HQ LEARNING: DISCOVERING MARKOVIAN SUBGOALS FOR NON-MARKOVIAN REINFORCEMENT LEARNING

Chris Bennett and Jaymin Kessler

April 22, 2003

written with $\LaTeX$because it is superior to anything M$ can come up with

1 REINFORCEMENT LEARNING

- Addresses the question of how an autonomous agent that senses and acts in its environment can learn to choose optimal actions to achieve its goals
- The ideas of rewards and penalties
- RL algorithms are related to dynamic programming algorithms frequently used to solve optimization problems

2 DIFFERENCES FROM OTHER FUNCTION APPROXIMATION TASKS

- Delayed Reward - temporal credit assignment - determining which of the actions in its sequence are to be credited with producing the eventual rewards
- Exploration - tradeoff between exploring unknown states and actions versus exploitation of states and actions the agent has already learned will yield a high reward
- Partially observable states - Can't see everything...for example, robot that can't see behind itself...thus it may need to consider its previous observations together with the current state
- Life-long Learning - a single agent learning to multiple things...this could introduce the possibility/need to remember prior knowledge

3 MARKOVIAN LEARNING

- In a Markov decision process (MDP) the agent can perceive a set S of distinct states of its environment and has a set A of actions that it can perform
- At each time step, the agent chooses an action at a given state, and receives a reward and new state from the environment
- S and A can be finite or not, and the reward and new state functions can be deterministic or non-determ.
- Goal is to find a policy that maximizes the rewards over time
  - many variations including discounting future rewards, non-discounted/finite horizon rewards, average reward

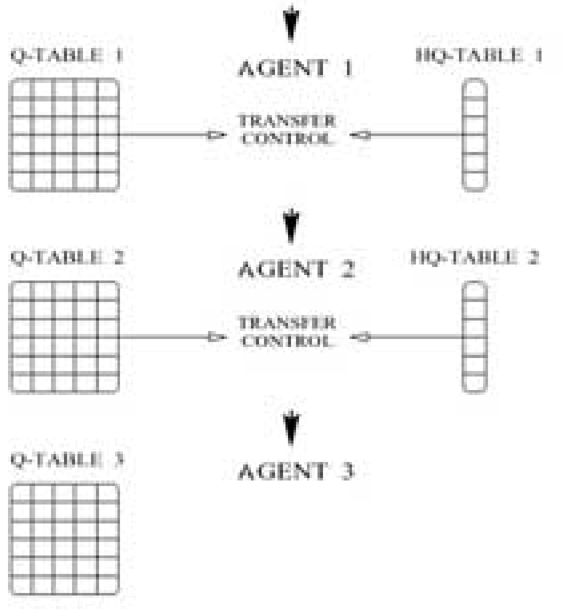
4 Q-LEARNING

- Learning a function to approximate the optimal policy for maximizing rewards
- This optimal policy requires perfect knowledge (often not even close to possible)
- Focuses on acquiring optimal control strategies from delayed rewards, even when the agent has no prior knowledge of the effects of its actions on the environment
- Q Learning depends on a Markovian interface
- That is, they fail if any memory of previous events is required

```
Simple Q Learning Algorithm
- For each s,a initialize the table entry Q(s,a) to zero.
- Observe the current state s
Do forever:
* Select an action a and execute it
* Receive immediate reward r
* Observe the new state s'
* Update the table entry for Q(s,a) as follows:
$Q(s,a) \leftarrow r + \gamma max(q(s',a'))$
* $s \leftarrow s'$
```

5 POMDPs

- Partially Observable Markov decision problem
- A number of algorithms exist, but they are only feasible for small problems - Schmidhuber (1991), McCallum (1993), Ring (1994), Kaelbling etal (1995), Jaakkola et al (1995)...
- Some larger scale algos exist - Schmidhuber et al (1996), Wiering and Schmidhuber (1996), and Zhao and Schmidhuber (1996)



6 AGENT ARCHITECTURE

- The system is composed of N agents, of which only one can be active at a time
- Active(t) = i is the agent that is currently active at time t
- With the exception of the last, all agents have a Q table, and HQ table and a transfer control unit.
- Each agents learn a policy mapping a state to an action; Agents learn to combine policies on their own
- All learning is done offline

7 Tasks are decomposed as follows:

- HQ table selects the probable best subgoal for an agent to accomplish
- Q table helps determine the probable best action to take next
- Q table size = $|O|x|A|$ (total possible observations by total possible actions)
- HQ table size = $|O|$ (total possible observations)

8 HQ POMDP ARCHITECTURE

POMDP is defined by the following vector: Z = $< S, S1, O, B, A, R, \gamma, D >$

- S is a finite set of environmental states.
- S_1 = initial state
- O is a finite set of observations
- B is a function mapping a state to an (ambiguous) observation. Subgoals are usually observations so B takes the current state and returns the current observations
- A is a finite set of actions
- R maps state/action pairs to reinforcement signals
- γ is a discount factor trading off immediate and delayed rewards
- D maps state/action pairs to a new state (state transition)
- S_{t+1} = D(S_t,A_t) where S_t (in S) is the current environmental state at time t and A_t (in A) is the action executed at time t

9 SELECTING A SUBGOAL

remember, a goal can be thought of as a target observation

$$P^{HQ_i}(O_j) := Pr_{max} \frac{Max_i(O_j)}{\sum_{O_k \in O} Max_i(O_k)} + \frac{1 - Pr_{max}}{|O|}$$

- Finds which observations maximize HQ
- $Max_i(O_j)$ is 1 if for every O_k , [$HQ_i(O_j) >= HQ_i(O_k)$]
- $Max_i(O_j)$ goes to zero for every observation that does not result in the maximum HQ value for HQ table i
- Pr_{max} can be used as a parameter to tradeoff between exploration of new subgoals and exploiting previously known fruitful goals.

Why not borrow that fun delta function from non-uniform mutation???

$$a = 1 - r^{(1 - \frac{t}{Tmax})^5}$$

10 SELECTING AN ACTION

Max-Boltzmann distribution:

$$P_t^{Q_i}(A_j) := Pr_{max} \frac{Max_i(O_t, A_j)}{\sum_{A_k \in A} Max_i(O_t, A_k)} + (1 - Pr_{max}) \frac{e^{Q_i(O_t, A_j)/T_i}}{\sum_{A_k \in A} e^{Q_i(O_t, A_k)/T_i}}$$

- $Max_i(O_t, A_j)$ is used to find the action A_j that maximizes the Q table for current observation O_t
- Note that the current agent's action choice depends only on the current observation
- Here the first term functions almost identically to the first term of the subgoal selector
- T_i is temperature and can be used to adjust randomness

11 TRANSFER CONTROL (to other agents)

IF
(FINAL GOAL NOT REACHED) AND
($B(S_t) = \hat{O}_i$) (for current subgoal $\hat{O}_i$) AND
(Active(t) $<$ M)

THEN
Active(t+1) = Active(t) + 1 AND
t_{i+1} = t +1

- this can be read as: if the final goal has not yet been reached but the current subgoal has, and the current agent is not the last agent, then make the next agent active and increase time unit by 1

12 LEARNING Q-VALUES

$Q_i(O_t, A_j)$ approximates the future discounted reward for executing Action A_j given observation O_t

optimal case :

$$Q_i(O_t, A_j) = \sum_{S_j \in S} P_t(S_j|O_t, \theta, i)(R(S_j, A_j) + \gamma V_{Active(t+1)}(B(D(S_j, A_j))))$$

where:

- $P_t(S_j|O_t, \theta, i)$ is the probability of the system being in state S_j at time t... given observation O_t, system parameters θ, and i = Active(t)
- HQ Learning does not depend on estimating this probability
- $R(S_j, A_j)$ maps state S_j and action A_j to a reinforcement value and multiplies it by the discount factor
- $D(S_j, A_j)$ takes the action currently being considered and gets the resulting state
- $B(D(S_j, A_j))$ takes the potential resulting state and gets an observation
- V is a a measure of how useful an observation is to a particular agent and is usually equal to the maximum Q value for observation Ot and all possible actions.

13 SPECIFIC Q LEARNING RULES

1. If C_i is active at time t, and C_j is active at time t+1, and $t < T$ then
   $Q_i(O_t, A_t) \leftarrow (1 - \alpha^Q)Q_i(O_t, A_t) + \alpha^Q(R(S_t, A_t) + \gamma V_j(O_{t+1})$
2. If agent C_i is active at time T, and the final action A_T has been executed, then
   $Q_i(O_T, A_T) \leftarrow (1 - \alpha^Q)Q_i(O_T, A_T) + \alpha^Q(R(S_T, A_T)$

- α^Q is the learning rate
- Retain part of the original Q value by multiplying it with $1 - \alpha^Q$
- Multiply the value of the current move by the maximum usefulness of the next move

14 SPECIFIC RULES FOR LEARNING HQ-VALUES

1. If agent C_i is invoked before agent C_{N-1}, then we update everything with
   $HQ_i(\hat{O}_i) \leftarrow (1 - \alpha^{HQ})HQ_i(\hat{O}_i) + \alpha^{HQ}(R_i + \gamma^{t_{i+1}-t_i} HV_{i+1})$

2. If $C_I = C_{N-1}$ then
   $HQ_i(\hat{O}_i) \leftarrow (1 - \alpha^{HQ})HQ_i(\hat{O}_i) + \alpha^{HQ}(R_i + \gamma^{t_N - t_i} R_N)$

3. If $C_i = C_N$ and $i < M$
   $HQ_i(\hat{O}_i) \leftarrow (1 - \alpha^{HQ})HQ_i(\hat{O}_i) + \alpha^{HQ} R_i$

- $\hat{O}_i$ is the current agent's chosen subgoal
- $R_i = \sum_{t-t_i}^{t^{i+1}-1} \gamma^{t-t_i} R(S_t, A_t)$ and represents C_i's discounted cumulative reinforcement during the time it will be active
- $HV_i := Max_{O_l \in O}(HQ_i(O_l))$ is the estimated discounted cumulative reinforcement to be received by C_i.

note complex dynamics resulting from the influence Q and HQ tables have on each other

15 A REALLY GREAT EXAMPLE

MAZE AND KEY PROBLEM