Problem set 5

Lars, Runar and Årne

March 2024

1 Stochastic Policies

Consider an agent learning to make cookies. The underlying environment is described in the following way. The agent starts in state empty bowl where it has two actions: wait, which will leave the agent in the same state; and prepare dough which deterministically transfer the agent to the full bowl state. In state full bowl the agent can choose between the following actions: wait, which will leave the agent in the same state; add sugar which deterministically transfer the agent to the sugared bowl state; or add salt which deterministically transfer the agent to the salted bowl state. In state sugared bowl the agent can choose between the following actions: taste which deterministically makes the robot get a sample of dough and analyze its con- tent in state sugar sampling; cook which deterministically transfer the agent to the baked sugary cookies state; and restart which deterministically makes the robot dis- card the current dough and produce a new one, thus leading back to full bowl. From the state sugar sampling the agent can either choose the action cook which deterministi- cally transfer the agent to the baked sugary cookies state; or the action return to bowl, which goes back to state sugared bowl. Symmetrically, in state salted bowl the agent can choose between the following ac- tions: taste which deterministically makes the robot get a sample of dough and analyze its content in state salt sampling; cook which deterministically transfer the agent to the baked salty cookies state; and restart which deterministically makes the robot dis- card the current dough and produce a new one, thus leading back to full bowl. From the state salt sampling the agent can either choose the action cook which deterministi- cally transfer the agent to the baked salty cookies state; or the action return to bowl, which goes back to state salted bowl. Both baked sugary cookies and baked salty cookies are terminal states, returning respectively a reward of $+10$ and -10.

1. Define the MDP underlying the above environment.

State empty_bowl

$$\begin{bmatrix} & empty\_bowl & full\_bowl \\ wait & 1 & 0 \\ prepare\_dough & 0 & 1 \end{bmatrix}$$

State full_bowl

	$full\_bowl$	$sugared\_bowl$	$salted\_bowl$
$wait$	1	0	0
add_sugar	0	1	0
add_salt	0	0	1

State sugared_bowl

	$sugar\_sampling$	$baked\_sugared\_cookies$	$full\_bowl$
$taste$	1	0	0
$cook$	0	1	0
$restart$	0	0	1

State salted_bowl

	$salt\_sampling$	$bake\_salted\_cookies$	$full\_bowl$
$taste$	1	0	0
$cook$	0	1	0
$restart$	0	0	1

state sugar_sampling

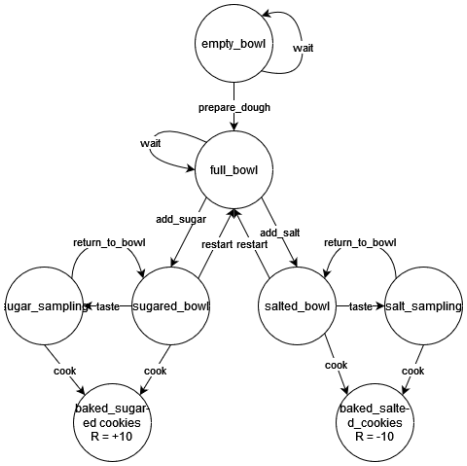
	$baked\_sugared\_cookies$	$sugared\_bowl$
$cook$	1	0
$return\_to\_bowl$	0	1

state salt_sampling

	$baked\_salted\_cookies$	$salted\_bowl$
$cook$	1	0
$return\_to\_bowl$	0	1

R(baked_sugary_cookies) = +10
R(baked_salted_cookies) = −10

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?

Yes we can. Given that we always start in the state empty_bowl we can define an optimal deterministic policy as

$$\pi(prepare\_dough|empty\_bowl) = 1$$

$$\pi(add\_sugar|full\_bowl) = 1$$

$$\pi(cook|sugared\_bowl) = 1$$

We could also define it for any other starting state and it would always be deterministically moving toward the state baked_sugared_cookies.

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

Since the environment is deterministic there is no reason to employ a stochastic policy. Since we can find an optimal value function the optimal policy would be to deterministically select the highest valued action at each step.

Suppose now that the agent has lost the ability to visually discriminate between salt and sugar (labels have been removed or the visual sensor of the agent has degraded). The environment is then re-defined as follows. State empty bowl and actions therein are unmodified. In state full bowl the agent can now choose only between two actions: wait, as before; and add ingredient which add a random ingredient between sugar and salt and deterministically transfer the agent to the enriched bowl state. Notice that the addition of either sugar or salt is tracked by a latent environment variable σ unknown to the agent. In state enriched bowl the agent can choose between the following actions: taste which makes the robot get a sample of dough and analyze its content – the resulting state will be either sugar sampling or salt sampling deterministically according to σ; cook which transfers the agent to the baked sugary cookies or baked salty cookies state, again deterministically according to σ; and restart which deterministically makes the robot discard the current dough and produce a new one, thus leading back to full bowl. The remaining states sugar sampling, salted bowl, baked sugary cookies and baked salty cookies follow the same behaviour as before.

5. Define the MDP underlying the above environment.

State empty_bowl

	$empty\_bowl$	$full\_bowl$
$wait$	1	0
$prepare\_dough$	0	1

State full_bowl

	$full\_bowl$	$enriched\_bowl(\sigma)$
$wait$	1	0
$add\_ingredient$	0	1

State enriched_bowl(σ)

	sugar_sampling	*salt_sampling*	*baked_sugared_cookies*	*baked_salted_cookies*	*full_bowl*
taste	σ	σ	0	0	0
cook	0	0	σ	σ	0
restart	0	0	0	0	1

state sugar_sampling

	baked_sugared_cookies	*sugared_bowl*
cook	1	0
return_to_bowl	0	1

state salt_sampling

	baked_salted_cookies	*salted_bowl*
cook	1	0
return_to_bowl	0	1

State sugared_bowl

	sugar_sampling	*baked_sugared_cookies*	*full_bowl*
taste	1	0	0
cook	0	1	0
restart	0	0	1

State salted_bowl

	salt_sampling	*bake_salted_cookies*	*full_bowl*
taste	1	0	0
cook	0	1	0
restart	0	0	1

R(baked_sugary_cookies) = +10
R(baked_salted_cookies) = −10

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?

Yes we can define an optimal deterministic policy as follows.

$$\pi(prepare\_dough|empty\_bowl) = 1$$

$$\pi(add\_ingredient|full\_bowl) = 1$$

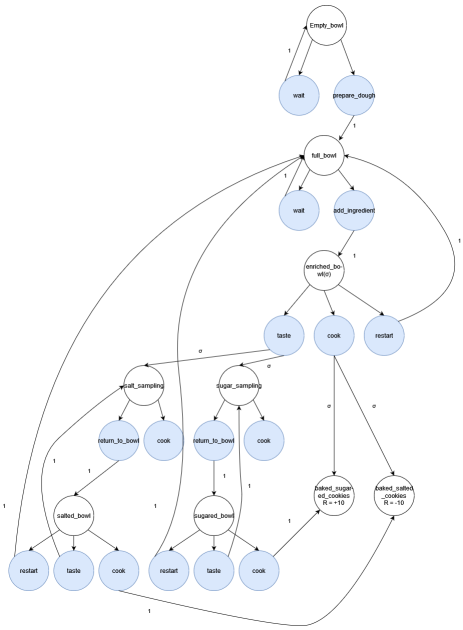
$$\pi(taste|enriched\_bowl(\sigma)) = 1$$

$$\pi(return\_to\_bowl|salt\_sampling) = 1$$

$$\pi(restart|salted\_bowl) = 1$$

$$\pi(cook|sugar_sampling) = 1$$

8. (*) Can you define an optimal stochastic policy for this environment? If so,



write it out. If not, why not?

Given the nature of the environment even though there is now an unknown factor we can still deterministically reach an optimal terminal state given enough time as long as the probability of getting a salted bowl isn't equal to 1. Even then the optimal policy is still deterministic as we get no penalty for simply restarting indefinitely.

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

This setup differs inn that we now have an unknown factor which decides whether we get a salted or sugared bowl. In this case that means that we need to check what ingredient we added and retry until we get sugar.

2

In class we discussed how to define a function approximator for the policy $\pi(a|s)$ using neural networks. Assume we have a discrete and finite number of actions A = a1, a2, ..., an. For instance, in chess, there is only a limited number of actions available.

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

To define a neural network that outputs a distribution over actions for a given state in reinforcement learning, we would typically design the architecture such

that the output layer produces a vector of scores, one for each possible action. These scores are then transformed into a probability distribution using a softmax activation function.

Her e eg veldig usikker ka de e ute etter, skal det være slide 12 og 13 fra week 9 - Reinforcement Learning: Policy Gradient

Assume now we have a continuous number of actions A = R. For instance, an agent can move any place along a line.

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

To model the continuous number of actions A=R we could use a Gaussian distribution. This distribution offers a continuous probability density function.

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

When we are working with a Gaussian distribution to model the policy in reinforcement learning, we need to consider the parameters such as the mean and standard deviation, since these determine the characteristics of the distribution.

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

To achieve this we simply need to define the output layer of the NN to output the parameters of a probabilistic distribution. In the case of a Gaussian distribution the NN would have to output a mean and a standard deviation.

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

Answer

3

Recall the gradient bandit algorithm for MABs. We defined the update rule as:

$$H^{(t+1)}(a_i) = H^{(t)}(a_i) + \alpha(R^{(t)} - \hat{E}[R])(1 - p^{(t)}(a_i))$$

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

The $\hat{E}[R]$ term is analogous to the baseline in policy gradient algorithms.

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

Subtracting $\hat{E}[R]$ from the reward makes it so we are considering the difference between the current reward and the expected reward from all previous time-steps rather than only the current reward. This reduces the variance of our steps which often leads to faster convergence. Also if we increase the mean of the rewards while keeping the variance the same, the performance stays the same when including the baseline term while it may deteriorate when omitted.