CS 237 Fall 2018 Lab Three: Poker Probability

Due date: PDF file due Thursday September 27th @ 11:59PM (10% off if up to 24 hours late) in GradeScope

General Instructions

Please complete this notebook by filling in solutions where indicated. Be sure to “Run All” from the Cell menu before submitting.

You may use ordinary ASCII text to write your solutions, or (preferably) Latex. A nice introduction to Latex in Jupyter notebooks may be found here: http://data-blog.udacity.com/posts/2016/10/latex-primer/ (http://data-blog.udacity.com/posts/2016/10/latex-primer/)

As with previous homeworks, just upload a PDF file of this notebook. Instructions for converting to PDF may be found on the class web page right under the link for homework 1.

Lab 03 Introduction

In this lab we will explore Poker Probability, which is calculating the probability of various hands in the game of poker. This is, again, exploring how to confirm our theoretical understanding with experiments. If our experiments, as we increase the number of trials, converge to our theoretical calculation, then we have almost certainly analyzed it correctly.

There are many versions of poker (see here (http://www.wikihow.com/Play-Poker)) but the game we will study is called "five-card draw." It is described (https://www.pokernews.com/strategy/5-card-draw-rules-how-to-play-five-card-draw-poker-23741.htm) as follows:

Once everyone has paid the ante, each player receives five cards face down. A round of betting then occurs. If more than one player remains after that first round of betting, there follows a first round of drawing. Each active player specifies how many cards he or she wishes to discard and replace with new cards from the deck. If you are happy with your holding and do not want to draw any cards, you “stand pat.” Once the drawing round is completed, there is another round of betting. After that if there is more than one player remaining, a showdown occurs in which the player with the best five-card poker hand wins.

Here is an excellent short YT video on the basics of Poker: YT (https://www.youtube.com/watch?v=xfgMC3G37VE)

The only part we will care about is the final calculation of which hand wins: basically, the least probable hand wins. When you learn poker, then, one of the first things you have to learn is the ordering of the hands from most to least likely. Poker probability refers to calculating the exact probabilities of hands. The Wikipedia article (https://en.wikipedia.org/wiki/Poker_probability) contains the exact results and the formulae used to calculate them.

In this lab we will develop a framework for dealing 5-card hands and empirically estimating the probabilities of various hands. In fact, we will be able to do nearly all the hands commonly encountered. Our only constraint is that for the rarest hand, a Royal Flush, since there are only 4 such hands, the probability is so small it would take too long to get a reasonable estimate, and so we shall ignore this case.

This lab should help your understanding of the counting techniques covered in lecture.
In [10]:
    # Here are some imports which will be used in code that we write for CS 237
    # Jupyter notebook specific

    from IPython.display import Image
    from IPython.core.display import HTML
    from IPython.display import display_html
    from IPython.display import display
    from IPython.display import Math
    from IPython.display import Latex
    from IPython.display import HTML

    # Imports potentially used for this lab
    import numpy as np # arrays and functions which operate on array
    from numpy import linspace, arange
    import matplotlib.pyplot as plt # normal plotting
    #import seaborn as sns             # Fancy plotting
    #import pandas as pd               # Data input and manipulation
    from random import random, randint, uniform, choice, sample, shuffle, seed
    from collections import Counter

    %matplotlib inline

In [11]:
    seed(0)
    choice([2,3,4])

Out[11]: 3

Preface: Card Games and Probability

First we will first explore how to encode a standard deck of 52 playing cards, how to perform various tests on cards, and how to deal hands. To remind you, here is the illustration showing all the cards: cards (http://www.cs.bu.edu/fac/snyder/cs237/images/PlayingCards.png).

In [12]:
    # We will represent cards as a string, e.g., 'AC' will be Ace of Clubs

    # Denominations: 2, ..., 10, 'J' = Jack, 'Q' = Queen, 'K' = King, 'A' = Ace
    Denominations = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A']

    # Suits 'S' = Spades, 'H' = Hearts, 'D' = Diamonds, 'C' = Clubs
    Suits = ['C', 'H', 'S', 'D']

    # Note that colors are determined by the suits (hearts and diamonds are red, others black, # so, AC is Black

    # List comprehensions are a great way to avoid explicit for loops when creating lists

    Deck = [(d+s) for d in Denominations for s in Suits]  # Note the double for loop

    print( Deck )

In [13]: # Now we can "deal" cards by choosing randomly from the deck

    seed(0) # seed makes sure that all your computations start with the same random sequence;
    # this not really important, and only necessary for debugging
    and grading.
    def dealCard(): # choice randomly chooses an element of a list
        return choice(Deck)
    print( dealCard() )

8C

In [14]: # When dealing a hand in cards, the selection of cards is without replacement, that is, cards are
    # the deck one by one and not put back. This can be simulated in the choice function by setting the replace
    # parameter to False.
    seed(0)
    def dealHand(withReplacement = False, size = 5):
        if withReplacement:
            return choice(Deck)
        else:
            return sample(Deck,size)
    print( dealHand() )

['8C', 'AC', '8S', '2S', '6C']
In [15]: # extract the denomination and the suit from a card
def denom(c):
    return c[0:-1]

def suit(c):
    return c[-1]

# The function rank(c) will simply return the position of the card c PLUS 2 in the list 2, 3, ...., K, A. This will be used in an essential
# way in our code below. Although in the diagram given lecture, Ace is below 2, the Ace is actually considered to be ordered
# above the King, for example in determining a straight, under "low rules."
# rank(2) = 2, ...., rank(10) = 10, rank(Jack) = 11, rank(Queen) = 12, rank(King) = 13, rank(Ace) = 14

def rank(c):
    return Denominations.index(denom(c))+2

# Now we want to identify various kinds of cards

def isHeart(c):
    return (suit(c) == 'H')

def isDiamond(c):
    return (suit(c) == 'D')

def isClub(c):
    return (suit(c) == 'C')

def isSpade(c):
    return (suit(c) == 'S')

def isRed(c):
    return (isHeart(c) or isDiamond(c) )

def isBlack(c):
    return (not isRed(c))

def isFaceCard(c):
    return rank(c) >= 11 and rank(c) <= 13

Example Problem: What is probability that a 5-card hand has exactly 3 red cards?

Remember that in finite probability, for any event A,

$$P(A) = \frac{|A|}{|S|}.$$ 

Therefore, what we need to do in problems involving the probability of various kinds of hands in card games is to count the number of possible such hands, and divide by the total number of all possible hands. We developed analytical tools in lecture to do this, but here we are going to estimate it with repeated trials of dealing hands and testing for a given kind of hand.

In general for all but the last problem, we will use 100,000 trials to get a reasonable estimate of the probability. Since $1/100000 = 0.00001$ this means our resolution for experimental probabilities is 5 decimal places.
In [16]: # Return True iff the number of red cards in the hand h is 3
def threeRed(h):
    redCards = [c for c in h if isRed(c)]
    return (len(redCards) == 3)

seed(0)

# Run the experiment for 10,000 trials
# Print out probability that a 5-card hand has exactly 3 red cards
num_trials = 10**5
trials = [dealHand() for k in range(num_trials)]    # create list of 10000 hands randomly dealt

if(num_trials <= 10):
    # Just for this example, you don't need to do this unless you are debugging
    print(trials)

hands = [threeRed(h) for h in trials]    # convert this to list of true and false values
if(num_trials <= 10):
    print(hands)
prob = hands.count(True) / num_trials         # count the number of True values and divide by num_trials

# probability for 100,000 trials should be close to analytical value of 0.3251
print('Probability of exactly 3 red cards in a 5-card hand is ' + str(prob))

Probability of exactly 3 red cards in a 5-card hand is 0.32368

In [17]: # If you like cryptic, then you can put it all, including the calculation of trials in one line!
# Here are two examples. Note: these run the experiment again, so they won't match the previous 2 answers precisely.
seed(0)
prob3 = sum( [1 for h in [dealHand() for k in range(num_trials)] if threeRed(h)] ) / num_trials
print('Probability of exactly 3 red cards in a 5-card hand is ' + str(prob3))

seed(0)
prob4 = sum( [1 for k in range(num_trials) if threeRed(dealHand())] ) / num_trials
print('Probability of exactly 3 red cards in a 5-card hand is ' + str(prob4))

Probability of exactly 3 red cards in a 5-card hand is 0.32368
Probability of exactly 3 red cards in a 5-card hand is 0.32368

Problem 1: What is probability that a 5-card hand has at least 3 Diamonds?

In [18]: # Print out probability that a 5-card hand has 3, 4, or 5 diamonds.
seed(0)

Probability of at least 3 diamonds in a 5-card hand is 0.09189
Problem 2: What is probability of a flush in Poker?

In Poker, a flush is 5 cards of the same suit, but excludes straight flushes and royal flushes; these, however, are so rare (there are only 40 of them in all), that they are around or below our resolution (0.00001), so we just will determine if all suits are the same.

In [19]: # Print out probability that a 5-card hand has all the same suit

Probability of a flush in 5-card poker is 0.00207

Problem 3: What is probability of a straight in Poker?

In poker, a straight a hand in which the ranks form a contiguous sequence, e.g., 2,3,4,5,6. The suits do not matter. Also, for simplicity, we will NOT use the "high rule (https://en.wikipedia.org/wiki/List_of_poker_hands#Straight)" whereby the Ace can be either low (below the 2) or high (above the King). We will assume that the Ace must always be high, which simplifies the calculation a little bit, since we can just check if we sort the cards by rank, they are contiguous.

In [20]: # Print out probability that a 5-card hand is a straight

Probability of a straight in poker 0.00368

Problem 4: Rank Signature of a poker hand

Let us define the rank signature of a hand as an ordered histogram of the ranks occurring in the hand; that is, we count the frequency of the ranks occurring in the hand, and order this sequence. Here are some examples:

- Five cards all of different ranks (e.g., Ace, 4, 2, King, 8): [1,1,1,1,1]
- One pair, 2 cards of the same rank, and 3 more all of different ranks (e.g., 2,2,6,3,Ace): [1,1,1,2]
- Two pair, 2 pairs (of different ranks) and one card of a different rank (e.g., 2,2,Ace,3,Ace): [1,2,2]
- Full house, 2 cards of the same rank, and 3 cards of the same rank (e.g., 8,Jack,8,8,Jack): [2,3]

Many poker hand can be defined solely in terms of the ranks involved. The importance of this concept is that once we write a function to estimate the probability of a given signature, we can then immediately calculate the probability of many different poker hands.

For this problem you must write a function which calculate the probability that a 5-card hand has a given signature and verify it by calculating the probability of no two ranks being the same (first choose 5 different ranks, then consider all possible enumerations of suits):

$$\frac{\binom{13}{5} \cdot 4^5}{\binom{52}{5}} = 0.5071$$

Note: This is NOT the same as "no pair/high card" since in Poker, this hand means you don’t have a flush or a straight (both of which have 5 cards of different rank). To find out the correct probability, you would need to subtract the possibility of a flush or a straight (or both, a straight-flush), which we will do in the next problem.

In [21]: ### Run the experiment for 100,000 trials

# Print out probability that a 5-card hand has a given signature

Probability of five different ranks is 0.50566

Problem Five: Using rank signature to calculate six different poker hands

Problem 5 (A): What is probability of No Pair/High Card in Poker?

For this, you must use the rank signature of 5 different ranks (as in the previous problem) but exclude any hands that are straights or flushes.

In [22]: # probability should be close to analytical value of 0.5012

Probability of No Pair/High Card is 0.49991
Problem 5 (B): What is probability of One Pair in Poker?

In [23]: # probability should be close to analytical value of 0.4226

Probability of one pair in poker is 0.42456

Problem 5 (C): What is probability of Two Pairs in Poker?

In [24]: # probability should be close to analytical value of 0.047539

Probability of two pairs in poker is 0.04618

Problem 5 (D): What is probability of Three of a Kind in Poker?

In [25]: # probability should be close to analytical value of 0.021128

Probability of three of a kind in poker is 0.02177

Problem 5 (E): What is probability of a Full House in Poker?

In [26]: # probability should be close to analytical value of 0.001441

Probability of a full house in poker is 0.00151

Problem 5 (F): What is probability of Four of a Kind in Poker?

In [27]: # probability should be close to analytical value of 0.000240

Probability of four of a kind in poker is 0.00032

Problem 5 (G): What is probability of a Straight Flush in Poker?

We will ignore the difference between this and a Royal Flush, so this is essentially calculating the cumulative probability (probability of this hand or better).

You will have to do this 10^6 times to get an accurate result.

In [28]: # probability should be close to analytical value of 0.000015 = approximately 1.539 * 10^-5

Probability of Straight Flush is 1.5e-05