Q1: Contextual Bandits

(a) From the PGM we can see that there is no directed edge between the action and the next state. Hence, the current action does not effect future state and hence the future reward. Hence, the environment is a Bandit setup. Dependent on whether the context space has one or more context, the environment is either Multi-Armed Bandit or Contextual Bandit respectively.

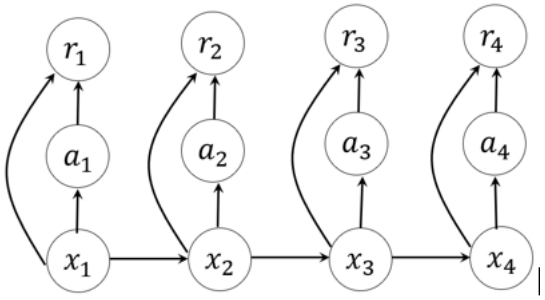

Figure 1: PGM

(b) The psuedocode of ETC algorithm is given below. Explanation:

1. In line 9, if $P(x, e_x) \geq M$, then it basically means that the agent has completed exploring action e_x for context x. So, we go to the next action in line 10.
2. Lines 15 and 16 means that if the last action, K, has been explored at least M times for all the contexts then the "exploration phase" is over and commit phase should begin.

Algorithm 1: Psuedocode for explore then commit

1 Set $Q(x, a) = 0$ and $P(x, a) = 0$ for all $x \in \{1, 2, \ldots, N\}$ and $a \in \{1, 2, \ldots, K\}$. $Q(x, a)$ is the estimated average reward for context-action pair (x, a). $P(x, a)$ is the number if times action a was taken for context x.
2 Set $e_x = 1$ for all $x \in \{1, 2, \ldots, N\}$. e_x stores the action to be taken for context x if the agent is in "explore phase".
3 Set $commit = False$.
4 **for** *every time slot* **do**
5 Observe context x.
6 **if** *commit* **then**
7 Choose action with the highest estimated Q-value, i.e. set $a \leftarrow \underset{k \in \{1,2,\ldots,K\}}{\arg\max}\ Q(x, k)$.
8 **else**
9 **if** $P(x, e_x) \geq M$ **then**
10 Set $e_x \leftarrow \min(e_x + 1, K)$.
11 Set $a \leftarrow e_x$.
12 Take action a and get the reward r.
13 **if** *not* $(commit)$ **then**
14 Update $Q(x, a)$ and $P(x, a)$ as follows,

$$\begin{aligned} Q(x, a) &\leftarrow Q(x, a) + \frac{1}{P(x, a) + 1}(r - Q(x, a)) \\ P(x, a) &\leftarrow P(x, a) + 1. \end{aligned}$$

15 **if** $P(\overline{x}, K) \geq M$ *for all* $\overline{x} \in \{1, 2, \ldots, N\}$ **then**
16 Set $commit \leftarrow True$.

(c) That is not so For ETC algorithm there are definitely some context-action pair that is sampled only M times. This is the glaring flaw of ETC because one of the fundamental rule of RL (and by extension contextual bandits) is that all state-action pair (context-action pair here) should be sampled infinitely often.

(d) For Algorithm 1, the commit phase for all contexts are starting together. This is sub-optimal because if there are some contexts that rare, then in order to finish exploration for those rare contexts, we may keep on picking sub-optimal actions for other contexts whose exploration has finished. In order to avoid this we can start the commit phase of a context immediately after we have finish exploring all the actions for that context. The following will be the changes in the psuedocode:

1. In line 3 we will have: Set $commit_x = False$ for all $x \in \{1, 2, \ldots, N\}$.
2. In lines 6, 13, and 16 we will have: $commit_x$ instead of $commit$.
3. In line 15 we will have: $P(x, K) \geq M$ instead of $P(\overline{x}, K) \geq M$ *for all* $\overline{x} \in \{1, 2, \ldots, N\}$.

Q2: Markov Decision Process

(a) The state at time t is (d_t, b_t, h_t). Qualitatively, the demand, battery level, and the battery health are the states.

Let the set of demands be $\{0, 1, \ldots .D\}$ here D is the maximum demand. The set of battery level is $\{0, 1, \ldots, N_b\}$ and the set of battery health is $\{0, 1, \ldots, H\}$. Hence, the state space is

$$\{0, 1, \ldots .D\} \times \{0, 1, \ldots, N_b\} \times \{0, 1, \ldots, H\}$$

where H is the cartesian product.

(b) The action is simply u_t, the amount of energy to draw from the transmission grid.

The action space is,

$$A(d, b, h) = \left\{ \max(0, d-b), \max(0, d-b)+1, \ldots, d + ceil\left(\frac{N_b - b}{\alpha_h}\right) \right\}$$

Explanation:

1. The lower bound of u is $\max(0, d-b)$ which can be explained as follows. If the battery level is b, and the demand $d \geq b$, then we have to draw a minimum of $d-b$ units from the grid. If $d < b$, we don't need to draw energy from grid at all.
2. The upper bound of u is $d + ceil\left(\frac{N_b - b}{\alpha_h}\right)$ which can be explained as follows. The upper bound of u is the current demand, d, plus the maximum amount of energy from the grid required to completely charge the battery. If the battery level is b, then $N_b - b$ is the additional energy that can be stored in the battery. To charge the battery up by $N_b - b$ units, we need to draw a maximum of $ceil\left(\frac{N_b - b}{\alpha_h}\right)$ from the grid.

(c) Let $c(d, b, h, u)$ be the cost of the state-action pair (d, b, h, u). Then,

$$c(d, b, h, u) = u^2$$

(d) The equation is,

$$b_{t+1} = \begin{cases} b_t - (d_t - u_t) & ; \ u_t \leq d_t \\ \min\left(N_b, b_t + ceil\left(\alpha_{h_t} \cdot (u_t - d_t)\right)\right) & ; \ u_t > d_t \end{cases} \tag{1}$$

Explanation:

1. If $u_t \leq d_t$, then the remaining energy $d_t - u_t$ is drawn from the battery to satify the customer demand.
2. If $u_t > d_t$, then the excess energy that is drawn from the grid used to charge the battery $u_t - d_t$. Out of $u_t - d_t$ units, battery only gets charged by $ceil\left(\alpha_{h_t} \cdot (u_t - d_t)\right)$ units depending on its health. Off course, battery can't be charged beyond it capacity N_b and hence the $\min(\cdot)$ operator.

(e) Let $f(d_t, b_t, h_t, u_t)$ denote the RHS of (1). Then, the Bellman optimality equation is,

$$V(d,b,h) = \min_{u \in A(d,b,h)} Q(d,b,h,u) \qquad \text{where,}$$

$$Q(d,b,h,u) = \begin{cases} u^2 + \beta \sum\limits_{d' \in \{0,1,\ldots,D\}} \theta_{d'} \cdot V\left(d', b', h\right) & ;\ b \leq 0.8N_b \\ u^2 + \beta \sum\limits_{d' \in \{0,1,\ldots,D\}} \theta_{d'} \cdot \left(\phi V\left(d', b', h'\right) + (1-\phi) V\left(d', b', h\right)\right) & ;\ b > 0.8N_b \end{cases} \tag{2}$$

and $b' = f(d,b,h,u)$, and $h' = \max(0, h-1)$.

Q3: Reinforcement Learning

You can approach Dr. Gone for the solution.

Q4: Deep Reinforcement Learning

(a) For policy gradient RL, the output of the neural network should be "probability like". To do this, we often use softmax activation function in the last layer of the neural network. Let p_a denote the output of the softmax function corresponding to action a. In policy gradient RL, we choose action a with probability p_a. Since we are using softmax activation function p_a will always be greater than zero for all a's. Hence, all actions has non-zero probability of getting chosen for all input states. This ensure that every state-action pair is sampled infinitely often.

(b) The action that has a higher Q-value is more likely to be optimal. So, we can assign probabilities to each action such that: (i) no action has zero probability. (ii) actions with higher Q-value has higher probability. One such approach is to set probability of action a, p_a, as follows,

$$p_a = \frac{e^{Q_a}}{\sum\limits_{k \in A} e^{Q_k}}$$

where Q_a is the Q-value of action a for the input state x. The psuedo is given below.

Algorithm 2: Psuedocode for weighted exploration

Input: DQN $\hat{Q}(\cdot\,; \phi)$, state x, and exploration probability ε.

1 Generate a random number between α between 0 and 1 uniformly at random.

2 Get the Q-value, Q_a, for state x and for all actions a in the action space, A, using the DQN $\hat{Q}(\cdot\,; \phi)$.

3 **if** $\alpha \leq \varepsilon$ **then**

4 Calculate a "probability like" value, p_a, for all $a \in A$ using the following formula,

$$p_a = \frac{e^{Q_a}}{\sum\limits_{k \in A} e^{Q_k}}$$

5 Select action a with probability p_a. Return a.

6 **else**

7 Selection the action with the larget Q-value, i.e. $a^\star = \arg\max\limits_{a \in A} Q_a$. Return a.

(c) We have,

$$\widehat{Q}(x, a; w) = V(x; w) - (a - \mu(x; w))^T P(x; w) (a - \mu(x; w)) \tag{3}$$

The RHS of (3) should be scalar because Q-function of a state-action pair is a scalar value. Also action $a \in \mathbb{R}^N$. So, for (3) to be dimensionally correct we need dimensions of $V(x; w)$, $\mu(x; w)$, and $P(x; w)$ to be 1, $N \times 1$, and $N \times N$ respectively. Since, $V(x; w)$, $\mu(x; w)$, and $P(x; w)$ are the outputs of the neural network, the number of outputs in its output layer is $1 + N + N^2$.

(d) It is a regression problem because the estimated Q-value, $\widehat{Q}(x, a; w)$, assumes continuous values.

(e) In order to compute actions, Deep Q-Learning has to solve the following optimization problem

$$\pi(x) = \arg\max_{a \in A} \widehat{Q}(x, a; w)$$

where $\widehat{Q}(\cdot; w)$ is the DQN, x is the state, a is the action, and A is the action space. The above optimization problem is a non-linear constrained optimization problem to solve becasue neural networks are very complex. Hence, Deep Q-Learning cannot be directly extended to continuous action space.

(f) To find action a that maximizes $\widehat{Q}(x, a; w)$, we find the partial derivative of $\widehat{Q}(x, a; w)$ in (3) with respect to a and set it equal to $\mathbf{0}$. Using chain rule of differentiation we get,

$$\begin{aligned} \frac{\partial \widehat{Q}(x, a; w)}{\partial a} &= -P(x; w)(a - \mu(x; w)) - P(x; w)^T (a - \mu(x; w)) \\ &= -\left(P(x; w) + P(x; w)^T\right)(a - \mu(x; w)) \end{aligned} \tag{4}$$

Now we set the RHS of (4) to $\mathbf{0}$ (this is a vector zero),

$$-\left(P(x; w) + P(x; w)^T\right)(a^\star - \mu(x; w)) = \mathbf{0} \tag{5}$$

$$\Rightarrow a^\star - \mu(x; w) = \mathbf{0} \tag{6}$$

$$\Rightarrow a^\star = \mu(x; w) \tag{7}$$

Equation (7) is the closed-form expression of the desired optimal action. As far as exam is concerned, you will get full marks if you do it till here. But, to be more rigourous, the following two steps are required:

1. To find the maxima, it is not enough to just set the derivative to zero. We also have to should that the double-derivative is negative. For multi-variable calculus, this means that the double-derivative (the hessian) should be negative definite. The double-derivate of (4) with respect to a is $-\left(P(x; w) + P(x; w)^T\right)$. Since $P(x; w)$ is positive definite, $-\left(P(x; w) + P(x; w)^T\right)$ is negative definite (this is straight forward linear algebra).

2. Jumping from (5) to (6) is possible only if $-\left(P(x; w) + P(x; w)^T\right)$ is full rank. Since $-\left(P(x; w) + P(x; w)^T\right)$ is negative definite, it is also full rank (this is also linear algebra that you should have covered but not so straight forward).

(g) We can use tbe following two approaches (you have to write only one approach):

1. Approach 1: Let $a^\star$ be the estimated optimal action. We can select an action $a^\star + \delta$ where δ is random noise that included to indice exploration. δ should have zero mean and its standard deviation should decrease as training progresses (the latter point is similar to decreasing exploration probability ε with time).

2. Approach 2: Let $a^\star$ be the estimated optimal action. We probability $1 - \varepsilon$ we choose $a^\star$ (exploitation). With probability ε we choose an action uniformly at random from the action space.

Q5: State Estimation

(a) This is a direct question about deriving the recursive belief estimation equation of the Bayes filter. You can find it in lecture 36 notes.

(b) I gave wrong hint for this question which might have mislead few students. We actually need to use Bayes filter equation for this question which is quite tedious. So, I will do the following:

1. I will transfer the 7 marks for this question to part a.

2. If someone had correctly solved this question, I will use it to compensate for the marks lost in the other questions.