"Easy" Klondike Solitaire Playing: A Reinforcement Learning Approach

Alexander Stroud

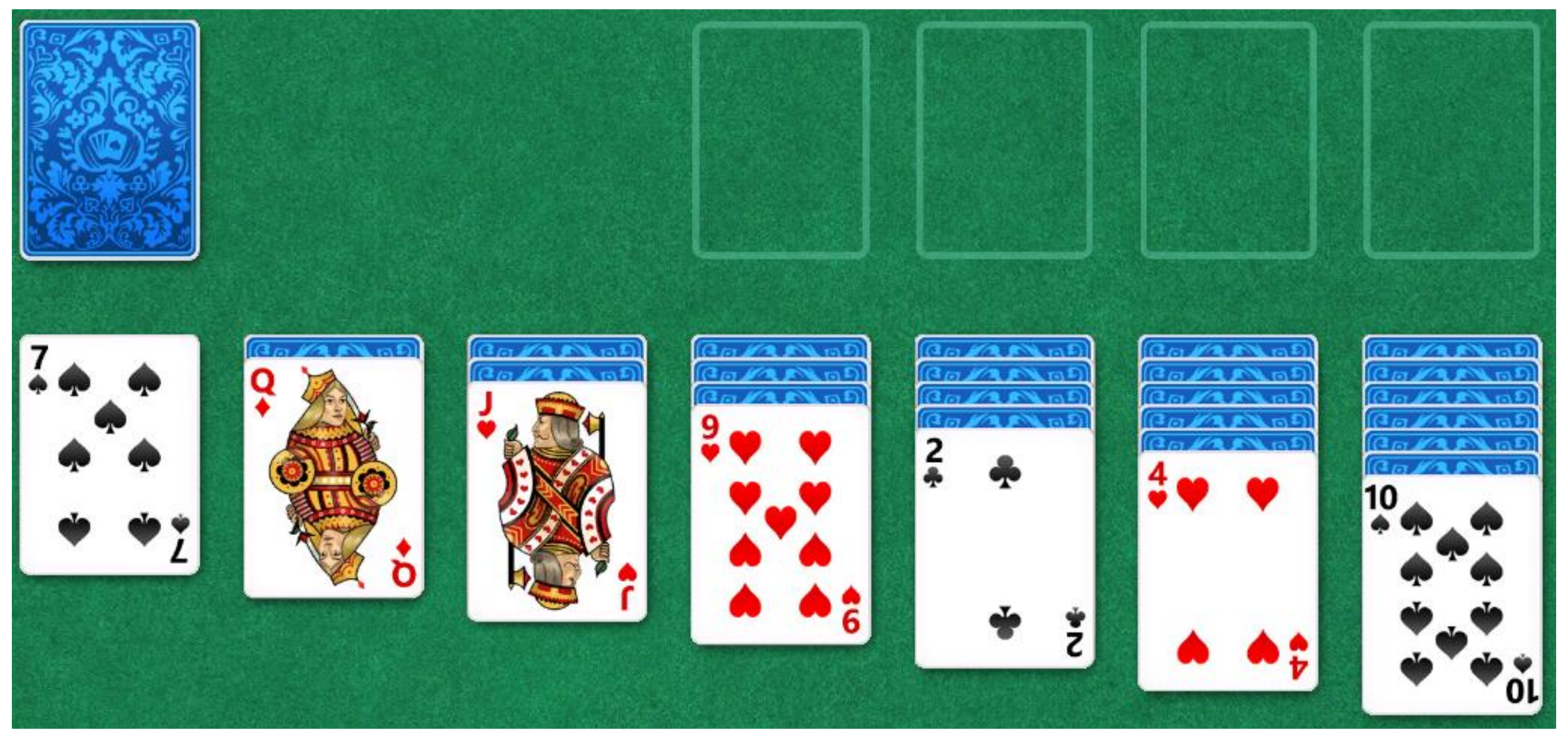
astroud@stanford.edu

Stanford

Problem Statement

Game: Klondike Solitaire
- Classic card game, see right
- Object: move all 52 cards from the deck and build piles to the four suit piles, stacking from low to high card value (Ace, 2, up to King).
- In "Easy" Klondike, one card at a time is flipped from the deck instead of three.

Question:
- Before the initial deal, what is the probability of winning the game?
- Secondarily, given the initial deal, what is the probability of winning the game?



Methods and Formulation

Solitaire can be easily formulated as a Markov Decision Process – the mechanics of the game are completely known
States have three main aspects:
- Suit Piles: the four suit piles are tracked by the number of cards in each pile.
- Build Piles: the ordered list of cards, and the number of unknown cards, in each of the seven build piles.
- Deck: the ordered list of deck cards, and an index indicating the next available card in the deck

Finally, states include a flag indicating the number of times the deck has been cycled through, to identify loss states.
A possible opening state corresponding to the above image is diagrammed below.

Suit Piles: (0,0,0,0)
Build Piles: (((7♠), 0), ((Q♦), 1), ((J♥), 2), ((9♥), 3), ((2♣), 4), ((4♥), 5), ((10♠), 6))
Deck Pile: ((3♥, A♣, 9♦, …, 6♦), 0)
Flag: 0
Actions: [(Build4, Build7, 1), (Build7, Build3, 1), (Flip Deck)]

Actions are defined as tuples containing the start and end piles (deck, suit, and seven build piles), plus the number of cards moved, when necessary.
Rewards are zero for all moves, except that moves into winning states have reward one.
End States are those where the flag is at least 2, there are no moves remaining, which both correspond to losses, or all of the cards have been moved to suit piles, which is a win.
The **discount factor** is 1, as we only care about winning, not how long it takes to win.
Transition probabilities are 1 for most transitions. The only non-deterministic transitions are those where all the cards are moved from a build pile with one or more unknown cards. In that case, since an unknown card is flipped and becomes face-up, its exact value is set as one of the not-yet-seen card values, with equal probability.

Internally, cards are represented as a single integer from 0 to 51. If we divide the integer by four, the integer quotient corresponds to the card value, and the remainder to the suit.

X mod 4	Suit
0	♣
1	♦
2	♥
3	♠

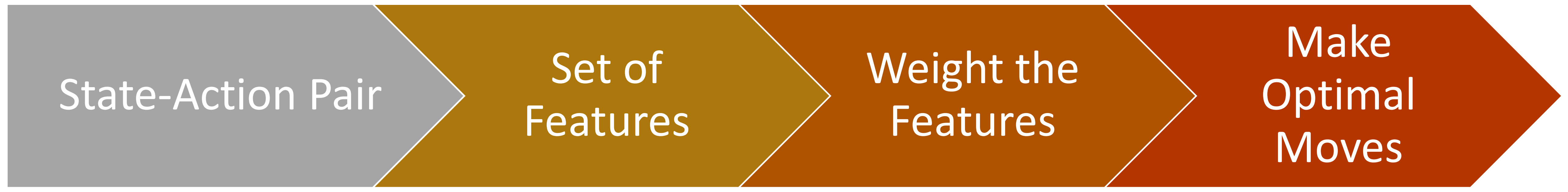
X / 4	Value
0	Ace
1	2
11	Queen
12	King

Card Conversions:
32 = 4 × 8 + 0 → 9♣
17 = 4 × 4 + 1 → 5♦
43 = 4 × 10 + 3 → J♠
6 = 4 × 1 + 2 → 2♥

Approach

MDP details are known – can we use Value Iteration?
- State space explosion renders this approach computationally infeasible
- Would also need to be run once for each possible start state, also infeasible (over 10^24 possible start states!)

Instead, use Q-Learning with feature mapping:



We play many games, choosing actions sometimes randomly, sometimes optimally, and updating our feature weightings with each move.
Features:
- Does the move reveal an unknown card?
- Does the move end in a suit pile?
- What is the greatest number of unknown cards in a build pile after the move?
- Does the move originate from a build pile? from the deck?
- How many unknown cards are there after the move?

We repeat this process four times, across all combinations of two randomness strategies and two reward systems.
Randomness: Moves are made either entirely randomly, or optimally with probability 0.8 and randomly otherwise.
Reward: as described under Methods, or with points given for moving cards to suit piles and revealing unknowns.

Results

Each strategy above was fitted on 5000 games, and then tested on 1000 games, all with random initial deals.

Randomness	Rewards	Fit Win%	Test Win%
Full	0/1	406 / 5000	0 / 1000
0.2	0/1	526 / 5000	0 / 1000
Full	Points	447 / 5000	0 / 1000
0.2	Points	644 / 5000	0 / 1000

Given these results, we conclude that a random player wins between 8 and 9 percent of games, and that the fit performance of the players that try to optimize indicate that a greater success rate is possible.

Error Analysis

Clearly, the player has issues, not winning a single game when instructed to play optimally.

Running several example games, we find that the player makes smart (by human judgement) choices when considering moves from the deck and moves to suit piles. However, since most of the above features have positive scores, the player will always choose a move between two build piles instead of flipping a card from the deck, resulting in an infinite loop as cards are swapped back and forth between two build piles.

This outcome could be altered by increasing the number of features in the problem (starting with the inclusion of deck-related features), as well as more precisely defining loss states to include infinite-loop scenarios.

Also, tuning the learning-rate parameter may help with learning, as in some cases the feature weights have ballooned to unreasonably large values.