Artificial Neural Networks

RL , Fuzzy Logic

Homework #5

Poorya MohammadiNasab

(400722138)

Contents

Problem 1

In this section, you need to provide an MDP (Markov Decision Process) model. It should be noted that you need to determine states, actions, state transition probabilities, and rewards for your model. (35pts)

1 - A)

In a village, we want to make a decision at the beginning of each month whether the sale of shrimps is allowed or not. Every time we decide to sell shrimps, the number of shrimps will be reduced and we gain a profit from the sale of them. It should be noted that if the population of shrimps is reduced too much, it costs us a lot of money to compensate for their population, otherwise, the whole shrimp industry in this village will go broke.

In an MDP, an agent interacts with an environment by taking actions and seek to maximize the rewards the agent gets from the environment. At any given time stamp t, the process is as follows:

- The environment is in state St
- The agent takes an action At
- The environment generates a reward Rt based on St and At
- The environment moves to the next state St+1

To express a problem using MDP, one needs to define the followings:

- **States** of the environment
- **Actions** the agent can take on each state
- **Rewards** obtained after taking an action on a given state
- **State transition** probabilities.

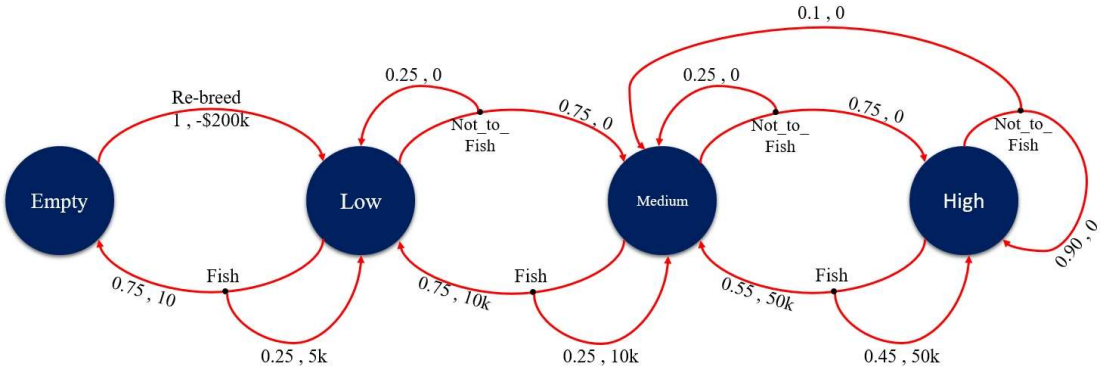
We need to decide what proportion of salmons to catch in a year in a specific area maximizing the longer term return. Each salmon generates a fixed amount of dollar. But if a large proportion of salmons are caught then the yield of the next year will be lower. We need to find the optimum portion of salmons to catch to maximize the return over a long time period. This problem can be expressed as an MDP as follows:

- **States:** The number of salmons available in that area in that year. For simplicity assume there are only four states; empty, low, medium, high. The four states are defined as follows:
    - **Empty:** no salmons are available.
    - **Low**: available number of salmons are below a certain threshold (**t1**)
    - **medium:** available number of salmons are between (**t1**), and (**t2**)
    - **high:** available number of salmons are more than (**t2**)

- **Actions:** For simplicity assumes there are only **two** actions; **fish** and **not_to_fish**. Fish means catching certain proportions of salmon. For the state empty the only possible action is not_to_fish.

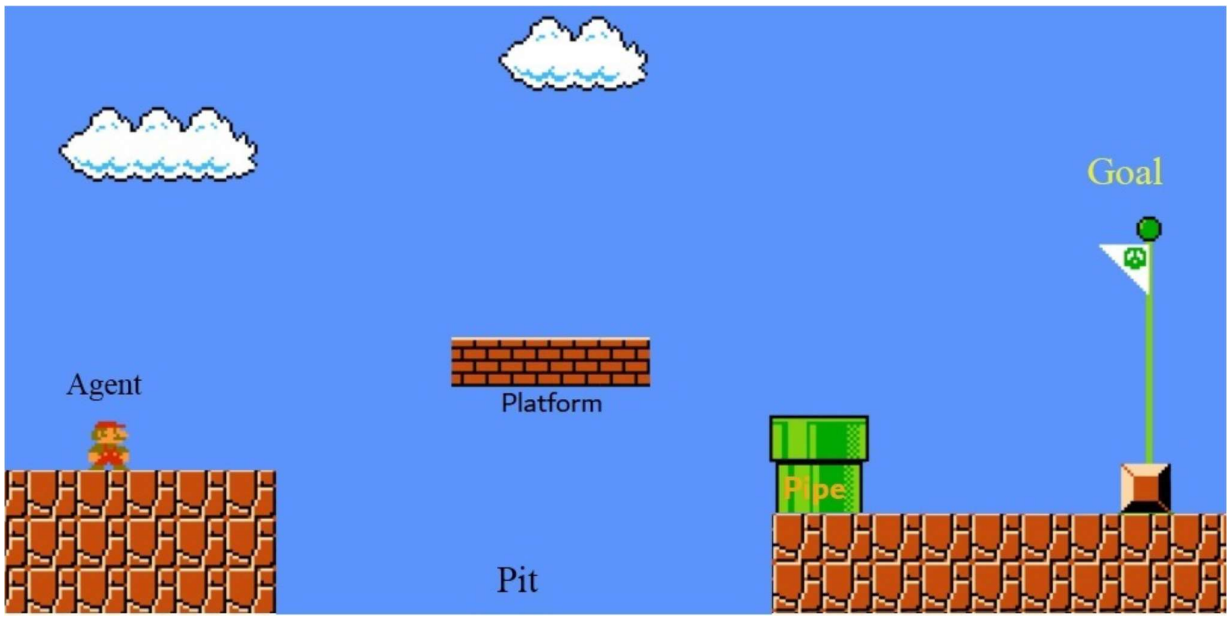
- **Rewards**: Fishing at certain state generates rewards, let's assume the rewards of fishing at state low, medium and high are \$5K, \$50K and \$100k respectively. If an action takes to empty state then the reward is very low -\$200K as it require re-breeding new salmons which takes time and money.

- **State Transitions**: Fishing in a state has higher a probability to move to a state with lower number of salmons. Similarly, not_to_fish action has higher probability to move to a state with higher number of salmons (except for the state high).



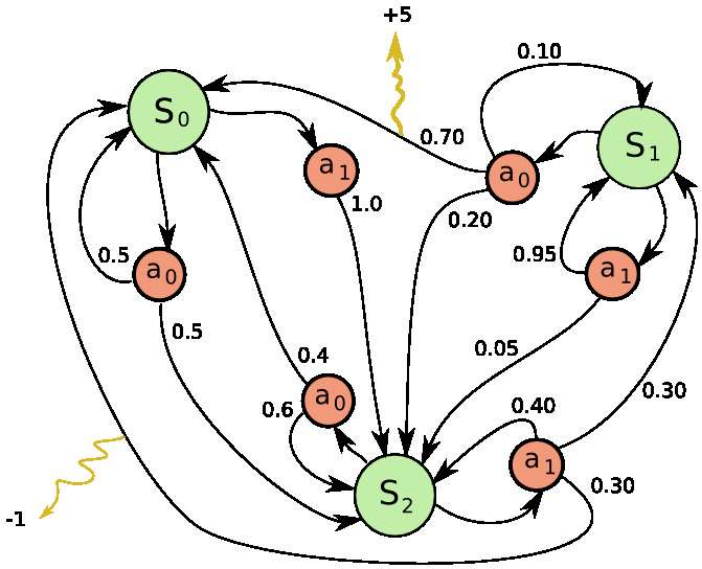
1 - B)

In the Mario game, the goal is to reach the end flag without falling in the pit or dying by enemies. Assume that Mario (our agent) can either jump or move forward. The speed of Mario affects its jump distance, for instance, if he jumps at high speed, he may slip off the edge of the platform and fall (either in the pit or on the green pipe), or if the speed is too low, he can't reach the platform after jumping. A piranha plant will also come out of the pipe stochastically and kill Mario if it hits him. The game ends whether Mario gets killed or reaches the flag.



For the representation of Super Mario Bros. as a Markov decision process (MDP), we use a 3 × 3 grid of variables that code terrain information. This grid can recognize platforms, coin blocks, and coins. Our MDP also accounts for the position (in X and Y axes) of the two enemies, closest to Mario. The position variables are discretized into 7 values that range from 0 to 6. We also added a binary variable that detects whether there is a cliff in front of Mario. Our bot can perform 10 different actions. Mario performs the next actions for both the left and right side of the screen: the actions walk, run, jump, and quick jump. Furthermore, Mario can do nothing, as well as perform a neutral jump.

For the speed trait, we designed RFs that create bots that advance either very quickly or slowly in the level. Regarding how the bot overcomes enemies, we crafted RFs that made the bot very cautious about enemies (i.e., it avoided them as much as possible) or highly likely to kill them. Additionally, we designed RFs that create bots that either collect as many coins as possible or ignore them. Finally, for the jump preference, some bots jump as much as possible, while others avoid jumping. With the aforementioned base play style as a foundation, we combined them to create 11 unique bot behaviors for MarioMix.

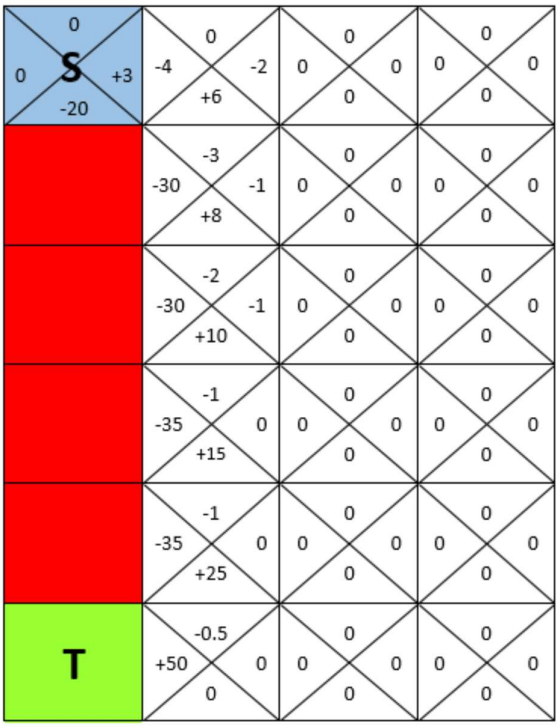


Problem2

Imagine our agent wants to go from state S to T. State T has a reward of +120 and states with red color have a reward of -90. Taking each step has a -1 reward. Run each episode with the following actions and update values by Q-Learning algorithm:

- Episode 1: Right, Down, Down, Down, Down, Down, Left.
- Episode 2: Right, Down, Down, Left.

Note that if the agent goes into a red-colored state or state T, the episode terminates. Set α = 0.9 and λ = 0.8. (25pts)



$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \lambda \, max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

EPISODE 1

$$R: Q(s_t, a_t) = 3 + 0.9\big((-1) + 0.8(6) - 3\big) \rightarrow Q(s_t, a_t) = 3.72$$

$$D: Q(s_t, a_t) = 6 + 0.9\big((-1) + 0.8(8) - 6\big) \rightarrow Q(s_t, a_t) = 5.46$$

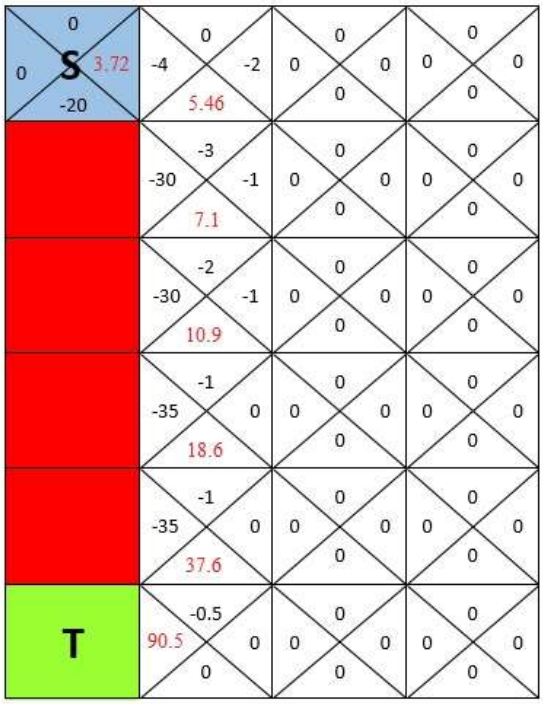
$$D: Q(s_t, a_t) = 8 + 0.9\big((-1) + 0.8(10) - 8\big) \rightarrow Q(s_t, a_t) = 7.1$$

$$D: Q(s_t, a_t) = 10 + 0.9\big((-1) + 0.8(15) - 10\big) \rightarrow Q(s_t, a_t) = 10.9$$

$$D: Q(s_t, a_t) = 15 + 0.9\big((-1) + 0.8(25) - 15\big) \rightarrow Q(s_t, a_t) = 18.6$$

$$D: Q(s_t, a_t) = 25 + 0.9\big((-1) + 0.8(50) - 25\big) \rightarrow Q(s_t, a_t) = 37.6$$

$$L: Q(s_t, a_t) = 50 + 0.9\big((-1) + 0.8(120) - 50\big) \rightarrow Q(s_t, a_t) = 90.5$$



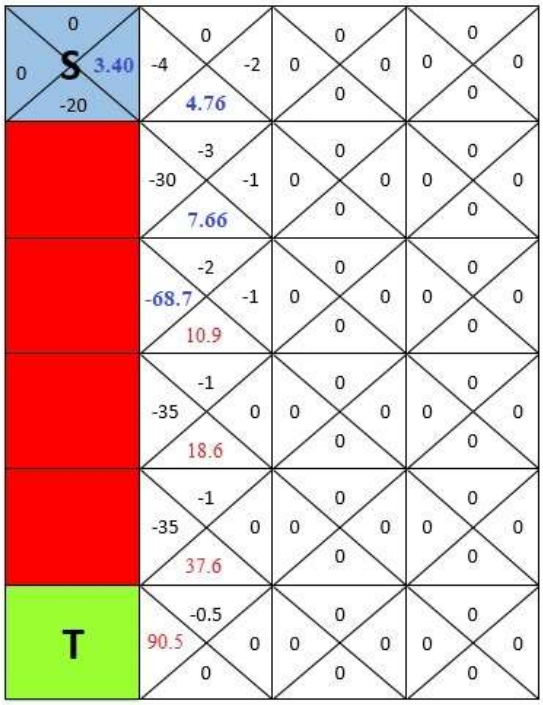
EPISODE 2

$$R: Q(s_t, a_t) = 3.72 + 0.9\big((-1) + 0.8(5.46) - 3.72\big) \rightarrow\ Q(s_t, a_t) = 3.40$$

$$D: Q(s_t, a_t) = 5.46 + 0.9\big((-1) + 0.8(7.1) - 5.46\big) \rightarrow\ Q(s_t, a_t) = 4.76$$

$$D: Q(s_t, a_t) = 7.1 + 0.9\big((-1) + 0.8(10.9) - 7.1\big) \rightarrow\ Q(s_t, a_t) = 7.66$$

$$L: Q(s_t, a_t) = -30 + 0.9\big((-1) + 0.8(-90) + 30\big) \rightarrow\ Q(s_t, a_t) = -68.7$$



Problem4

We want to model a fuzzy controller in this part, the fuzzy controller will be for a steam turbine.

- Inputs: temperature and pressure (5 descriptors each)
- Output: throttle setting (7 descriptors)

After modeling the fuzzy controller answer this question.

"If for inputs temperature is 70% and pressure is 30% determine the throttle position."

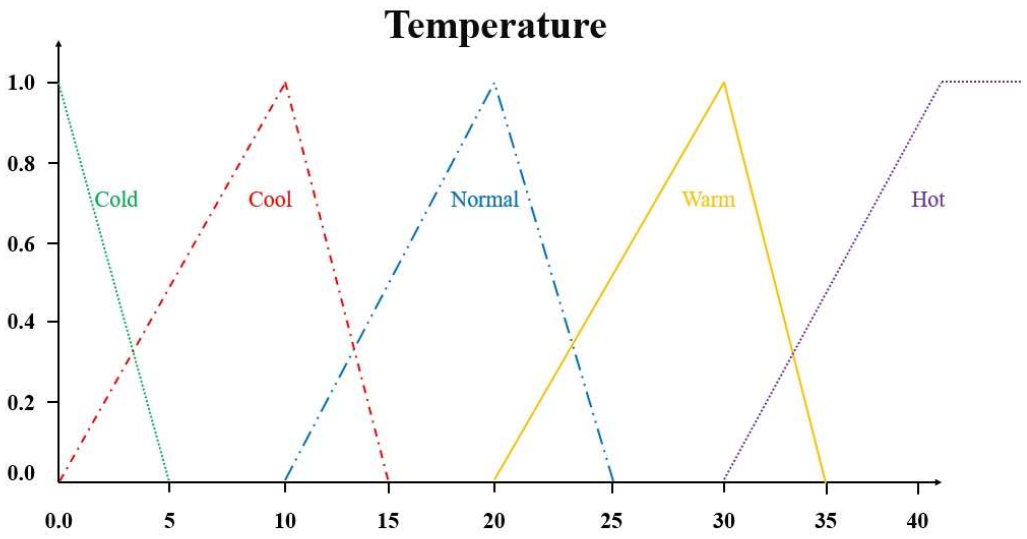
- **Step 1:** Identification of variables
  - **Inputs:** Temperature and pressure
  - **Output:** Throttle setting of steam turbine

- **Step 2:** Fuzzy subset configuration
  Assign a linguistic descriptor for each fuzzy subset

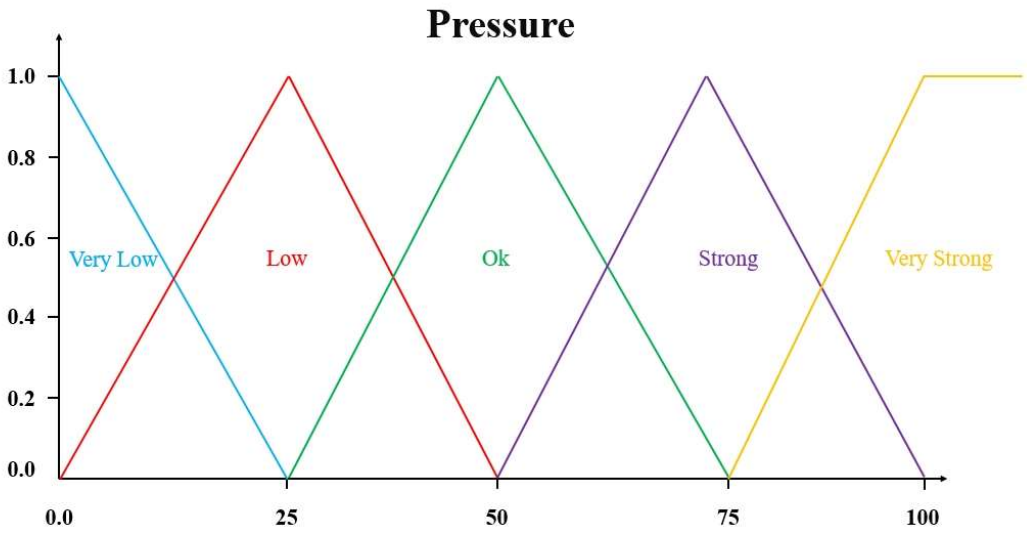
  - **Temperature:** Cold, Cool, Normal, Warm, Hot
  - **Pressure:** Very Low, Low, Ok, Strong, Very Strong
  - **Throttle Setting:**
    - **N3:** Very Large Negative
    - **N2:** Large Negative
    - **N1:** Negative
    - **Z:** Zero
    - **P1:** Positive
    - **P2:** Large Positive
    - **P3:** Very Large Positive

- **Step 3:** Obtain Membership Function

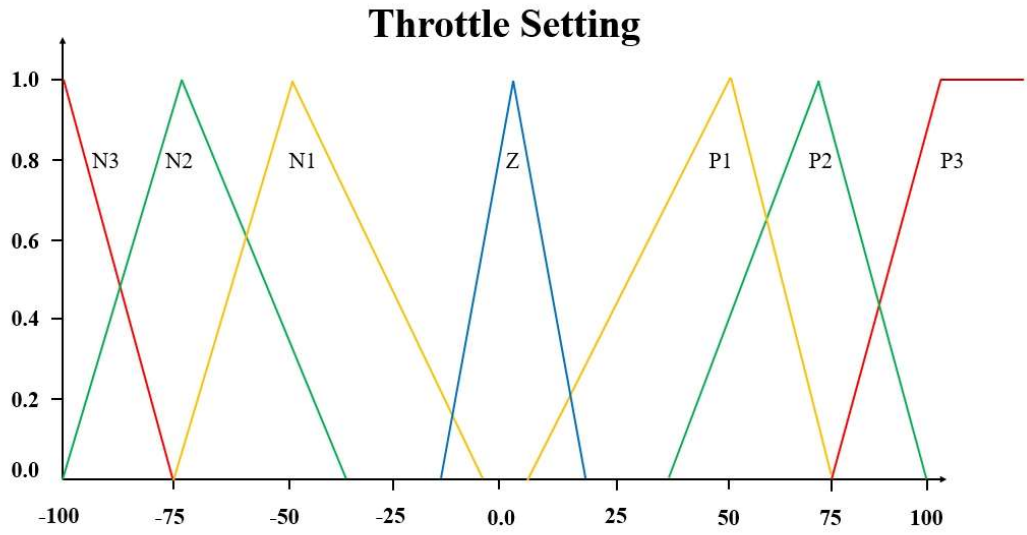
  Define membership functions for descriptors. The fuzzy membership function for temperature is depicted in following figure:



The fuzzy membership function for Pressure is depicted in following figure:



The fuzzy membership function for Throttle Setting is depicted in following figure:



- **Step 4:** Identification of output
  We can derive following rules,
  - **Rule 1:** If temperature is cool and pressure is low, then throttle setting is P2
  - **Rule 2:** If temperature is cool and pressure is ok, then throttle setting is Z
  - **Rule 3:** If temperature is normal and pressure is low, then throttle setting is P2
  - **Rule 4:** If temperature is normal and pressure is ok, then throttle setting is Z

For inputs temperature is 70% and pressure is 30%, **rule 4** will be fired.

$x_1 \rightarrow Temprature, x_2 \rightarrow Pressure$

$\mu_{x_1} = 0.85\,, x_1^{avg} = 30$

$\mu_{x_2} = 0.4\,, x_2^{avg} = 50$

$x^* = \frac{x_1^{avg}\mu_{x_1} + x_2^{avg}\mu_{x_2}}{x_1^{avg} + x_2^{avg}}$ $\quad x^* = \frac{0.85*30+0.4*50}{30+5} = \frac{25.5+20}{80} = 0.57 \rightarrow Defuzzification \rightarrow$ Throttle setting = **-5**

References

1) www.youtube.com/watch?v=1XRahNzA5bE

2) codecrucks.com/designing-fuzzy-controller-step-by-step-guide/

3) towardsdatascience.com/real-world-applications-of-markov-decision-process-mdp-a39685546026

4) Arzate, Christian & Igarashi, Takeo. (2021). MarioMix: Creating Aligned Playstyles for Bots with Interactive Reinforcement Learning.

5) www.comp.nus.edu.sg/~rishav1/blog/2016/mario-bros-RL/