Data Analysis and Machine-Learning

Chapter 4:

Monte Carlo Method



1. Monte Carlo Simulation: Gambler’s Ruin

Monte Carlo method, a computation algorithm based on random numbers, is a representative type of algorithm logic for understanding the mechanism of AI and machine learning. Consider the following representative example for understanding.

Supposing 45% winning rate for a gambler with seed money of $5,000 who either wins $150 or loses $150 from the gamble, what would be the probability distribution for this gambler’s ruin? It is possible to generate a following sample algorithm based on Bernoulli distribution.

#Simulation

from scipy.stats import bernoulli

seed = 5000

p = 0.45

balance = []

bet = [-150, 150]

while True:

    i = bernoulli(p).rvs(1)[0]

    seed += bet[i]

    balance.append(seed)

    if seed<=0:

        break

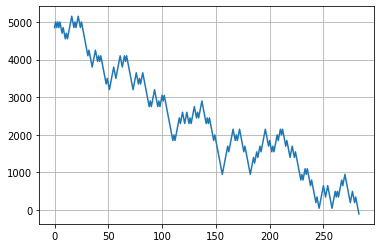
#Visualization

import matplotlib.pyplot as plt

plt.plot(balance)

plt.grid()

plt.show()



In this particular case, the gambler loses all his seed money at around 280th game. What we really want to know, however, is the probability distribution of the probability variable X. In average, when does the gambler lose all his seed money, and what would be the variance of the distribution? Let us figure this out by iterating MC simulation 500 times:

#Simulation

ngames = []

p = 0.45

bet = [-150, 150]

for \_ in range(500):

    seed = 5000

    cnt = 0

    while True:

        cnt += 1

        i = bernoulli(p).rvs(1)[0]

        seed += bet[i]

        if seed<=0:

            break

    ngames.append(cnt)

#Mean and Variance

import numpy as np

display(np.mean(ngames))

display(np.var(ngames))

Output:

344.0

35330.08

In average, gambler loses all his money at 344th game.

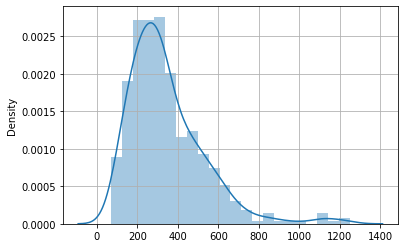
#Visualization of the Distribution

import seaborn as sns

sns.distplot(ngames)

plt.grid()

plt.show()



Now, when would be the average and variance of the first time that the gambler loses the game?

ngames = []

p = 0.45

bet = [-150, 150]

for \_ in range(500):

    seed = 5000

    cnt = 0

    while True:

        cnt +=1

        i = bernoulli(p).rvs(1)[0]

        seed += bet[i]

        if i==0:

            break

    ngames.append(cnt)

display(np.mean(ngames))

display(np.var(ngames))

Output:

1.86

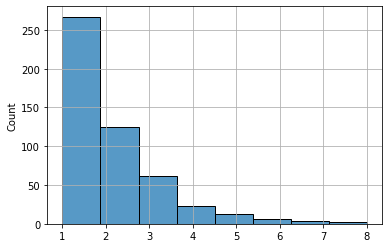
1.5403999999999995

#Visualization

sns.histplot(ngames, bins=8)

plt.grid()

plt.show()



a computation methodology

is a representative for AI and machine-learning.