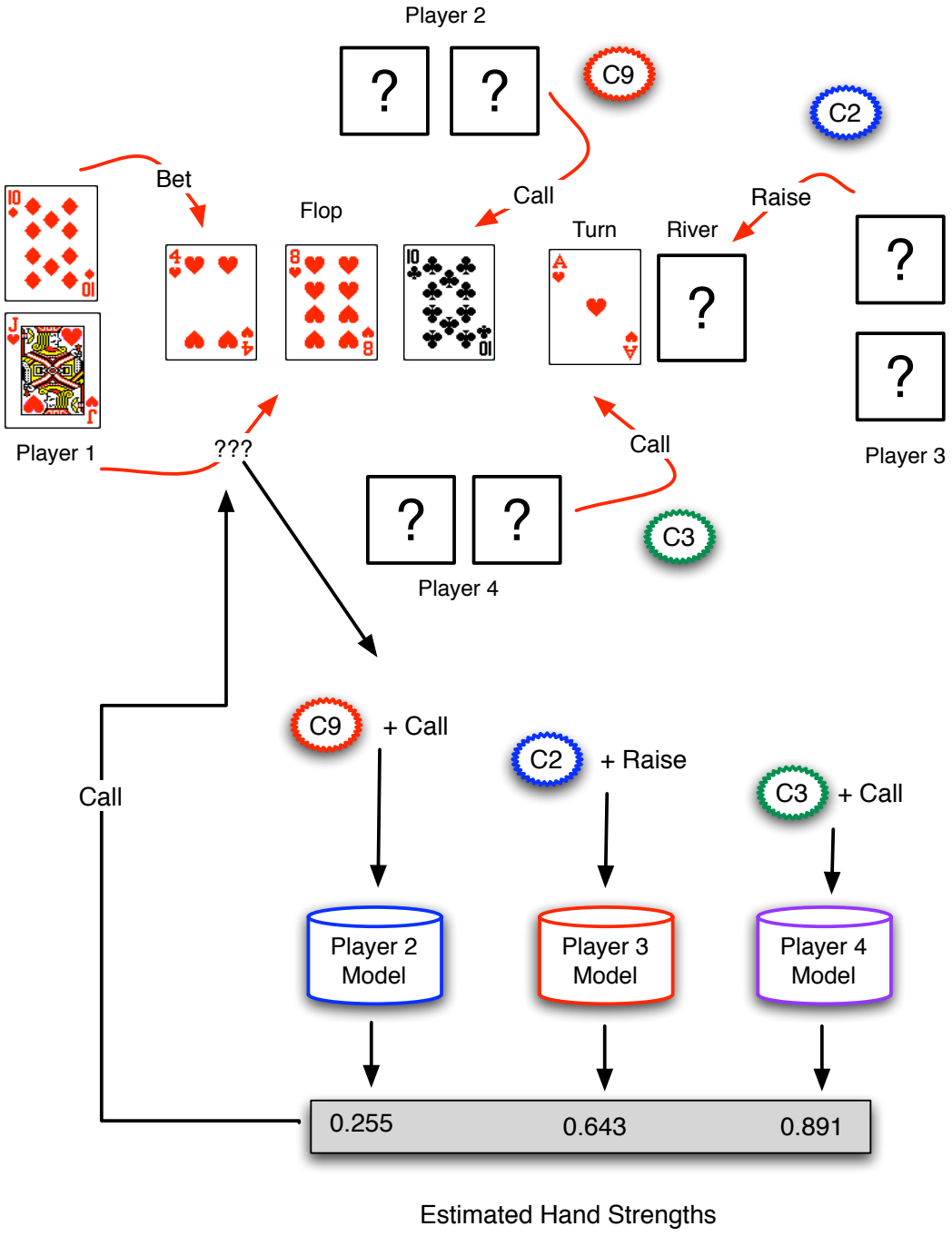

Figure 25: Illustration of the use of opponent models to estimate the hand strengths of the remaining players. These estimates then help player 1 decide to call player 3's raise.

3.7 Game Contexts

One pivotal aspect of the opponent models discussed above are *contexts*, which can be defined in many different ways, depending upon the features of a poker game that the modeler deems most salient.

Useful features include:

1. The betting round, whether (1) pre-flop, (2) post-flop, (3) post-turn or (4) post-river.
2. The number of players remaining in the hand.
3. The number of raises in the current betting round.
4. The *pot odds* = C:P, where:

   C = the amount needed to call the current bet, and
   P = the size of the pot (i.e., the total amount bet so far in the current hand).

   Expressed as a fraction, the pot odds are:

$$\frac{C}{C+P}$$

As long as a) the contexts are defined using factors with discrete values, such as the betting round, which has just 4 values, and b) the action set is discrete (i.e. fold, call, bet/raise), then over the course of many hands, context-action pairs will reoccur. This will give many opportunities to observe the hand strengths associated with a particular context-action pair, and given enough trials, the averages of those hand strengths should have some statistical significance. This enables the automated poker player to estimate hand strength based on an observed context-action pair.

Discretization of a factor such as pot odds is easily done by creating a small set of bins for the fractional odds, F, such as:

Bin 1: $F \leq 0.1$

Bin 2: $0.1 < F \leq 0.2$

Bin 3: $0.2 < F \leq 0.3$

Bin 4: $F > 0.3$

Figure 26 illustrates the problem of poorly-defined contexts. If they are too specific, they only pertain to a few situations and thus only yield a few data points (i.e., a few hand-strength values). Conversely, an over-general context will apply to too many situations, producing uninteresting and/or misleading average hand-strength values. Notice that the context in the upper table entry was generalized by both a) dropping a factor (e.g. number of remaining players), and b) widening the target range of variables (e.g. pot odds and number of raises).

4 Wrap Up

You now have a set of basic tools for building an automated poker player:

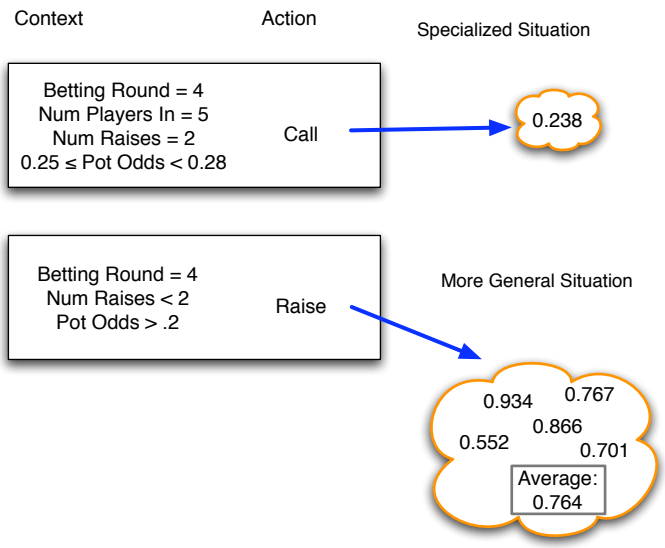

Figure 26: The number of factors involved in a context, and their level of granularity, affect the number of times that the context actually occurs, and hence the number of data samples acquired within a given time period. The upper table entry has a context that is overspecialized due too many specific factors, so it applies in very few cases. Hence, very few associated hand strengths have been accumulated. Conversely, the lower table entry has a much more general context, which applies to more situations and thus ropes in several hand-strength values.

1. Power ratings of hands - to indicate the presence of the important poker hands, such as full houses and straights, and to determine the winner of a showdown.
2. Pre-flop rollout probabilities - to estimate how well a given pair of hole cards will fare.
3. Hand-strength values - to indicate the success of a pair of hole cards plus the available community cards against all other possible combinations of hole cards.
4. Opponent models - to estimate an opponent's current hand strength (or pre-flop rollout probability) based on her previous behavior, i.e., on the cached relationships between her bet/call/raise actions and her hand strength (rollout probability).

The success of a poker bot based on these tools depends upon factors such as the number of rollout rounds used to calculate hand strength (which is often determined by your computer's speed) and the wise choice of contexts for opponent modeling. Since pre-flop rollouts are done off-line, it's worth the investment to run many rollouts, say 10,000, to get accurate table values. Power ratings, although not trivial to compute, are reasonably straightforward values that should not vary much (if at all) between poker bots, since all must reflect the standard classification and ranking of poker hands.

Beyond this, a great deal of poker-bot success will depend upon the heuristic rules that incorporate the information provided by the above four tools. For example, how do you use the estimated hand strengths of your 4 opponents (from the opponent models) to determine your current action? Like Waterman, you may decide to set up a small rule base for these decisions, and to modify various thresholds in those rules over time, i.e. to do simple machine learning.

At any rate, you have now crossed over from the science (and mathematics) to the art of poker playing, and from here, you make the call. Have fun!!