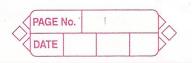
ARTIFICIAL INTELLIGENCE (AG)



REINFORCEMENT LEARNING

- Reinforcement Learning: Learning with feedback to agent about what is good or bad to decide which action to take. ex. chess,

 Ping-pong game.
 - + Task is to use observed rewards to learn an optimal (or nearly optimal) policy for an environment.
 - + TWO types:
 - 1. Passive Learning: Agency policy is fixed; the task is to learn utility of states (or state-action pairs); may involve learning a model of environment.
 - 2. Active Learning: Must also learn what to do; learn the policy also as there is no fixed policy. The principal issue is exploration; must explore as much as possible of its environment in order to learn how to behave in it.
- * Passive Reinforcement Learning:
- Fixed policy 'T', in state 's', always executes the action T(s).
- Goal to learn how good the policy is, i.e. how good the utility function UT(s) is.
- Similar to policy evaluation task of policy iteration algorithm.
 - + But, doesn't know transition model T (s, a, s'), which is probability of reaching state s' from 's' after doing action 'a'.
 - + And doesn't know the reward function R(s') which specifies reward for each state.
- Objective is to use the information about rewards to learn the expected utility UT(s) associated with each non-terminal state.
- Equation: UT(s) = E \(\frac{5}{t=0} \) | T, So=5 \\
 \text{discount Factor.} \]
- Ways to implement Passive Reinforcement Learning:
- · Direct utility estimation: States that utility of a state is the expected total reward from that state onward.
 - At end of each sequence, the algorithm calculates the



observed reward-to-go for each state and updates the estimated utility for the state accordingly, just by keeping a running average for each state in a table. - Ignores the dependence between states. 2. Adaptive Dynamic Programming (ADP): Agent tries to learn the transition model of the environment as it goes around and solving the Markov Decision Process using a dynamic programming method. - This adds T(s, T(s), s') and observed rewards R(s) into Bellman's Equation which is: $U^{\Pi}(s) = R(s) + Y \sum T(s, \Pi(s), s') U^{\Pi}(s')$. - Also adopt modified policy iteration approach of updating utility estimation after each change to the learned model. 3. Temporal Difference Learning: - Uses difference in utilities between successive states, often called temporal difference learning. - given with equation: U"(s) < U"(s) + x(R(s)+ Y U"(s')-U"(s)) - gives best of both the previous learning techniques. - Does not need model to perform its update. - First defines the condition that hold locally when the utility estimates that correct and then to write an update equation that moves the update estimates towards this ideal equilibrium equation. * Active Reinforcement Learning (ARL) - Agent must decide what action to take - First learn a complete model with outcome probabilities for all actions, rather than just the model for the fixed policy. - Learn utilities defined by the optimal policy; i.e. obeying Bellman's equation. U(s) = R(s)+ 1 max Es P(s a, s.) U(s'). - AFter learning optimal utility function, the agent can extract an optimal action by one-step look-ahead to max. expected utility.

- Agent must trade off between exploitation to max reward and

exploitation to max. long term well being.

*	EXI	plor	ati	on

- Any scheme used for ARL has to be Greedy in the limit of infinite exploration (GLIE).
- A GLIE scheme must try each action in each state, an unbounded number of times to avoid having a finite probability that an aptimal action is missed because of an usually bad series of outcomes.
- GLIE can be implemented as:
 - + By agent choosing random action a fraction 1/t of the time to follow the greedy policy otherwise slow.
 - + By assigning weights to actions not tried often, while tending to avoid actions of low utility. Done by assigning higher utility estimate to unexplored state-action pairs.

Equation: $u^{t}(s) \leftarrow R(s) + r \max_{\alpha} f(\xi P(s, \alpha, s') y^{t}(s'), N(\alpha, s))$ Here, f(u, n) is exploration function.

against curiousity (preference for lower 'n')

- * Learning an Action Value Function
- Learns an action-value representation instead of learning utilities. Method known as Q-learning.
- Motion Q (a,s) to denote value of doing action 'a' in state 's'.
 ... U(s) = max Q(a,s)
- Does not need a model for either learning or action selecting.
- Therefore, called model free method.

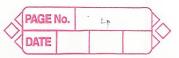
 $Q(a,s) = R(s) + Y \sum P(s|a,s') \max_{a'} Q(a',s')$

The updated equation is:

 $Q(a,s) \leftarrow Q(a,s) + x (R(s) + x max Q(a',s') - Q(a,s))$

* Generalization in RI

- One Way Function approximation: It means using any sort of representation for the function other than table.



-	+ Makes it practical to represent very large state spaces.
(A)	+ Takes less space and allows for inductive generalization over
	I/P states.
1	+ chances of failure or longer delay for convergence.
	+ Makes more sense to use online learning algorithm.
3.	+ Apply Widrow-Hoff rule or delta rule for decreasing error.
	$Qi \leftarrow Qi - \times \partial Ei(S) = Qi + \times (2ij(S) - UQ(S)) \partial UQ(S)$
	+ Function Approximation allows reinforcement learner to
	generalize from its experience.
	+ Applying linearity of function parameters To and a learning
-1	equation becomes
	Qi-Qi+x[R(s)+r NQ(s')-Û(s)] (dûQ(s)/dQi) for utilities and
neturi exposures	$Qi \leftarrow Qi + \times [R(s) + \delta \max \hat{Q}_{Q}(a', s') - \hat{Q}_{Q}(a, s)] \left(\partial \hat{Q}_{Q}(a, s) \right) $ for Q -values.
	a' dQi
	antonit of tenders ti (au) 4 syste
	tizi (un sertesta esta esta esta la terra di fregga produce concerna di
	Line Colonia Maria Colonia Col
	nainerelle harten and the constant and much it is an entre
	The state of the s
	e ta contación de la contidar con la fina souten a colonida est del colonida en el colonida en e
	· 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
	· · · · · · · · · · · · · · · · · · ·
	THE PLANTS WAS THEN SHAFT BY A STREET
g PPR sange and the gas gives	
7.	- Company of the state of the s
	Jan - and redictional and and and the foreign