CS 237 Lab Three: Poker Probability

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.

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 two rarest hands, the probability is so small it would take too long to get a reasonable estimate, and so we shall ignore them.

This should complete your understanding of the counting techniques covered in lecture.

In [89]:

1 # Here are some imports which will be used in code that we write for CS 237
2 # Jupyter notebook specific
3
4 from IPython.display import Image
5 from IPython.core.display import HTML
6 from IPython.display import display_html
7 from IPython.display import display
8 from IPython.display import Math
9 from IPython.display import Latex
10 from IPython.display import HTML
11
12 # Imports potentially used for this lab
13
14 import numpy as np # arrays and functions which operate on array
15 from numpy import linspace, arange
16 import matplotlib.pyplot as plt # normal plotting
17 import seaborn as sns # Fancy plotting
18 import pandas as pd # Data input and manipulation
19
20 from numpy.random import random, randint, uniform, choice, shuffle
21 from collections import Counter
22
23 #matplotlib inline
24

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 from hw 01 showing all the cards: cards (http://www.cs.bu.edu/fac/snyder/cs237/Homeworks/images/hw01_PlayingCards.png).
# We will represent cards as a string, e.g., 'AC' will be Ace of Clubs

# Denominations: '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)

# Now we can "deal" cards by choosing randomly from the deck

def dealCard():
    return choice(Deck)  # choice randomly chooses an element of a list

print(dealCard())

# When dealing a hand in cards, the selection of cards is without replacement, that is, cards
# the deck one by one and not put back. This can be simulated in the choice function by setting
# parameter to False.

def dealHand(withReplacement = False, size = 5):
    return choice(Deck, replace=withReplacement, size=size)

print(dealHand())

['QS', '9H', '7S', '10H', '3D']
In [93]:
# extract the denomination and the suit from a card

def denom(c):
    return c[0:-1]

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

# The RANK of a denomination will be its position in the list 2, 3, ..., K, A. This will be the way in our code below. Although in the diagram linked above, Ace is below 2, the Ace is actually above the King, for example in determining a straight.

# rank(2) = 0, ..., rank(10) = 8, rank(Jack) = 9, rank(Queen) = 10, rank(King) = 11, rank(Ace) = 12

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

# 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) >= 9 and rank(c) <= 11

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40

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

In [97]:
# Run the experiment for 10,000 trials

# Print out probability that a 5-card hand has 3, 4, or 5 diamonds.

def atLeast3Diamonds(h):
    pass

num_trials = 100000
trials = [dealHand() for k in range(num_trials)]
hands = [atLeast3Diamonds(h) for h in trials] # convert this to list of true
prob = hands.count(True) / num_trials

# Should be close to analytical value of 0.0928
print('Probability of at least 3 diamonds in a 5-card hand is ' + str(prob))

Probability of at least 3 diamonds in a 5-card hand is 0.09389

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, that they will not affect our approximation, so we just have to determine if all suits are the same.
In [98]:
1 # Run the experiment for 10,000 trials
2 # Print out probability that a 5-card hand has all the same suit
3
4 def isFlush(h):
5      pass
6
7 num_trials = 10000
8
9 trials = [dealHand() for k in range(num_trials)]
10
11 # Using the *bit cryptic* method shown above
12 prob = sum( [1 for h in trials if isFlush(h)] ) / num_trials
13
14 # probability should be close to analytical value of 0.001965
15 print('Probability of a flush in 5-card poker is ' + str(prob))

Probability of a flush in 5-card poker is 0.0026

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. As with flushes, we exclude straight flushes and royal flushes.

In [101]:
1 # Print out probability that a 5-card hand is a straight
2
3 def isStraight(h):
4      pass
5
6 num_trials = 10000
7
8 # Using the *cryptic* method shown above
9 prob = sum( [1 for k in range(num_trials) if isFlush(dealHand())] ) / num_trials
10
11 # probability should be close to analytical value of 0.003925
12 print('Probability of a straight in poker ' + str(prob))

Probability of a straight in poker 0.002

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:

- No pair/ high card, 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): [1,1,2]
- 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,Jack): [2,3]

It is not important what suits are involved in these hands, and so they 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 pair / high card.
Problem Five: Using rank signature to calculate five different poker hands

For all of these except the last, you should do 10,000 trials.

Problem 5 (A): What is probability of one pair in Poker?

In [103]:
# probability should be close to analytical value of 0.422569
3 print('Probability of one pair in poker is ' + str(probability_of_rank_signature([1,1,1,2],100))
Probability of one pair in poker is 0.4247

Problem 5 (B): What is probability of two pairs in Poker?

In [104]:
# probability should be close to analytical value of 0.047539
3 print('Probability of two pairs in poker is ' + str(probability_of_rank_signature([1,2,2],1000))
Probability of two pairs in poker is 0.0506

Problem 5 (C): What is probability of three of a kind in Poker?

In [105]:
# probability should be close to analytical value of 0.021128
3 print('Probability of three of a kind in poker is ' + str(probability_of_rank_signature([1,1,3],1000))
Probability of three of a kind in poker is 0.0225

Problem 5 (D): What is probability of a full house in Poker?

In [106]:
# probability should be close to analytical value of 0.001441
3 print('Probability of a full house in poker is ' + str(probability_of_rank_signature([2,3],1000))
Probability of a full house in poker is 0.0009

Problem 5 (E): What is probability of four of a kind in Poker?

The probability here is so small that you should run the experiment 1000,000 times, which will give you five digits of precision.

In [107]:
# probability should be close to analytical value of 0.000240
3 print('Probability of four of a kind in poker is ' + str(probability_of_rank_signature([1,4],10000000))
Probability of four of a kind in poker is 0.00036
In [ ]: 1