2MMS50 Assignment 1

Ivan Lee (2273780)

May 12, 2025

Two types of investments (choices), Cash account which is risk-free, getting $R_{f,t}$ and stock getting $R_{r,t}$ both at time t to $t+1$.

Goal 1: Find the fraction ω_t of wealth to invest in stocks.

$$R_{r,t} \begin{cases} (1+u)R_{f,t} = 1.836 & \text{w.p. } \frac{1}{2} \\ (1+d)R_{f,t} = 0.714 & \text{w.p. } \frac{1}{2} \end{cases}$$

With $u = 0.8, d = -0.3$ and $R_{f,t} = 1.02$

Goal 2: Find C_t which is how much the individual can consume at a time t, measured as an amount. The happiness is measured with the following utility function.

$$u(C_t) = max\left\{\log C_t, 0\right\}$$

Goal 3: Find the ω_t and C_t to maximise the following.

$$\mathbb{E}_0\left[\sum_{\tau=0}^{\infty} \delta^\tau u\left(C_t\right)\right]$$

Where δ is the discount factor and in our case $\delta = 1$. We set $\omega_t \in [0,1]$ and discretised into 5 evenly spaced points. And wealth W_t is rounded down to the nearest point in $\{0, 1, \ldots, 10\}$

$$\omega_t \in [0, 0.25, 0.5, 0.75, 1]$$

Salary

$$S_t \begin{cases} 1 & w.p. \frac{1}{2} \\ 2 & w.p. \frac{1}{2} \end{cases}$$

1 Part A

State space $\mathcal{I} = \{0, 1, \ldots, 10\}$

Action space given state i is $\mathcal{A}(i) = \{(\omega, C) | \omega \in \{0, 0.25, 0.5, 0.75, 1\}, C \in \{0, 1, \ldots, i\}\}$ where i is the amount of wealth at that state

Direct rewards is the utility gained from the consumption chosen at the state i and is given by $r_a(i) = u(C) = max\{\log C, 0\}$

Recursive relationship of W_t (the value of wealth at the start of the year before consumption, salary and investments)

$$W_t = (s_{t-1} - C_{t-1} + W_{t-1}) \cdot ((1 - \omega_t) R_{f,t} + \omega_t R_{r,t})$$

Transition Probabilities is the chance we move to the next state from the current state.

Given these limitations, it is possible to calculate the transition probabilities for each state and each action. The transition probabilities are given by the following algorithm:

If the consumption is greater than the state, then the action is invalid and the transition probabilities are all zero. Otherwise, the transition probabilities are calculated as follows:

1. Calculate the wealth before investment as $W_{t-1} + S_{t-1} - C_{t-1}$.
2. For each investment outcome, calculate the next state as $W_{t-1} + S_{t-1} - C_{t-1}$ multiplied by the investment outcome and rounded down to the nearest integer for the fee.
3. If the next state is less than 0, set it to 0. If it is greater than 10, set it to 10.
4. With the frequency tables of each outcome of that state saved, the transition probabilities are calculated as the frequency of each outcome divided by the total number of outcomes.

Given that by way of calculation, the sum of all probabilities in each row is 1. Because $\frac{n}{n} = 1$

Algorithm is given in the appendix.

2 Part B

$\omega_t = 0.75$ and $C_t = W_t$ for f^{rg}

Since we consume all available wealth at each state, the only investable income is the salary. The transition probabilities are the same for all states and consumptions. The transition probabilities are given by the following matrix:

$$p^{0.75,W_t}(i,j) = \begin{bmatrix} 0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The long run exepected utility is given by the following equation (Since log(0) is undefined, we set it to 0):

$$0.25 \cdot \log(0) + 0.5 \cdot \log(1) + 0.25 \cdot \log(3) = 0.25 \cdot \log(3) \approx 0.119$$

I think that the superhuman would not be very happy with this policy in the long run, Because the expected utility is very low. The superhuman would be better off with a slightly less consumption in poor states and a higher consumption in rich states. This would give a higher expected utility and a better long run outcome.

3 Part C

To perform one step policy iteration, we try to improve the policy by picking an action a that maximizes the expected utility of future steps given that the rest of the policy is unchanged.

We start with the policy of $(\omega, C) = (0.75, W_t)$ and try to improve it by choosing a different action a for the first state $i = 0$. The expected utility of the current policy is given by:

$$0.25 \cdot \log(0) + 0.5 \cdot \log(1) + 0.25 \cdot \log(3) = 0.25 \cdot \log(3) \approx 0.119$$

We test the new policy that states if the state is 1, then we consume 0.

Our new transition probabilities are given by the following matrix:

$$
p^{0.75,*}(i,j) = \begin{bmatrix}
0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.25 & 0.25 & 0.25 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.25 & 0.5 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

Since for all states that are not 1, we will consume all the available wealth, they will have an investable income of 1 or 2 from the salary. However in the 50% chance they end up un state 1, they will not invest the wealth and have equal probability of getting state $1, 2, 3, 4$

Therefore the expected utility of the new policy is given by:

$$
\begin{aligned}
0.25 \cdot \log(0) + 0.5 \cdot (0.25 \cdot \log(1) + 0,25 \cdot \log(2) + 0,25 \cdot \log(3) + 0.25 \cdot \log(4)) + 0.25 \cdot \log(3) \\
= 0.5 \cdot (0,25 \cdot \log(2) + 0,25 \cdot \log(3) + 0.25 \cdot \log(4)) + 0.25 \cdot \log(3) \approx 0.291
\end{aligned}
$$

Since this is higher than the expected utility of the original policy, this is the updated policy. For all states exceot for 1, consume all wealth and invest only the salary. However in state 1, consume no wealth and invest with the same $\omega = 0.75$ as before.

4 Part D

Successive Approximation consists of the following steps:

1. Set $n := 0$, choose $\epsilon > 0$, and initialize $v_0(i) = 0$
2. Compute

$$
v_{n+1}(i) := \max_{a \in \mathcal{A}_i} \left\{ r^a(i) + \alpha \sum_{j \in \mathcal{I}} p^a(i,j) v_n(j) \right\}
$$

   and let

$$
f_{n+1}(i) \in \arg\max_{a \in \mathcal{A}_i} \left\{ r^a(i) + \alpha \sum_{j \in \mathcal{I}} p^a(i,j) v_n(j) \right\}
$$

3. Let

$$
M_n := \max_{i \in \mathcal{I}} \{ v_n(i) - v_{n-1}(i) \}
$$

and

$$m_n := \min_{i \in \mathcal{I}} \{v_n(i) - v_{n-1}(i)\}$$

If $M_n - m_n < \epsilon$, stop the algorithm. Otherwise, set $n := n + 1$ and repeat steps 2 and 3.

For this problem, we will set the following parameters:

- $\alpha = 0.99$
- $\epsilon = 4$
- $v_0(i) = 0$ for all $i \in \mathcal{I}$

We also add a minimum number of iterations to the algorithm, so that we do not stop too early. The algorithm will stop when the maximum change in the value function is less than ϵ AND when the minimum number of iterations is reached.

Using the technique, we have found that the optimal policy for these parameters is

$$[(0,0),(1,0),(1,0),(1,0),(1,0),(1,2),(1,3),(1,3),(1,4),(0.5,4),(0,4)]$$

Something interesting about it is that because we set our discount factor α to 0.99, we weigh the future rewards very highly. this caused our policy to take more risk with a higher ω and lower consumption in the lower states, to maximise the expected utility in the future.

In fact on further investigation, we also tried a discount factor of 0.1, the other extreme case where current rewards are weighed much higher than future rewards. The policy we got was exactly the same as the one we assumed in part B. Probably because we are instead concerned with immediate consumption rather than wealth accumulation for future consumption.

$$[(0,0),(0,0),(0,2),(0,3),(0,4),(0,5),(0,6),(0,7),(0,8),(0,9),(0,10)]$$

5 Part E

In order to ensure that the probability of consuming nothing is less than 0.2, we will have to make some modifications to our successive approximation algorithm.

1. Firstly, we have to add a function to calculate the probability of being in each state based on the policy.
2. Then we have to sum up the probabilities of being in a state where the policy is to consume nothing. This is the probability of consuming nothing.
3. If we are indeed in a state where the probability of consuming nothing is greater than 0.2, we will have ignore this iteration of v and therefore f.

By following the above steps, we do not consider any policy where our probability of consuming nothing is greater than 0.2. Hence we can safely run this algorithm and get a policy that satisfies the condition. Such that the superhuman does not get mad at us!

6 Appendix

Listing 1: Python Implementation of Transition Probabilities

```
import math
import numpy as np
from rich import print

u=0.8
d=-0.3
rf=1.02

rru=(1+u)*rf
rrd=(1+d)*rf
investment_outcome = [rru, rrd]

states = [i for i in range(11)]
omega = [0, 0.25, 0.5, 0.75, 1]
salary = [1,2]

def next_state(state, omega, salary, consumption):
    possible_states = [0 for i in range(11)]
    if consumption > state:
        return None
    before_investment = state + salary - consumption
    for i in investment_outcome:
        next_state = math.floor(before_investment * ((1-omega) * rf +omega * i))
        if next_state < 0:
            next_state = 0
        if next_state > 10:
            next_state = 10
        possible_states[next_state] += 1
    return possible_states
#possible state transitions, print a seperate one for each omega and each consumption

for ohm in omega:
    for c in range(11):
        transitions = []
        for i in states:
            state_row = [0 for i in range(11)]

            for s in salary:
                next_states = next_state(i, ohm, s, c)
                if next_states is not None:
                    state_row = np.add(state_row, next_states)
            if sum(state_row) > 0:
                state_row = np.divide(state_row, sum(state_row))
            transitions.append(state_row)
        print(f"Consumption {c} and omega {ohm}")
        print(np.array(transitions))
        print()
```

Listing 2: Python Implementation of Transition Probabilities

```
n = 0
alpha = 0.99
epsilon = 4
v0 = [0 for i in range(11)]
Mn = math.inf
mn = 0
# define the actions
actions = [(i, y) for i in (0, 0.25, 0.5, 0.75, 1) for y in range(11)]
fn = []

def reward(c):
    if c == 0:
        return 0
    else:
        return max(math.log(c),0)

# successive approximation, added a minimum number of iterations
# to ensure convergence
```

```
    while Mn - mn >= epsilon or n < 1000:
        Mn = 0
        mn = math.inf
        v1 = [0 for i in range(11)]
        f = [0 for i in range(11)]
        for ohm, c in actions:
            for i in range(11):
                # amount to consume is i
                if c > i:
                    continue
                v = reward(c) + alpha * sum([results[(ohm, c)][i][j] * v0[j] for j in range(11)])
                if v > v1[i]:
                    v1[i] = v
                    f[i] = (ohm, c)
                # print(f"State {i}, Action {ohm, c}: Value {v1[i]}")
                if v1[i] > Mn:
                    Mn = v1[i]
                if v1[i] < mn:
                    mn = v1[i]

        if n % 100 == 0:
            print(f"Iteration␣{n}:")
            print(f"Max␣Value:␣{Mn},␣Min␣Value:␣{mn}")
            print(f"Optimal␣action:␣{f}")
            print(f"Value␣function:")
            pp.pprint(v1)

        v0 = v1
        n += 1

    print(f"Final␣Value␣Function:␣{v1}")
    print(f"Optimal␣Actions:␣{f}")
```