Unicamp - Instituto de Computação

MC886 B / MO416 A

Introdução à Inteligência Artificial

Prof. Dr. Jacques Wainer

Exercício 2

Aluno:
George Gigilas Junior - 216741

Introdução

Este exercício corresponde ao exercício 13.10 do livro, cujo enunciado está disponível juntamente com a atividade. Este arquivo corresponde às minhas soluções para os itens a, b e c do problema. Após discutido por email e em aula, vou adotar que a ordem em que aparecem os símbolos na máquina de caça-níqueis importa.

Solução

Item a - Calcular o retorno esperado da máquina de caça-níqueis

Para isso, primeiro vou calcular o número total de combinações possíveis:

- Temos 3 espaços de símbolo em cada combinação.
- São 4 símbolos possíveis para cada espaço.

Com isso, conclui que o número de combinações é $4 . 4 . 4 = 64$.

Agora, para calcular o retorno esperado, devemos calcular a média de retornos de coins, levando em conta o número de combinações que resulta em cada retorno possível. Dois casos especiais são: “CHERRY/CHERRY/?” e “CHERRY/?/?”, que possuem mais de uma combinação para o mesmo retorno.

- Em “CHERRY/CHERRY/?”, o último símbolo pode ser qualquer um, com exceção de CHERRY, senão a combinação cai em outro caso (com outra premiação). Portanto, são 3 casos de “CHERRY/CHERRY/?” que resultam no retorno de 2 coins.
- Em “CHERRY/?/?”, o primeiro ponto de interrogação não pode ser CHERRY, senão a combinação cai no caso de “CHERRY/CHERRY/?”. Já o segundo ponto de interrogação pode ser qualquer valor. Assim, são 3 . 4 = 12 casos em que “CHERRY/?/?” resulta no retorno de 1 coin.
- Vale lembrar que o número de ocorrências em que o prêmio é 0 coins é: o número total de eventos menos o total de eventos que dão prêmio de 1 coin ou mais.
- Estou considerando S como sendo o conjunto que contém todas as combinações de símbolos.

Assim, temos:

$$\frac{1}{64} . \sum_{x \in S} retorno(x)$$

$$= \frac{1 . 21 + 1 . 16 + 1 . 5 + 1 . 3 + 3 . 2 + 12 . 1 + (64 - 19) . 0}{64} = \frac{63}{64} = 0,984375$$

Sendo assim, o retorno esperado é de 0,984375 coins para uma coin colocada na máquina. Isso significa que, em média, a cada coin colocada, o jogador perde 0,015625 coins, resultando em um lucro para a máquina.

Item b - Calcular a probabilidade de colocar uma coin e vencer

Como definido em sala de aula, o jogador vence quando obtém algum lucro após colocar uma coin na máquina. Em outras palavras, o jogador vence quando o retorno da máquina é estritamente maior que 1 coin.

Para calcular essa probabilidade, basta contar o número de combinações que resultam em um retorno maior que 1 coin e dividir pelo número total de combinações possíveis. Assim, temos:

$$P(retorno > 1) = \frac{1+1+1+1+3}{64} = \frac{7}{64} = 0,109375$$

Dessa forma, a probabilidade de vencer após uma rodada é de 10,9375%. Ou seja, é muito mais provável que o jogador perca dinheiro do que ele ganhe.

Item c - Estimar a média e a mediana do número de jogadas a se fazer até ficar com 0 coins, começando com 8 coins.

Para calcular essas grandezas, como sugerido no enunciado, fiz um programa em Python que estima esses dois valores, a partir de uma simulação da máquina de caça-níqueis. No caso, eu determinei que o programa fizesse 10.001 jogos até ir à falência (ficar com 0 coins). Assim, acumulei em uma variável o número de jogadas que o jogador fez até ir a falência, para depois dividir pelo número total de jogos (10.001) e calcular a média. Além disso, guardei esses valores em um vetor, ordenei esse vetor e peguei a 5000ª posição, que corresponde à mediana. O programa está a seguir:

```
import random

# I'm defining 1 as BAR, 2 as  BELL, 3 as LEMON and 4 as CHERRY

numberOfRounds = []      # Stores the number of rounds it took to go
broke, for each iteration
partialMean = 0          # Accumulates the number of rounds it took to
go broke, for each iteration
                         # It helps calculating the mean

for i in range(10001):

    coins = 8            # Initial coin amount
    count = 0            # Counts how many rounds it took to go broke

    while 1:
        count += 1
        coins -= 1
```

```
        firstSlot = random.randint(1, 4)
        secondSlot = random.randint(1, 4)
        thirdSlot = random.randint(1, 4)

        if (firstSlot == 1 and secondSlot == 1 and thirdSlot == 1):
            coins += 21
        elif (firstSlot == 2 and secondSlot == 2 and thirdSlot == 2):
            coins += 16
        elif (firstSlot == 3 and secondSlot == 3 and thirdSlot == 3):
            coins += 5
        elif (firstSlot == 4 and secondSlot == 4 and thirdSlot == 4):
            coins += 3
        elif (firstSlot == 4 and secondSlot == 4 and thirdSlot != 4):
            coins += 2
        elif (firstSlot == 4 and secondSlot != 4):
            coins += 1

        if(coins == 0):
            numberOfRounds.append(count)
            partialMean += count
            break

mean = partialMean / 10001
numberOfRounds.sort()
median = numberOfRounds[5000]

print("The estimated mean is " + str(mean) + " and the estimated median
is " + str(median))
```

Após rodar esse programa, a saída foi:
The estimated mean is 563.940705929407 and the estimated median is 15

A partir desse resultado, à primeira vista podemos estranhar essa diferença tão grande entre os dois valores. Isso se deve ao fato de que os outliers (também conhecidos como pontos fora da curva) acabam elevando bastante a média calculada. Quando um jogador tem muita sorte, ele pode acabar levando muitas rodadas para falir, e isso desloca a média. Por exemplo, se o jogador ganha 21 moedas com a combinação “BAR/BAR/BAR”, isso faz com que sejam necessárias, no mínimo, 21 novas jogadas até a falência. Já a mediana corresponde ao valor que está no meio do vetor ordenado de ocorrências, e é um valor mais adequado para fazer análises estatísticas, por ignorar os valores nas duas extremidades do vetor.

Portanto, podemos concluir que, na maioria das vezes, começando com 8 coins, o jogador leva cerca de 15 rodadas para falir.