CMPUT 652: Reinforcement Learning with Robots **Fall 2019**

Lecture 5: September 17

Instructor: Rupam Mahmood *Scribe: Matthew McLeod*

5.1 Discussion Last Lecture

In the previous lecture, we covered the difference between exact and sampled based MSE as well as the optimization for both of these cases.

Course Administration

The following is a summary of the course administration component of the beginning of class.

For the final project, you will be working with a robot. There are 3 available types: the UR5 arm, iRobot Create 2s, and a collection of Dynamixel servomotors.

- There is one available UR5 arm from Kindred Inc.
- A show of hands was done for people interested in working with iRobot Create 2s.
- A collection of smaller robotic components that could be assembled to do a task.

It is possible to try working with a robot not listed here, but extensive research is required to ensure that it will be appropriate and feasible for the course. Talk to professor Mahmood should you have questions about this. In order to determine who works with which robot, priority will be given to those who have the highest assignment grades up until that point.

5.2 Bandits

The k-armed bandits problem refers to having multiple actions associated with a reward distribution. There are additional complications that can be added to the bandits problem such as contextual bandits. Contextual bandits is when there are different reward distribution per action conditioned on a stimulus being shown to the agent. However, in the remaining of the lecture, we will not consider contextual bandits and remain in the k-armed bandit setting.

5.3 Policies

Policies are a key component in understanding how an agent should act in a bandits scenario. A policy is a function that takes in an action as input and outputs the probability of taking the action. A policy can either be deterministic or stochastic.

A policy is defined as:

$$\pi(a) = p(a) = P_\pi\{A = a\} \tag{5.1}$$

where, a is the action and $\pi(a)$ is the probability of the action.

The value of taking action a is the action-value function q and is shown below:

$$q(a) = \mathbb{E}[R|A = a] \tag{5.2}$$

where R is the reward. The optimal action to maximize reward is then:

$$a^* = \underset{a}{\operatorname{argmax}}\, q(a) \tag{5.3}$$

If you have a policy π that follows the optimal actions, it is called an optimal policy and denoted with π^*.

In order to find the action-value function, stochastic gradient descent (SGD) can be used to reduce the MSE. Recall the form of SGD from the previous lecture. The update rule for SGD applied to $q(a)$ is shown below.

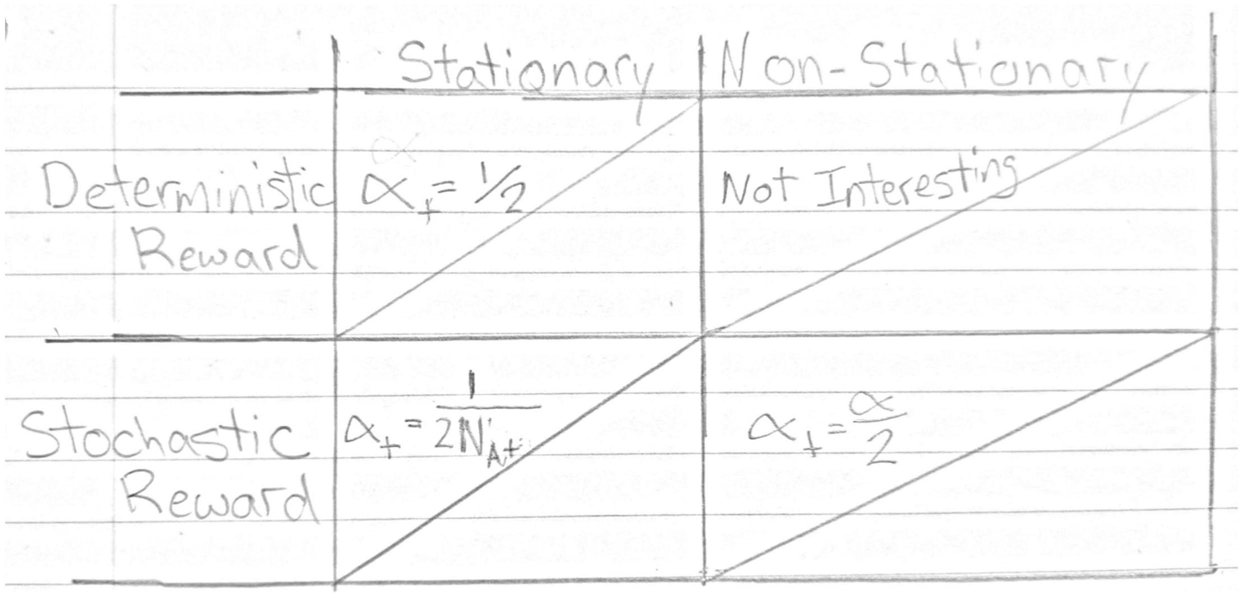
$$\hat{q}_{t+1}(A_t) = \hat{q}_t(A_t) + 2\alpha_t[R_t - \hat{q}_t(A_t)]$$

where, $\hat{q}$ is the incremental estimate of q, α_t is step size applied to the error signal.

Remember that α has a large influence on how the function estimate is updated. A constant α exponentially weights more heavily recent error signals than older error signals. If α decays at a rate that is the inverse to the number of times the action has been, then all the error signals have been equally weighted.

5.4 Analysis of Bandits

Since bandits can occur in different environments and reward systems, it is useful to consider each combination individually. Table 5.1 shows the optimal α values for the different settings.



	Stationary	Non-Stationary
Deterministic Reward	$\alpha_t = 1/2$	Not Interesting
Stochastic Reward	$\alpha_t = \frac{1}{2N_{A_t}}$	$\alpha_t = \frac{\alpha}{2}$

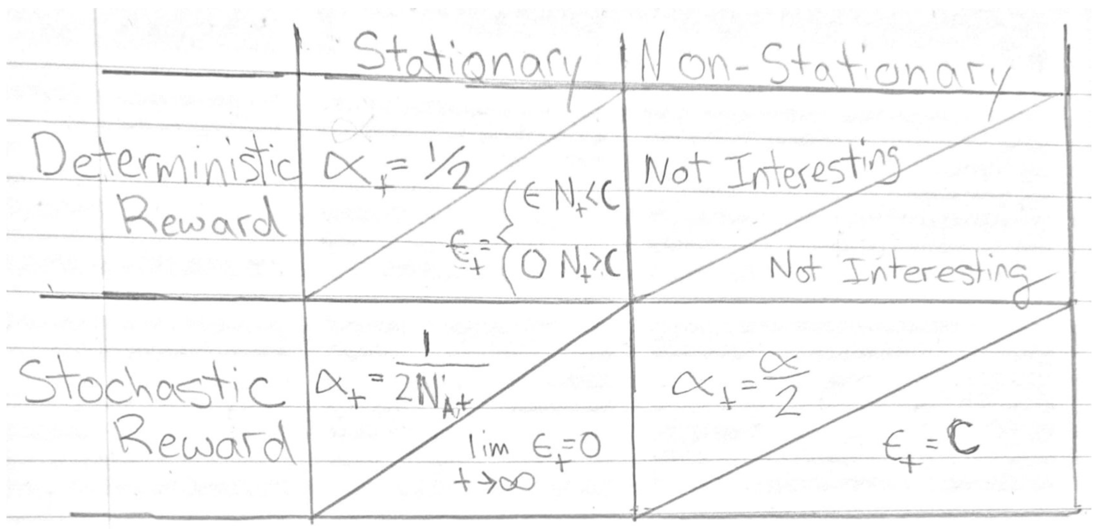
Figure 5.1: The optimal α_t for different environments and reward structures.

Stationary means that the reward distribution never changes. Non-stationary means that the reward distribution may shift as the agent interacts with the environment. This may result in non-optimal actions becoming optimal. A deterministic reward is when the agent will always receive the same reward for a given action while stochastic reward means that the reward is being sampled from a distribution (excluding distributions like Dirac).

For the case of a stationary environment with a deterministic reward , it is best to have $\alpha_t = \frac{1}{2}$ so that the estimate of q only relies on the previous reward. This allows the estimate to immediately jump to the correct reward estimate from only a single trial. If the reward signal changes to a stochastic reward, then it is best to let α_t be equal to $\frac{1}{2N_{A_t}}$ where N_{A_t} is the number of times action A_t has been chosen. This allows the agent to equally weight all sampled rewards and converge towards the true expected reward per action.

For the non-stationary case, a deterministic reward is not interesting since the reward is changing and essentially is a non-stationary stochastic reward. For the case of a non-stationary stochastic reward, α_t can be set to a constant to track in response to each sample at the same rate. By having α_t set to a constant, it allows the agent to exponentially weight the rewards and thus 'forget' the older ones. This enables more effective tracking of the moving distribution of rewards.

There is also a problem of exploration versus exploitation in the bandits problem. It does not matter what α value the agent has for an action if the action is never chosen. It is also important that the agent samples enough times each action to get an accurate estimate of the reward for the action. One solution is the ϵ-*greedy* method where $\epsilon \in [0, 1]$. The agent acts optimally (also called greedily) with a probability of $1 - \epsilon$, and the agent acts randomly with ϵ probability. Table 5.2 shows reasonable ϵ values for each scenario as well.



	Stationary	Non-Stationary
Deterministic Reward	$\alpha_t = 1/2$ $\epsilon_t = \begin{cases} \epsilon & N_t < C \\ 0 & N_t > C \end{cases}$	Not Interesting Not Interesting
Stochastic Reward	$\alpha_t = \frac{1}{2N_{A_t}}$ $\lim_{t \to \infty} \epsilon_t = 0$	$\alpha_t = \frac{\alpha}{2}$ $\epsilon_t = C$

Figure 5.2: The optimal α_t and ϵ_t for different environments and reward structures.

For the stationary environment with a deterministic reward, ϵ_t should explore every action. Once every action has been tried atleast once, then ϵ should be 0 as this allows full exploitation. Since the environment is stationary, the optimal action will always be the same. When the reward signal is stochastic, then a decreasing ϵ is ideal and converges towards 0 as the action-value function converges towards the true expected value of the optimal action.

For similar reasons to table 5.1, the non-stationary deterministic reward is not interesting. For a stochastic reward, ϵ should be a constant value. This allows continual exploration that enables the discovery of shifting optimal actions as the reward distributions change.

5.5 Policy Gradient Methods

The action-value function is useful for cases of a finite and discrete number of actions such as the bandit setting analyzed thus far. However, when the action space is continuous relying on value functions becomes difficult. To act optimally, one must take the max out of the set of actions. This becomes difficult to do in continuous domains. Therefore, policy gradient methods are an alternative which alleviate this problem.

If the objective function for the discrete case is:

$$\mathcal{V}(\theta) = \mathbb{E}[q_\theta(A_t)] = \sum_{a \in \mathcal{A}} \pi_\theta(a) q(a), \tag{5.4}$$

then for the continuous case:

$$\mathcal{V}(\theta) = \int_{\mathcal{A}} \pi_\theta(a) q(a) da \tag{5.5}$$

To maximize the objective function, take the derivative with respect to the parameterization of θ. Since we want to typically minimizing functions as this is the more common paradigm in optimization, we can multiply the function by -1.

$$\begin{aligned}
\nabla_\theta \mathcal{V}(\theta) &= -\int \nabla_\theta \pi_\theta(a) q(a) da \\
\nabla_\theta \mathcal{V}(\theta) &= -\int \pi_\theta(a) \frac{\nabla_\theta \pi_\theta(a)}{\pi_\theta(a)} q(a) da \\
\nabla_\theta \mathcal{V}(\theta) &= -\int \pi_\theta(a) \nabla_\theta \log \pi_\theta(a) q(a) da \\
\nabla_\theta \mathcal{V}(\theta) &= -\mathbb{E}[\nabla_\theta \log \pi_\theta(A_t) R_t]
\end{aligned} \tag{5.6}$$

Therefore, to apply this in stochastic gradient descent:

$$\begin{aligned}
\theta_{t+1} &= \theta_t - \alpha_t \nabla \ell_t \\
\theta_{t+1} &= \theta_t + \alpha_t \nabla_\theta \log \pi_\theta(A_t) R_t.
\end{aligned} \tag{5.7}$$

5.6 Fundamental Difference between Supervised Learning and Reinforcement Learning

At this point, it is reasonable to wonder what is the difference between supervised learning and reinforcement learning. The table below shows the comparison of the supervised learning dataset of MNIST and a contextual bandits scenario.

Example Environments	MNIST	contextual Bandits
Input	Image: $\mathbf{X}$	Context: $\mathbf{X}$
Processing Output	Prediction: $f_\theta(\mathbf{X})$	Action: $f_\theta(\mathbf{X})$
Corrective Signal	True Label: Y	Reward: R

The true difference relies on the fact that the agent is interacting and influencing the environment. Both methods fall into function approximation, but in reinforcement learning the mind of the agent is important as it effects future events.

5.7 Cross Entropy Relationship to MSE

Recall that maximizing the log likelihood results in finding the MSE solution for supervised regression under the assumption that the target values are distributed normally. For the sample based method, we want to minimize:

$$-\frac{1}{t}\sum_{i=1}^{t}\log(p(Y_i|\theta, X_i). \tag{5.8}$$

Cross entropy is defined as:

$$\begin{aligned} &= -\int p^*(y)\log[p_\theta(y)dy] \\ &= -\mathbb{E}[\log p_\theta(y)] \end{aligned} \tag{5.9}$$

The cross entropy shown in equation 5.9 is the expected version of the sample based log-likelihood objective shown in equation 5.8. And from a function approximation perspective, cross-entropy objective, that is, trying to match the underlying distribution with a parameterized function, also makes sense. Therefore, minimizing the cross entropy can be seen as a more fundamental objective, from which MSE can be derived.