**Feudal Q Learning**

**Some Notes**

**Summary:**

1. Manager doesn’t know which sub manager active
2. Commands specified become part of submanager state space (so they learn how to satisfy)
3. Reinforcement to sub manager comes from world, to provide all levels with information about length of paths (counter) and also from manager(sub task reward)
4. Escape clause if command ineffectual
5. The collection of sub-managers can therefore face a temporal credit assignment problem: reward or punishment only comes from their manager when the state changes at the managerial level { even though the state can change at the sub-managerial level without this happening.
6. primitive…. Q values as normal
7. until level 1 state changes then subtask reward and potnentially c?
8. Managers do get to see the amount of reinforcement accrued from the world in the course of execution of their commands (they need this to help work out if the set task was appropriate).
9. if a super managerial task is achieved by a sub-managerial action, but that action violates a managerial command, then neither the manager nor the sub-manager should be rewarded.

i.e. if 4 levels

level 1 0,0 wants to go south 1,0

level 2 1,0 wants to go east 1,1

level 3 (4,2) goes south

🡪 (5,2) new state level 2 this should be penalized

🡪 (2,0) new state level 1(1,0) this should not be rewarded

1. Actions specified in terms of transitions they’d like at their own level
   1. Top manager only wants to search \* 🡪 (0-(0,0),(0,0),\*)[‘4’;0]
   2. Sub manager 🡪 (1-(1,1),(0,1),\*)[0,1,2,3,4] action selected =’N’
   3. Primitive\_manager 🡪 (2-(2,2),(0,2),’N’) [0,1,2,3] action selected =’S’
2. How to define States
   1. Manager state (level,state(agent),\*,goal[agent]) [action values]
   2. Sub-manager state (level,state(agent),sub\_task,goal(agent)) [action values]
   3. Primitive agent state (level,state(agent),sub\_task,goal(agent))[action values]
3. Agent knows: it’s location, location of goal and needs to get to goal in as few as steps as possible

Example Walkthrough

1. Manager🡪choose\_action(\*)🡪specify task to sub manager(\*)🡪
2. Sub\_manager🡪 choose action🡪 specify task to sub\_manager🡪
3. Primitve agent🡪 choose action 🡪 make move

Reward Map:

Move🡪

For level l-1….0:(0 top manager, l🡪 primitive agent)

Change in manager state[level[?

If manager state changes and fails, sub manager just active gets penalty by the manager and also pays the previous estimate of cost of satisfying managers command before state changed (c\_iz)

Yes🡪

If yes 🡪 Task achieved?

If yes r=0 + ciz

If no r=-10 +ciz

Control given back to manager [l]

No🡪 Add to counter

Is c\_iz[level]>timeout[level]:

Yes🡪 manager[level] gains control again

No🡪Primitive agent move again

1. Epsilon greedy with respect to boltzman , epsilon roughly 0.1 (didn’t implement)
2. Action replaced but agent doesn’t know this
3. Contrasted with normal q learning agent state (8\*8,8\*8,4) (state,goal,action)
4. Feudal systmen state space divide by factor of 2
   1. 8\*8🡪4\*4🡪2\*2🡪1\*1
   2. Action space =[\*],[‘NSEW\*],[NSEW]
   3. Some sub tasks can never be satisfied

For each trial, the agent knows the coordinates of the goal { the intent is that during learning it works out how to use this information to navigate to it. The location of the goal is also represented in a hierarchical manner, taking advantage of the way that space is split 🡪 Assumption here that there is a smaller state space due to this

If (and only if ) the goal lies within the domain at a given level, then the manager's action at that level will just be search

Escape clause 2\*number of lowest level states within managers purview….

i.e. at level 1 for 4 level maze,manager has 16 states under it 🡪 32

**Experiment Details**

* epoch=20
* 20000 trials over 50 samples
* 8\*8(4 level)
* 16\*16(5 level)
* 32\*32 ( 6 level)

This is not surprising { the hierarchical decomposition of the task suits the hierarchical decomposition of space exactly, and there is the potential for almost perfect sharing of information between the di  
erent tasks (ie the di  
erent goals). The advantage increases as the size of the example increases (in fact we were unable even to run conventional Q-learning for the 64 64 maze). Suggests that state space is smaller???

The Feudal system is slightly more expensive than the conventional system in the rst few epochs (5 for 8 8, 2 for 16 16, 1 for 32 32), presumably because exploration is happening at all levels.

The graphs also show some potential problems. The system has some diculty in learning at high levels not to select actions that are unsatisable (eg going South at 1-(2,1). This is because their unsatisfiability has to be learnt using time-outs.(prevented this in my implementation) No learning happens at level 1 if the level 1-(2,1) manager species action South, and the agent moves to 1-(1,1) or 1-(2,2). One could imagine penalising a manager directly for selecting an action that its subordinates never seem to be able to achieve, but that might hinder them from being made to try hard enough.