Methods in Artificial Intelligence Homework 1

Christoffer Brevik

January 24, 2025

1 Poker Hands

1.1 How many atomic events are there in the joint probability distribution

1.1.1 Explanation

In poker, the lowest hand available is the *High Card*, where the highest card available is used. Therefore every single combination of cards will lead to a hand. Therefore, to estimate all possible hands (atomic events), we will use combination to calculate all possible combinations of cards.

1.1.2 Calclation

As we have 52 cards, and will draw 5 cards from the pile, we will use:

$$C(n,r) = \frac{n!}{r!(n-r)!}$$

Here n is the total cards (52) and r is how many cards we draw from the pile (5). We therefore get:

$$C(52,5) = \frac{52!}{5!(52-5)!}$$

$$C(52,5) = \frac{52!}{5! \times 47!}$$

Solving this, we get

$$C(52,5) = 2598960$$

We see there are *2598960* different hands with 5 cards from a standard pile.

1.2 What is the probability of each atomic event

1.2.1 Explanation

As the atomic events calculated in section 1 are equally possible, we can find the chance of one of these events happening by dividing by total possibilities.

1.2.2 Calculation

$$P(event) = \frac{1}{total_events}$$

$$P(event) = \frac{1}{2598960} \approx 3.85 \times 10^{-7}$$

1.3 hat is the probability of being dealt a royal straight flush? Four of a kind?

1.3.1 Royal Flush - Explanation

A royal flush is a straight flush with the best five cards in the game (10, J, Q, K & A). This means that we need al cards in the same suit, but that this is possible for all five suits. With this in mind we can take the chance of one of the suits getting royal flush and multiplying by four. Still, the order does not count, so we also have to account for the fact that any of the valid cards can be picked at any position, given that all

1.3.2 Royal Flush - Calculation

We need 5 specific cards from the pile, but that order does not matter (Here we can use nCr, but as the calculations are quite simple I will just do this manaually). Knowing that the pile reduces for each card picked and order does not matter, we get:

$$\frac{5}{52} \times \frac{4}{51} \times \frac{3}{50} \times \frac{2}{49} \times \frac{1}{48}$$

We now multiply this by 4, as we have four possible suits

$$P(Royal\_Flush) = 4 \times (\frac{5}{52} \times \frac{4}{51} \times \frac{3}{50} \times \frac{2}{49} \times \frac{1}{48})$$

Solving the equation we to get:

$$P(Royal\_Flush) = \frac{1}{649740} \approx 1.54 \times 10^{-6}$$

1.3.3 Four of a Kind

As we already have the number of total hands (atomic events), we can just estimate how many of these can be considered a four of a kind.

$$P(Four\_of\_a\_Kind) = \frac{Total\_Four\_of\_a\_Kind\_Hands}{Total\_Hands}$$

We already have Total hands from section 1. To find out how many four of a kinds exist, we can start with the fact that four of a kind goes over every suit, and it therefore only exists 13 different ones in the game.

$$Total\_Four\_of\_a\_Kind\_types = 13$$

Still, the fifth card also matters as, while not accounting for four of a kind, still makes the hand unique. as our four cards reduced the pile from 52 to 48, we know that there are 48 different cards that can account for the last card. We therefore get

$$Total\_Four\_of\_a\_Kind\_Hands = 13 \times 48$$

$$Total\_Four\_of\_a\_Kind\_Hands = 624$$

We can now estimate the chance of getting one of these 624 combinations

$$P(Four\_of\_a\_Kind) = \frac{Total\_Four\_of\_a\_Kind\_Hands}{Total\_Hands}$$

$$P(Four\_of\_a\_Kind) = \frac{624}{2598960}$$

Solving this we get

$$P(Four\_of\_a\_Kind) = \frac{1}{4165} \approx 2.40 \times 10^{-4}$$

2 Slot Machine

2.1 Compute the expected “payback” percentage of the machine.

2.1.1 Explanation

To estimate this, we estimate the expected payout for each situation seperatly, then combine these

2.1.2 Calculation

Bar/Bar/Bar
To estimate predicted payback from this combination alone, we use

$$E_p(Br/Br/Br) = P(Br/Br/Br) * R(Br/Br/Br)$$

Estimating the probability, we get

$$P(Br/Br/Br) = (\frac{1}{4})^3$$

$$P(Br/Br/Br) = \frac{1}{64}$$

Getting this combination gives us 20 coins, we get the expected payout

$$E_p(Br/Br/Br) = \frac{1}{64} * 20$$

$$E_p(Br/Br/Br) = \frac{20}{64}$$

Bell/Bell/Bell
The probability is the same as $P(Br/Br/Br)$ but gives us 15 coins, we get the expected payout

$$E_p(Bl/Bl/Bl) = \frac{15}{64}$$

Lemon/Lemon/Lemon
The probability is the same as $P(Bl/Bl/Bl)$ but gives us just 5 coins, we get the expected payout

$$E_p(L/L/L) = \frac{5}{64}$$

Cherry/Cherry/Cherry
The probability is the same as $P(L/L/L)$ but gives us 3 coins, we get the expected payout

$$E_p(C/C/C) = \frac{3}{64}$$

Cherry/Cherry/*Not-Cherry*
Here we can get any value for the last wheel, but we need to assure we don't cover the case where we get *Cherry/Cherry/Cherry*, effectively removing cherry as an option. The probability is therefore

$$P(C/C/?) = \frac{1}{4} \times \frac{1}{4} \times \frac{3}{4}$$

$$P(C/C/?) = \frac{3}{64}$$

This gives us 2 coins, so we get the expected payout

$$E_p(Br/Br/Br) = \frac{3}{64} * 2$$

$$E_p(C/C/?) = \frac{6}{64}$$

Cherry/*Not-Cherry*/*Any*
Here we can get any value for the second wheel, except cherry (As this would clash with C/C/?). The last wheel however, can be anything. The probability is therefore

$$P(C/?/?) = \frac{1}{4} \times \frac{3}{4} \times \frac{4}{4}$$

$$P(C/?/?) = \frac{12}{64}$$

This returns our coin, so we get the expected payout

$$E_p(C/?/?) = \frac{3}{64}$$

2.1.3 Combining rewards

As all expected payouts account for probability, we can just add them together

$$E_p(total) = E_p(Br/Br/Br) + E_p(Bl/Bl/Bl) + E_p(L/L/L) + E_p(C/C/C) + E_p(C/C/?) + E_p(C/?/?)$$

$$E_p(total) = \frac{20}{64} + \frac{15}{64} + \frac{5}{64} + \frac{3}{64} + \frac{6}{64} + \frac{12}{64}$$

Solving this equation we get

$$E_p(total) = \frac{61}{64} \approx 0.9531$$

Meaning that we expect to loose on average

2.2 Compute the probability that playing the slot machine once will result in a win

The probability that playing the slot machine will result in a win is that same as the probability that we will get any of the win situations. We have already estimated this in the previous task, so we only need to combine this

$$P(Win) = P_W(Br/Br/Br) + P_W(Bl/Bl/Bl) + P_W(L/L/L) + P_W(C/C/C) + P_W(C/C/?) + P_W(C/?/?)$$

Here we know that the probability of getting only Bars, only Bells, only Lemons and only Cherries are equal, we can simplify to

$$P(Win) = 4 \times P_W(Br/Br/Br) + P_W(C/C/?) + P_W(C/?/?)$$

$$P(Win) = 4 \times \frac{1}{64} + \frac{3}{64} + \frac{12}{64}$$

$$P(Win) = \frac{19}{64} \approx 0.297$$

2.3 Estimate the mean and median number of plays you can expect to make until you go broke, if you start with 10 coins, with Python

Instead of implementing the results directly, I used the probabilities of one occuring as well as the corresponding payout. This way I would accuratly representent the game without having to actually implement the wheels and corresponding results.

2.3.1 Code Location

My code for this assignment is available on my Github at Metoder i Kunstig Intelligens

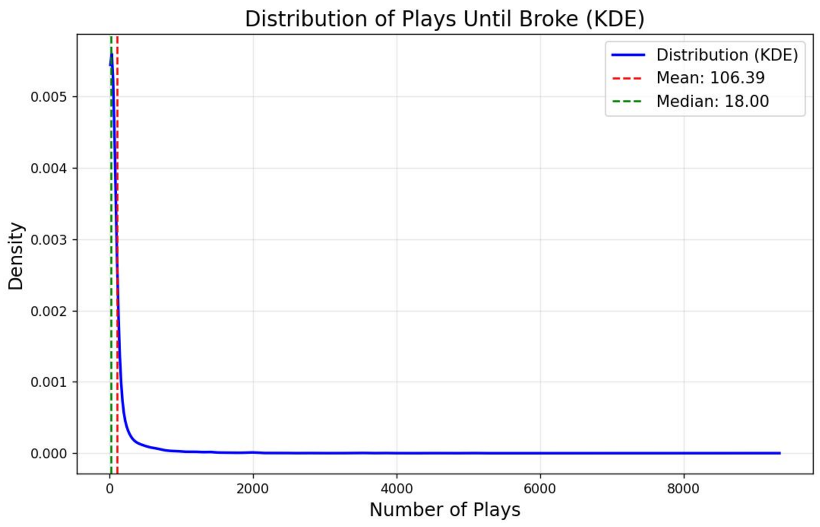

Figure 1: Graph showing number of plays for 10,000 actors with 10 coins each

2.3.2 Results

[h]

From Figure 1 we can see that while some actors make it to almost 10,000 plays. Most are well under 2000. My program returns this

```
Mean plays until broke: 106.3915
Median plays until broke: 18.0
```

This aligns with the graph and shows that it is rare to make it past 100 plays. The difference between mean and median here is due to some outliers gaining a lot trough the game, even though this is very unlikely

3 Python for Simulating events

3.1 Birthday Problem

3.1.1 Code Location

As in the previous task, my code for this assignment is available on my Github at Metoder i Kunstig Intelligens

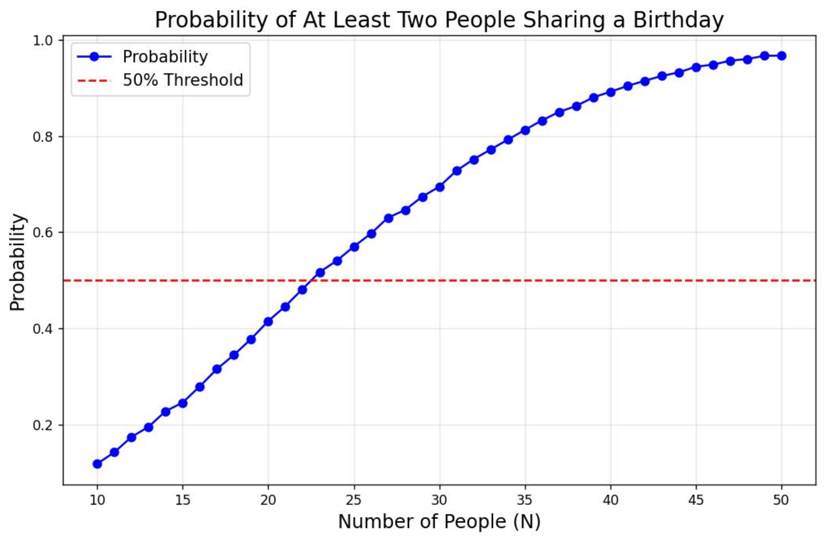

Figure 2: Probability of two or more people sharing a birthday in the distribution [10, 50]

3.1.2 Code snippets

Create a function that takes N and computes the probability of the event via simulation.
To solve this assignment, I used this function:

```
# Function to simulate probability of at least two people sharing a birthday
def birthday_simulation(N, runs=10000):
    successes = 0

    for _ in range(runs):
        # Generate random birthdays for N people
        birthdays = [random.randint(1, 365) for _ in range(N)]
        # Check for duplicates
        if len(birthdays) != len(set(birthdays)):
            successes += 1

    # Probability of at least one duplicate
    return successes / runs
```

Use the function created in the previous task to compute the probability of the event given N in the interval [10, 50]. In this interval, what is the proportion of N where the event happens with the least 50% chance? What is the smallest N where the probability of the event occurring is at least 50%?
To solve this assignment, I implemented this function:

```
# Used only for task b
def probabilities_in_range(start=10, end=50, target_probability=0.5, trials=10000):
    probabilities = []
    for N in range(start, end + 1):
        prob = birthday_simulation(N, trials)
        probabilities.append((N, prob))

    # Find proportion where probability >= target_probability
    count_above_threshold = sum(1 for _, prob in probabilities if prob >= target_probability)
    proportion = count_above_threshold / (end - start + 1)

    return probabilities, proportion
```

3.1.3 Result

As the two sub-tasks are intertwined, I will just answer the task as a whole. Running my program you get the graph shown in Figure 2, the console also returns

```
The chance of 25 people contain atleast two people sharing a birthday is 0.1183
Proportion of N in [10, 50] with probability >= 50%: 0.6829
```

3.2 Person for every day Problem

3.2.1 Code Location

As in the previous tasks, my code for this assignment is available on my Github at Metoder i Kunstig Intelligens

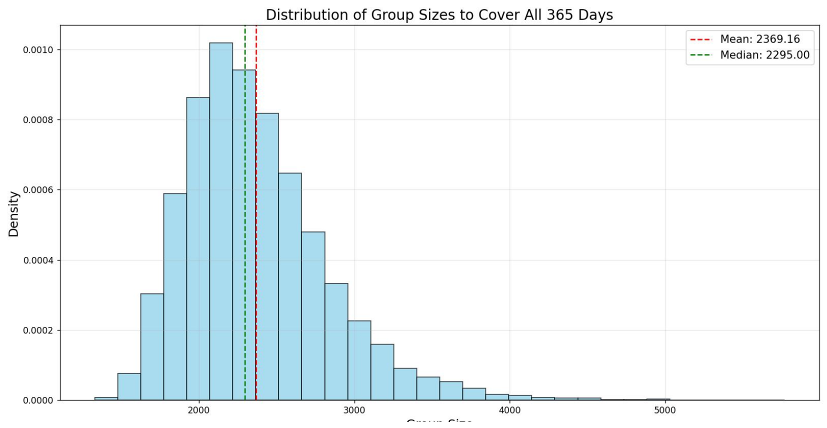

Figure 3: Distribution of people required to get 365 different birthdays

3.2.2 Code snippets

To solve this assignment, I used this function:

```
# Function to simulate the group size required to cover all 365 days by peoples birthdays
def simulate_group_size(trials=10000):
    group_sizes = []

    for _ in range(trials):
        days_covered = set()
        group_size = 0

        while len(days_covered) < 365:
            # Add a person with a random birthday
            birthday = random.randint(1, 365)
            days_covered.add(birthday)
            group_size += 1
        group_sizes.append(group_size)

    # Return the average group size
    return np.mean(group_sizes)
```

3.2.3 Result

Using my function described above and plotting the results, I got Figure 3, the console also returns

```
Expected (mean) group size to cover all 365 days: 2369.16
Median group size to cover all 365 days: 2295.00
```