Inf368 – project 5 – Problems

Exercise 1 – Stochastic Policies

1. Define the MDP underlying the above environment.

To define the Markov Decision Process (MDP) underlying the given environment, we need to specify the following components:

States: These are the different situations or configurations the agent can be in. Based on the description provided, the states in this environment are:

> S = {Empty Bowl, Full Bowl, Sugared Bowl, Salted Bowl, Sugar Sampling, Salt Sampling, Baked Sugary Cookies (Terminal), Baked Salty Cookies (Terminal)}

Actions: These are the actions the agent can make in each state ($A(s)$). From the description, the actions available in each state are as follows:

> A (Empty Bowl) = {Wait, Prepare Dough}
>
> A (Full Bowl) = {Wait, Add Sugar, Add Salt}
>
> A (Sugared Bowl) = {Taste, Cook, Restart}
>
> A (Salted Bowl) = {Taste, Cook, Restart}
>
> A (Sugar Sampling) = {Cook, Return to Bowl}
>
> A (Salt Sampling) = {Cook, Return to Bowl}
>
> A (Baked Sugary Cookies) = {} (Terminal state has no action)
>
> A (Baked Salty Cookies) = {} (Terminal state has no action)

Transitions: Transitions represents the probabilities of transitioning from one state to another given an action. Based on the description, the transitions are deterministic for all actions. Meaning a given a state and an action it will a hundred percent go to the same other state.

> From Empty Bowl:
>
> $$P(Empty\ Bowl \mid Empty\ Bowl, Wait) = 1$$
>
> $$P(Full\ Bowl \mid Empty\ Bowl, Prepare\ Dough) = 1$$
>
> From Full Bowl:
>
> $$P(Full\ Bowl \mid Full\ Bowl, Wait) = 1$$
>
> $$P(Sugared\ Bowl \mid Full\ Bowl, Add\ Sugar) = 1$$

$$P(Salted\ Bowl \mid Full\ Bowl, Add\ Salt) = 1$$

From Sugared Bowl:

$$P(Sugar\ Sampling \mid Sugared\ Bowl, Taste) = 1$$

$$P(Baked\ Sugary\ Cookies \mid Sugared\ Bowl, Cook) = 1$$

$$P(Full\ Bowl \mid Sugared\ Bowl, Restart) = 1$$

From Salted Bowl:

$$P(Salt\ Sampling \mid Salted\ Bowl, Taste) = 1$$

$$P(Baked\ Salty\ Cookies \mid Salted\ Bowl, Cook) = 1$$

$$P(Full\ Bowl \mid Salted\ Bowl, Restart) = 1$$

From Sugar Sampling:

$$P(Baked\ Sugary\ Cookies \mid Sugar\ Sampling, Cook) = 1$$

$$P\ (Sugared\ Bowl \mid Sugar\ Sampling, Return\ to\ Bowl) = 1$$

From Salt Sampling:

$$P\ (Baked\ Salty\ Cookies \mid Salt\ Sampling, Cook) = 1$$

$$P\ (Salted\ Bowl \mid Salt\ Sampling, Return\ to\ Bowl) = 1$$

Rewards: These are the rewards obtained when transitioning from one state to another. Terminal states give fixed rewards:

Baked Sugary Cookies: Reward = +10

Baked Salty Cookies: Reward = -10

All other transitions have a reward of 0.

Discount Factor: This represents the agent's preference for immediate rewards over future rewards.

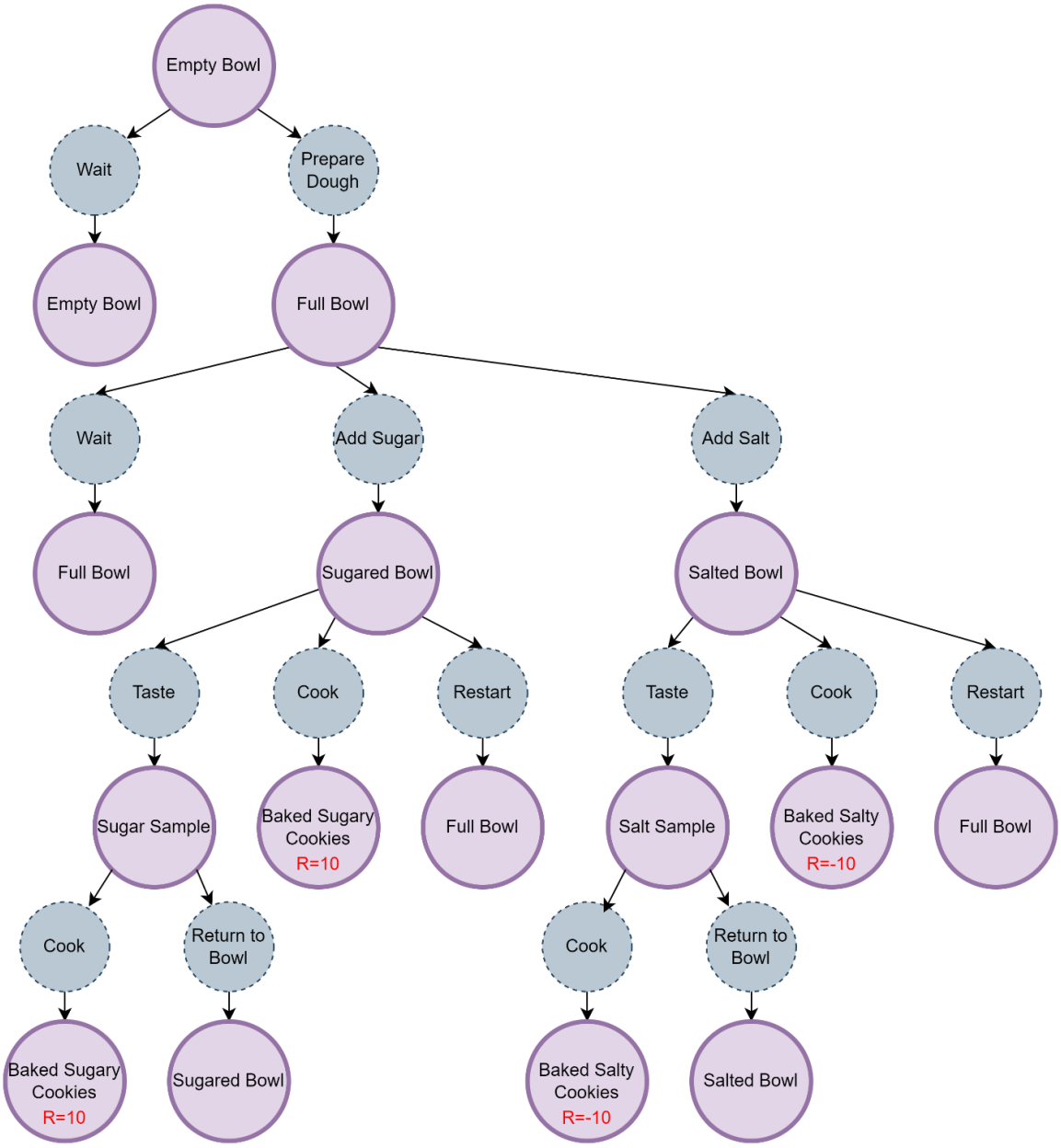
Given the above components, we can formally define the MDP $M = (S, A, P, R, \gamma)$:

- S: Set of states as described above.
- A: Set of actions as described above.
- P: Transition probabilities, which are deterministic in this case.
- R: Reward function, where $R(s, a, s')$ is 0 for all non-terminal transitions and +10 or -10 for terminal transitions as described above.

- γ: Discount factor, $\gamma \in [0,1]$ and affects the agent's preference in the time of the rewards.

With these components defined, we have a formal representation of the MDP underlying the cookie-making environment.

2. Draw out the backup diagram for this environment.



3. (*) Can you define an optimal deterministic policy for this environment? If so, write it out. If not, why not?

Given the environment described, we can define an optimal deterministic policy. In this environment, the ultimate goal is to maximize the reward, and since the rewards are explicitly given for ending in either the "Baked Sugary Cookies" state (+10) or the "Baked Salty Cookies" state (−10), the optimal policy would naturally steer the agent towards actions that lead to the highest reward.

Here is the optimal deterministic policy:

From Empty Bowl: Choose Prepare Dough. (This is because staying in the "Empty Bowl" state does not progress towards earning a reward.)

From Full Bowl: Choose Add Sugar. (Adding sugar leads towards the positive reward state, "Baked Sugary Cookies".)

From Sugared Bowl: To maximize the reward, choose Cook. (This action directly leads to "Baked Sugary Cookies" with a reward of +10.) The actions Taste or Restart do not directly contribute to achieving the maximum reward from this state, although Taste might be part of an exploratory policy in a more complex or less deterministic environment.

From Sugar Sampling: Since the agent would only be in this state if an exploratory or sub-optimal action (Taste from "Sugared Bowl") was chosen: Choose Cook to proceed to "Baked Sugary Cookies" and secure the +10 reward.

From Salted Bowl and Salt Sampling: In the optimal policy context, the agent would avoid these states because they lead to a negative reward. However, if the agent finds itself in Salted Bowl due to exploration or initial conditions: Choosing Restart from "Salted Bowl" could potentially redirect the agent back to "Full Bowl" and then follow the optimal path through "Add Sugar" to eventually reach "Baked Sugary Cookies".

Policy Definition:

Optimal path policy:

- $\pi^*(Empty\ Bowl) = Prepare\ Dough$
- $\pi^*(Full\ Bowl) = Add\ Sugar$
- $\pi^*(Sugared\ Bowl) = Cook$ → reached optimal terminal state $Baked\ Sugary\ Cookies$

Redirect policy to the optimal path (shouldn't be here: only be here if a suboptimal policy was taken or during exploration):

- $\pi^*(Sugar\ Sampling) = Cook$ → reached optimal terminal state $Baked\ Sugary\ Cookies$

- $\pi^*(Salt\ Sampling) = Salted\ Bowl$
- $\pi^*(Salted\ Bowl) = Restart$

This environment allows for the definition of an optimal deterministic policy because the transitions are deterministic and the rewards for terminal states are clearly defined. The optimal policy is straightforward in maximizing the reward by directing the agent towards actions that lead to the positive reward terminal state ("Baked Sugary Cookies") while avoiding actions that lead to the negative reward terminal state ("Baked Salty Cookies"). The deterministic nature of the environment, combined with the clear reward structure, makes it possible to define such an optimal policy.

4. (*) Can you define an optimal stochastic policy for this environment? If so, write it out. If not, why not?

For this specific environment, an optimal stochastic policy is not practical or necessary because 1) Deterministic Nature of Transitions and Rewards: Each action leads to a specific, predictable outcome without variation. Therefore, introducing randomness in the choice of actions (which is what a stochastic policy would do) does not improve the expected outcome. And 2) Clear Preference of Actions: Given the goal is to maximize rewards, and the outcomes of actions are deterministic, there's always a best action to take in every state that leads towards the highest reward (e.g., moving towards "Baked Sugary Cookies" for a reward of +10 as fast as possible).

Strictly set, yes, we can define the optimal stochastic policy, but all the probabilities would be either 100% or 0% and would simply result in the optimal deterministic policy for it to represent the optimal policy π^*. This is due to the fact that the environment has one state that yields a positive reward, and all actions yield deterministic states meaning the environment itself is deterministic.

5. Define the MDP underlying the above environment.

To define the Markov Decision Process (MDP) underlying the given environment, we need to specify the following components:

States: These are the different situations or configurations the agent can be in. Based on the description provided, the states in this environment are:

> S = {Empty Bowl, Full Bowl, Enriched Bowl, Sugar Sampling, Salt Sampling, Baked Sugary Cookies (Terminal), Baked Salty Cookies (Terminal)}

Actions: These are the actions the agent can make in each state ($A(s)$). From the description, the actions available in each state are as follows:

A (Empty Bowl) = {Wait, Prepare Dough}

A (Full Bowl) = {Wait, Add Ingredient}

A (Enriched Bowl) = {Taste, Cook, Restart}

A (Sugar Sampling) = {Cook, Return to Bowl}

A (Salt Sampling) = {Cook, Return to Bowl}

A (Baked Sugary Cookies) = {} (Terminal state has no action)

A (Baked Salty Cookies) = {} (Terminal state has no action)

Transitions: These represent the probabilities of transitioning from one state to another given an action. Based on the description, some transitions are deterministic, and some are based on the latent variable σ. Assuming this variable is a binary (either sugar or salt) the transition probabilities can be written as such:

From Empty Bowl:

$$P(Empty\ Bowl \mid Empty\ Bowl, Wait) = 1$$

$$P(Full\ Bowl \mid Empty\ Bowl, Prepare\ Dough) = 1$$

From Full Bowl:

$$P(Full\ Bowl \mid Full\ Bowl, Wait) = 1$$

$$P(Enriched\ Bowl \mid Full\ Bowl, Add\ Ingredinet) = 1$$

From Enriched Bowl:

$$P(\ Salt\ Sampling \mid \text{Enriched } Bowl, Taste\ ) = \begin{cases} 1 & if\ \sigma\ is\ salt \\ 0 & otherwise \end{cases}$$

$$P(\ Sugar\ Sampling \mid \text{Enriched } Bowl, Taste\ ) = \begin{cases} 1 & if\ \sigma\ is\ sugar \\ 0 & otherwise \end{cases}$$

$$P(Baked\ Salty\ Cookies \mid \text{Enriched } Bowl, Cook) = \begin{cases} 1 & if\ \sigma\ is\ salt \\ 0 & otherwise \end{cases}$$

$$P(Baked\ Sugary\ Cookies \mid \text{Enriched } Bowl, Cook) = \begin{cases} 1 & if\ \sigma\ is\ sugar \\ 0 & otherwise \end{cases}$$

$$P(Full\ Bowl \mid \text{Enriched } Bowl, Restart) = 1$$

From Sugar Sampling:

$$P(Baked\ Sugary\ Cookies \mid Sugar\ Sampling, Cook) = 1$$

$$P\ (Sugared\ Bowl \mid Sugar\ Sampling, Return\ to\ Bowl) = 1$$

From Salt Sampling:

$$P\ (Baked\ Salty\ Cookies \mid Salt\ Sampling, Cook) = 1$$

$$P\ (Salted\ Bowl \mid Salt\ Sampling, Return\ to\ Bowl) = 1$$

Rewards: These are the rewards obtained when transitioning from one state to another. Terminal states provide fixed rewards:

Baked Sugary Cookies: Reward = +10

Baked Salty Cookies: Reward = -10

All other transitions have a reward of 0.

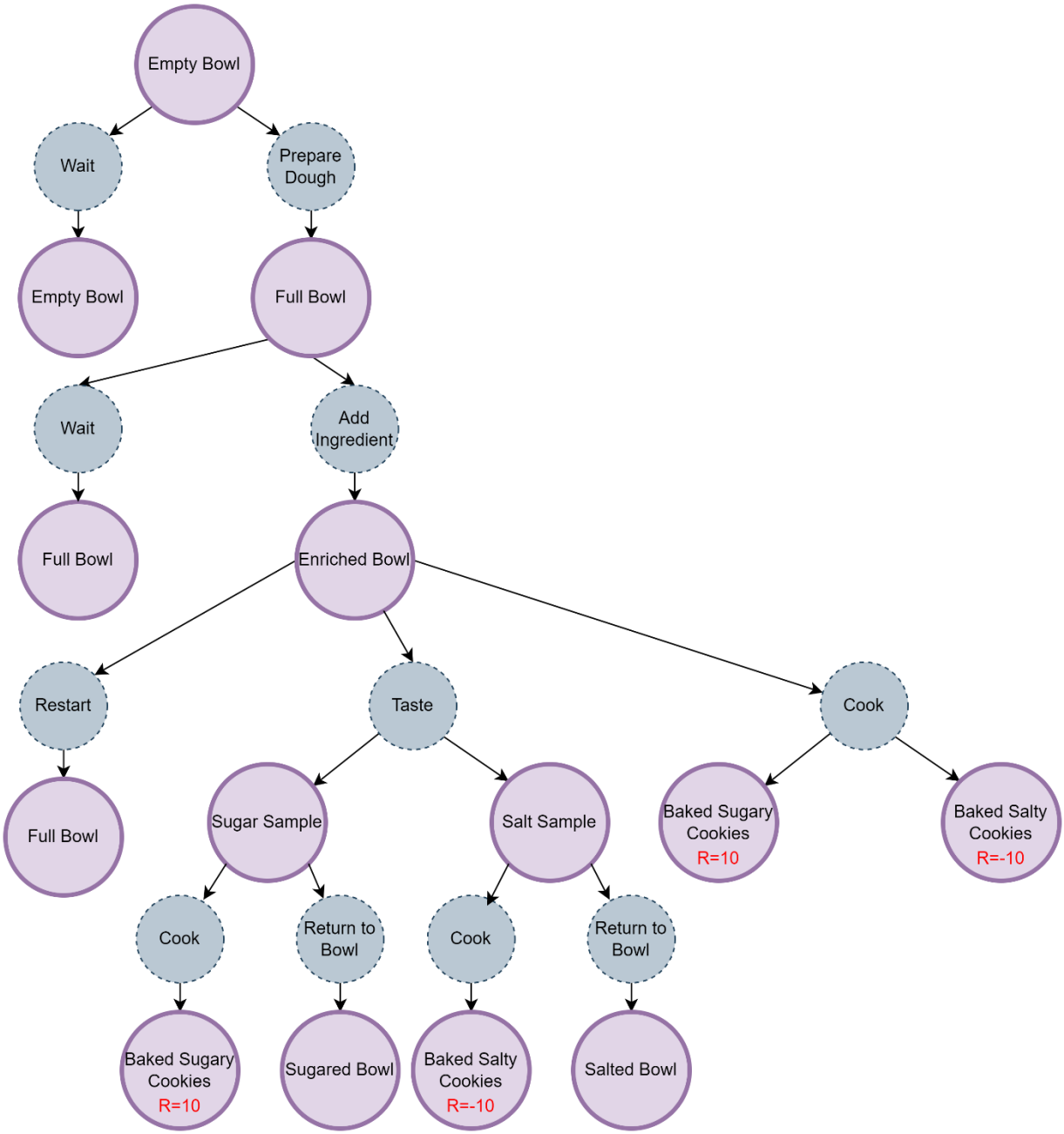
Discount Factor: This represents the agent's preference for immediate rewards over future rewards.

Given the above components, we can formally define the MDP $M = (S, A, P, R, \gamma)$:

- S: Set of states as described above.
- A: Set of actions as described above.
- P: Transition probabilities, which are deterministic in this case.
- R: Reward function, where $R(s, a, s')$ is 0 for all non-terminal transitions and +10 or -10 for terminal transitions as described above.
- γ: Discount factor, $\gamma \in [0,1]$ and affects the agent's preference in the time of the rewards.

With these components defined, we have a formal representation of the MDP underlying the cookie-making environment.

6. Draw out the backup diagram for this environment.



7. (*) Can you define an optimal deterministic policy for this environment? If so, write it out. If not, why not?

In the altered environment where the agent cannot visually discriminate between salt and sugar, defining an optimal deterministic policy becomes more challenging. The introduction of the

latent variable σ, which determines whether the added ingredient is sugar or salt, adds uncertainty to the environment, making it difficult to determine a single deterministic action that guarantees the highest reward in all situations.

The agent cannot directly observe whether the ingredient added to the dough is sugar or salt. The latent variable σ determines this, but it is unknown to the agent. Without knowledge of σ, it is impossible to determine a single deterministic action that always leads to the optimal outcome. The outcome of actions such as "Taste" or "Cook" in the "Enriched Bowl" state depends on the value of σ. If σ is sugar, the agent should choose actions that lead to "Baked Sugary Cookies", and if σ is salt, the agent should choose actions leading to "Baked Salty Cookies". However, since the agent does not know the value of σ, it cannot determine the optimal action with certainty.

Due to the uncertainty introduced by the latent variable σ, it is not possible to define a single optimal deterministic policy that guarantees the highest reward in all situations. The optimal action in each state depends on the value of σ, which is not observable to the agent. Without knowledge of σ, the agent cannot make deterministic decisions that lead to the optimal outcome in all cases.

8. (*) Can you define an optimal stochastic policy for this environment? If so, write it out. If not, why not?

Defining an optimal stochastic policy for this environment is possible, considering the uncertainty introduced by the latent variable σ, which determines whether the added ingredient is sugar or salt. A stochastic policy allows the agent to probabilistically select actions based on the observed states and available information. However, it's important to note that while a stochastic policy is possible, it might not be optimal in all cases due to the inherent uncertainty in the environment.

The optimal stochastic policy would assign probabilities to actions based on the current state and the agent's beliefs about the latent variable σ. The policy should aim to maximize the expected reward over time by balancing exploration and exploitation. Here's how the policy may look:

- $\pi(Prepare\ Dough \mid Empty\ Bowl) = 1$

  (deterministic action will always choose prepare dough)

- $\pi(Add\ Ingredient\ (Sugar) \mid Full\ Bowl) = p_{sugar}(probability\ of adding\ sugar)$

- $\pi(Add\ Ingredient\ (Salt) \mid Full\ Bowl) = p_{salt}(probability\ of adding\ salt)$

  ($p_{sugar} + p_{salt} = 1$) the sum of the possible adding ingredient actions will be 1.

- $\pi(Taste \mid Enriched\ Bowl) = 1$

- $\pi(Cook\ (Sugar) \mid Enriched\ Bowl) = p_{sugar}(cooking\ with\ sugar)$

- $\pi(Cook\ (Salt)|\ Enriched\ Bowl) = p_{salt}(cooking\ with\ salt)$

  $(p_{sugar} + p_{salt} = 1)$ the sum of cooking with each ingredient should be 1.

- $\pi(Restart \mid Enriched\ Bowl) = 1$
- $\pi(Cook \mid Sugar\ Sampling) = 1$
- $\pi(Return \mid Salty\ Sampling) = 1$

In the Full Bowl state, the agent probabilistically selects between adding sugar or salt based on its understanding about the likelihood of each ingredient being present. Similarly, in the Enriched Bowl state, the agent selects between cooking sugary or salty cookies based on its understanding about the latent variable σ. By probabilistically selecting actions, the agent can explore different possibilities (e.g., trying both sugar and salt) while still exploiting its current knowledge to maximize reward.

The main challenge in defining an optimal stochastic policy is the uncertainty introduced by the latent variable σ. Without direct observation of σ, the agent's beliefs about the likelihood of sugar or salt being present are based on prior knowledge or observations. While a stochastic policy allows for adaptation to uncertainty, it might not always result in the highest possible reward. The agent's actions are probabilistic, and there's a risk of suboptimal decisions due to imperfect information.

In conclusion, yes, it's possible to define an optimal stochastic policy for this environment, its effectiveness depends on the agent's ability to accurately estimate the likelihood of different outcomes based on available information. The policy aims to balance exploration and exploitation to maximize the expected reward over time, despite the inherent uncertainty introduced by the latent variable σ.

9. (*) How does this setup differ from the previous one?

The setups differ in many ways. Most noticeably in the introduction of the latent variable, which affected the states, and the actions as well as the transition probabilities and resulting policy.

The altered environment introduces a latent variable σ, which determines whether the added ingredient is sugar or salt. This variable is not directly observable to the agent and adds uncertainty to the environment. In the original environment, actions lead to deterministic outcomes, while in the altered environment, actions may lead to multiple possible states depending on the value of σ. In addition, in the original environment, state transitions and rewards are directly observable to the agent, while in the altered environment, the latent variable and its influence on transitions are not observable.

Exercise 2 – Continuous Actions

1. Recall how we defined the neural networks so that the output would give a distribution over the actions.

We have a discrete action space with a finite number of actions, each of these actions corresponds to an output neuron of a neutral network. For the neural network to output a distribution over the actions, the network is set up to output raw values for each action and then a softmax function is applied to the raw values to convert these values into probabilities. The softmax function is defined as:

$$P(a_i) = \frac{e^{z_i}}{\sum_{k=1}^{n} e^{z_k}}$$

2. Suggest a possible distribution function to model your policy.

Now, for a continuous action space, a good distribution function to model out policy would be to use the Gaussian distribution because it provides probability density for every point in the continuous space.

3. Consider what sort of parameters or statistics you would need to work with the distribution you have chosen.

The Gaussian distribution uses the parameters mean and the standard deviation, in this scenario the mean represents the most likely action to be chosen and the standard deviation shows how much exploration is around that specific action.

4. (*) How would you define your neural network to have a distribution over $\mathbb{R}$ in output?

Here is how we defined our neural network:

We first defined the input layer, that takes in the state representation, which is a vector containing values that described the current situation.

Then we have the hidden layers that process the input. There are several hidden layers that we will use, with the activation function ReLU (Rectified Linear Unit) to introduce non-linearity. Each of these hidden layers transforms the representation of the state to help the network learn complex patterns.

After we have an output layer 1 which outputs the mean, a real number, another output layer 2 that outputs the log standard deviation then applies the exponential function to get the standard deviation. The output layer 1 has one neuron as the action space is one dimensional. The neuron's output is the mean of the Gaussian distribution, in this section there is no activation function needed since the mean is a real number. In the output layer 2, it also has only one

neuron because it is a one-dimensional action space. To make sure that the output, the standard deviation is always positive, there is added an exponential function to the neuron's output.

5. What assumptions underlie your choice for the probability distribution?

In choosing the Gaussian distribution for our model, we're assuming:

- We expect that actions will have one most likely value, and as we move away from this value, they become less likely.
- The way the environment reacts to different actions is steady and predictable.
- There are no off-limits actions; any action we can think of is allowed.
- If we make small tweaks to our actions, the results will change just a little bit—no big surprises.

These assumptions are important for our model to work right. If they don't match up with how things work, we might need to use a different model that fits better with the real-life action space and how actions actually affect things.

Exercise 3 – MAB and Baselines

1. (*) Which term in this expression plays a role analogous to a baseline in policy gradient algorithms?

In policy gradient algorithms, the term analogous to the baseline is one that is subtracted from the reward to compute the advantage or the advantage estimate. The advantage estimate indicates how much better or worse an action is compared to the expected reward.

In the given expression, the term $\hat{E}[R]$ serves a role like the baseline. Here's why:

Expected Reward Estimate ($\hat{E}[R]$):

The term $\hat{E}[R]$ represents an estimate of the expected value of the reward. This estimate can be obtained through various methods, such as using a moving average or other statistical techniques. When subtracted from the actual reward R^t, it provides an estimate of the advantage or disadvantage of selecting action a_i at time t.

Subtracting $\hat{E}[R]$ helps in reducing the variance of the updates. By subtracting a baseline, which represents the average or expected return, the update focuses more on the relative advantage of taking an action, rather than the absolute magnitude of the reward.

2. (*) How does this term affect the learning process in the gradient bandit algorithm?

This baseline term can affect gradient policy algorithms in many ways:

Variance Reduction:

By subtracting the estimated expected reward ($\hat{E}[R]$) from the actual reward (R^t), the algorithm focuses on the relative advantage or disadvantage of selecting an action compared to the expected return. This helps in reducing the variance of the updates. Without the baseline, the updates would be influenced by the absolute scale of the rewards, which could lead to high variance in the gradient estimates. With the baseline, the updates primarily capture the difference between the observed reward and the expected reward, leading to more stable and consistent learning.

Bias Control:

Introducing a baseline can potentially introduce bias into the gradient estimates. However, the bias introduced by using a baseline is usually less significant compared to the reduction in variance. The bias introduced by the baseline can often times be acceptable because it helps in stabilizing the learning process and often leads to faster convergence.

Learning Efficiency:

By focusing on the relative advantage of actions, rather than the absolute magnitude of rewards, the algorithm can learn more efficiently. This is particularly important in scenarios where the scale of rewards varies widely or when dealing with noisy or stochastic environments. The use of a baseline allows the algorithm to converge faster and more reliably to optimal policies.

Impact on Exploration:

The presence of a baseline can influence the exploration-exploitation trade-off in the learning process. By reducing the variance of the updates, the algorithm may become more conservative in exploring new actions, especially when the estimated expected reward is high. However, this effect depends on the specific baseline used and its interaction with the exploration strategy of the algorithm.

In summary, incorporating the term $\hat{E}[R]$ as a baseline in the gradient bandit algorithm primarily affects the learning process by reducing the variance of the updates, controlling bias, improving learning efficiency, and influencing the exploration-exploitation trade-off. Overall, it leads to more stable and effective learning in the context of multi-armed bandit problems.

Exercise 4 – Actor-critic methods

In which sense actor and critic may be assimilated to these two processes of reinforcement learning?

Policy Evaluation and the Critic:

> **Policy evaluation** in reinforcement learning refers to the process of determining the value function of a given policy, which quantifies how good each state or state-action pair is under that policy. The value function indicates the expected return an agent would receive starting from a particular state (or state-action pair) and following a specific policy thereafter. In the actor-critic framework, the **critic** plays a role similar to policy evaluation. The critic evaluates the actions taken by the actor by estimating the value function. It does so by learning from the observed rewards and states experienced during interactions with the environment. The critic's objective is to estimate the value of being in a particular state and taking a specific action, which helps in assessing the goodness of the actions chosen by the actor.

Policy Improvement and the Actor:

> **Policy improvement** involves selecting actions that are expected to yield better returns or improve the agent's performance. This process is typically guided by the value function estimates obtained during policy evaluation. In actor-critic methods, the **actor** is responsible for policy improvement. It selects actions based on the estimated value function provided by the critic. The actor's objective is to adjust its policy to maximize the expected return or reinforce actions that lead to better outcomes. The actor's policy may be represented explicitly, such as a parameterized policy in the form of a neural network and updated using techniques like gradient ascent to maximize the expected return.

Assimilation of Actor and Critic to Policy Evaluation and Policy Improvement:

> Just as policy evaluation (estimating the value function) informs policy improvement (selecting better actions), the critic in actor-critic methods evaluates the actions taken by the actor based on the observed rewards and states. It provides feedback on the quality of the actions, akin to policy evaluation. Similarly, the actor uses the information provided by the critic to improve its policy by selecting actions that are expected to yield better returns. This mirrors the process of policy improvement.

In summary, in actor-critic methods, the critic performs a role akin to policy evaluation by estimating the value function, while the actor functions similarly to policy improvement by selecting actions based on the critic's feedback to maximize the expected return. This analogy helps in understanding how actor-critic methods combine these two essential components of reinforcement learning.