Harvard Hold'em Poker Bot
Final Specifications

Prepared by:

Bry Power - bapowers57@gmail.com
Michael Delaney- michaeldelaney@gmail.com
Qi Xiao - xiaoqi.pku@gmail.com
Esty Cohen - aoemom@gmail.com

Signatures/Interfaces

- Classifier
  - Main methods
    - train(x, y): train naive Bayes according to x y, where x = features and y = labels
    - predict(x): perform classification
    - probability (data, action): takes the trained data and the action and returns a dict of all ranks with their corresponding probabilities
    - predict_probability(x): return probability estimate
    - accuracy(x, y): return the mean accuracy on given data and label x, y
  - Internal methods
    - normalize(x, y): compute the posterior probability ensuring that the probability and its inverse add up to 1, while maintaining the ratio between the two, where x = computed probability and y = its inverse
    - prior_probability_hand(hand): return the prior probability of a hand
    - prior_probability_rank(rank): return the prior probability of a rank
- Poker engine
    - poker(hands) : hand
    - hand_rank(hand) : int (representation of a ranking of hand)
    - card_ranks(cards): list (of ranks)
  - Functions:
    - *Deal:* Which allocates x amount of cards for y amount of players.
    - *Straight, Flush, Kind, Two-pair:* These functions to match for strengths

- *Assertion tests:* encapsulates testable logic for all functions we have

Modules/Actual Code

Algorithm Progress:

- Prior probabilities
  - Method returning the prior probability based on the hand.

```
def prior_probability_hand(card1, card2, suitedness = False):
    if (card1 == card2):
        return 0.00452
    elif (suitedness == True):
        return 0.00302
    else:
        return 0.00905
```

  - Method returning prior probability based on the rank.

```
def prior_probability_rank(rank):
    if (rank == 1):
        return 0.0211
    elif (rank == 2):
        return 0.02263
    elif (rank == 3):
        return 0.02565
    elif (rank == 4):
        return 0.03772
    elif (rank == 5):
        return 0.07093
    elif (rank == 6):
        return 0.05129
    elif (rank == 7):
        return 0.07695
    elif (rank == 8):
        return 0.09957
    else (rank == 9):
        return 0.59416
```

*probabilities calculated based on Sklansky hand groups

- Posterior probabilities by rank given a player action, returned as a dictionary with the rank being the key and the probability being the value.

```
def probability(self, trained_probs_by_rank, action):
    ratios = []
    probabilities={}
    for rank in trained_probs_by_rank:
        prior_probability = self.prior_probability_rank(rank)
        prob = trained_probs_by_rank[rank].get(action)
        prob_ratio = prior_probability*prob
        ratios.append(prob_ratio)
    normalizer = functools.reduce(lambda x, y: x + y, ratios)
    rank = 1
    for ratio in ratios:
        posterior_probability = ratio/normalizer
        probabilities.update({rank:posterior_probability})
        rank += 1
    return probabilities
```

Poker Engine Progress:

Evidence of our progress can be seen within our repository. We have started to shape the framework/skeleton of our poker engine. Later we will revisit our functions to optimize them. We need to add the game play itself, such as fold, raise, check, etc.

```
44  def hand_rank(hand):
45          "Return a value indicating how high the hands ranks."
46          #counts is the count of each rank; ranks lists corresponding ranks
47          #E.g. '7 T 7 9 y' => counts = (3, 1, 1); ranks = (7, 10, 9)
48          groups = group(['--23456789TJQKA'.index(r) for r,s in hand])
49          counts, ranks = unzip(groups)
50          if ranks == (14, 5, 4, 3, 2):
51                  ranks = (5, 4, 3, 2, 1)
52          straight = len(ranks) == 5 and max(ranks)-min(ranks) == 4
53          flush = len(set([s for r,s in hand])) == 1
54          return (9 if (5,) == counts else
55                  8 if straight and flush else
56                  7 if (4, 1) == counts else
57                  6 if (3, 2) == counts else
58                  5 if flush else
59                  4 if straight else
60                  3 if (3, 1, 1) == counts else
61                  2 if (2, 2, 1) == counts else
62                  1 if (2, 1, 1, 1) == counts else
63                  0), ranks
64
```

Timeline

- Week 1: April 17 - April 24
    - Complete and validate the interface
    - Continue to build the poker engine as much as possible without predictive algorithm.
    - Validate feature values of the Naive Bayes algorithm from a finite set (hand ranks).
    - Then create the probabilistic model and determine if it needs to be accompanied with a decision rule (naive bayes classifier).
    - Validate model -- use extensive testing.
    - Program the model and integrate it into the poker engine.
    - Adjust the poker engine to use our naive bayes model in its playing strategy.
    - Functionality checkpoint/ Check in meeting/Code optimization
- Week 2: April 24 - May 1
    - Integrate a graphical user interface
    - Integrate a relational database to store player state (statistics, scores, pot, etc.)
    - Create the demo video
    - Write and finalize full report

Progress Report

Our group will be dividing the core implementation into two main parts - the algorithm and the poker bot. Two group members will work on each part and provide an interface to the other two to work with. We have exchanged complete interfaces with each other and each half of the group has started their implementation. We have implemented functionality to compute probabilities of ranks of hands based on trained data and next will implement the functions to train the data. In addition, we are going to create a light GUI that will display basic elements of the poker game. Within our repository, we have a UI.py file for a basic window that we will eventually build on using the Tkinter framework.

Version Control

We will be using a GitHub repository. Thomas Jiang, our teaching fellow has been given read/write access. The contents of our repository can be viewed at the URL provided below: https://github.com/MichaelDelaney/HarvardPokerBot/