B3W
Pokerbots Strategy Report

JAMES PERAIRE
RAY WANG
CHRIS SWEENEY

Intro

Our bot, B3W, ultimately uses the well-known investing and gambling criterion known as the Kelley criterion, developed by John Kelley in 1956 at AT&T Bell Labs. The Kelley criterion calculates the optimal bet for a situation in which we know the probability of winning (the pot equity) and the net odds of the pot.

We created two iterations of our bot – one that tracked other players' playing styles and adapted our bot's strategy accordingly. However, we submitted another version of our bot that plays strongly for value, calling and folding intelligently based on our equity and stack size.

Bot v1
Statistics

In addition to the standard percentages of calls, checks, raises, and folds, we track the following statistics for each player:

- AGGRESSIONFACTOR
- AGGROFREQ
- VPIP
- PER_VPIP
- PFR

- PER_PFR
- WTSD
- PER_WTSD
- WMSD
- PER_WMSD

Where VPIP is the preflop stat “voluntarily put in pot,” PFR is “pre-flop raise,” WTSD is “went to showdown,” WMSD is “won money at showdown.”

These variables combined give us a sense of how our opponent is playing. Using benchmark statistics, we conclude, for example, that a player whose VPIP > 50 and PFR > 2/3 *VPIP is playing LAG, so we increase our aggression factor.

Alone, VPIP would not tell us much. If we detect a VPIP of, say, under 15, we would determine that our opponent is tight, but we need to look at the PFR to see if the bot’s just a calling station or if it’s being more aggressive based on its position, signalling that the bot is TAG.

The aggression frequency and aggression factor can detect when a bot is playing very passively (both would be < 30-40) or very aggressively (both would be > 75). Aggression frequency treats calls, folds, or checks as passive events and converges faster than aggression factor.

We have benchmark statistics for our bot, called play style parameters:
- AGGRESSION
- MIN_ODDS_CALL
- MIN_ODDS_RAISE
- EQUITY_MULT
- RAISE_MULT

AGGRESSION is a multiplying factor that determines the aggressiveness of our play.
MIN_ODDS_CALL is the minimum equity of winning for us to play the hand, determining our looseness.
EQUITY_MULT is a multiplying factor less than 1 that compensates for the fact that we often overestimate our equity when playing good bots.
RAISE_MULT is a multiplying factor greater than 1 that increases our Kelley bet if we have a particularly good hand.

Adaptation

Our benchmarks are continually updated throughout the match as we collect data on other players. They are weighted such that earlier values are considered more heavily than recent values. This allows us to change strategy if the other bot is also changing its play over the long term.

Kelley Criterion

The formula for the Kelley criterion is

$$\frac{p * (b + 1) - 1}{b}$$

where p is the pot equity (calculated using the pbots_calc library) and b is the net odds of the pot. The naiive Kelley Criterion would suggest that we never raise, as if we do not add anything to the pot, our odds would be infinity. To compensate for this, however, instead of using the current pot size to calculate the odds, we wrote a simple function that calculated the predicted pot for a certain raise, and used that.

The Kelley formula gives a factor that is then multiplied by our stack size to produce the optimal betting amount.

If our equity is sufficiently high enough that we want to raise, we will raise the Kelley-optimal amount (see bot v2).

Bot v2:

The first version of our bot performed poorly in testing and did not call optimally, although the adaptation parameters updated correctly.

Our second iteration had more robust equity tests but did not incorporate the tracking methods of the other bot. Where the Kelley bet falls short is the fact that, when our stack drops very low (< 100), our bets and raises are too small to extract any value. To compensate, if our stack is lower than 150, we calculate the Kelley bet for a stack size of 150.

We also saw the problem that we would call too aggressively and lose huge pots. We included a strategic folding method that folds if our odds are too small and the amount needed to stay in the hand is too large; in short, we know when it is not profitable to continue playing. This significantly increased the hands we won when we continued playing.

Room for improvement

To create a better bot, we could have combined the strategy-tracking with the optimal calling and folding. Such a bot would be able to compete with bots that also size bets very well.

Overall, the Kelley bet is a highly predictable strategy that can be exploited by good bots. We could have implemented a system that would bluff or play inoptimally for certain lengths of time to deceive better bots.

But Kelley betting plays strongly for value, and is able to do well against weaker bots that are not able to predict our ranges.

We could have also paid more attention to positions, which are very important in three-player tables. Strong bots would change strategy based on whether they are on the button, SB, or BB, and we do not account for that.

The current predicted pot size is also a rough estimate. We could have improved the predicted pot size by iterating over the optimal wage. That is, we would first decide to raise an arbitrary amount, find the predicted pot size for that amount, and then extract a “new” bet size. For a few iterations we would keep doing this until the “new” bet size converged to the optimal bet. We ended up not implementing this due to concerns that the bet may not converge or take too long to converge, and thus take up too much time. Additionally, the simple Kelley bet was already modified by certain playing parameters so we figured that if our predicted optimal bet was too high or too low, the other parameters would be able to adjust for that.