Final Project Report

Rohini Kapoor

August 21, 2019

1 Introduction

In this project, I try to simulate a mental modeler which detects the cause of back pain as an MDP environment, and run various learning algorithms to determine if this environment is indeed suited for RL.
In the community of back pain, there was a disagreement between doctors as to what the true cause of back pain is(physical/psychological). The actual cause of the pain should determine the treatment for every patient individually. Doctors now believe that the cause is a combination of various factors. We have a mental modeler from one of the doctors who has shared how the various factors weigh in to affect back pain.
As the treatment depends on each individual, we can view this problem as a Reinforcement Learning problem, where the agent should suggest the best treatment to each individual based on the various factors and the relation between them. The goal of this project is to represent the given mental modeler as an MDP, and to run different RL algorithms to determine if this problem can indeed be solved using RL. For simplicity, I have used Q-learning and SARSA as my RL algorithms to check if the problem converges.

2 Background

2.1 Fuzzy Computing Logic

Fuzzy-Logic Cognitive Mapping (FCM) is a parameterized form of concept mapping which allows to develop qualitative static models that are translated into semi-quantitative dynamic models. It is a way to structure expert knowledge using a soft system programming approach that is similar to the way humans make decisions. The system consists of components of the systems, relationship between components(positive or negative) and weights of those relationships. Such a model can be used to model a complex system with high uncertainty and little available data.

2.2 Mental Modeler

Mental Modeler is modeling software based on FCM that helps people capture their knowledge in a standardized format that can be used for scenario analysis. It helps users develop semi-quantitative models. It involves defining the components in the system, relationship between these components, and running

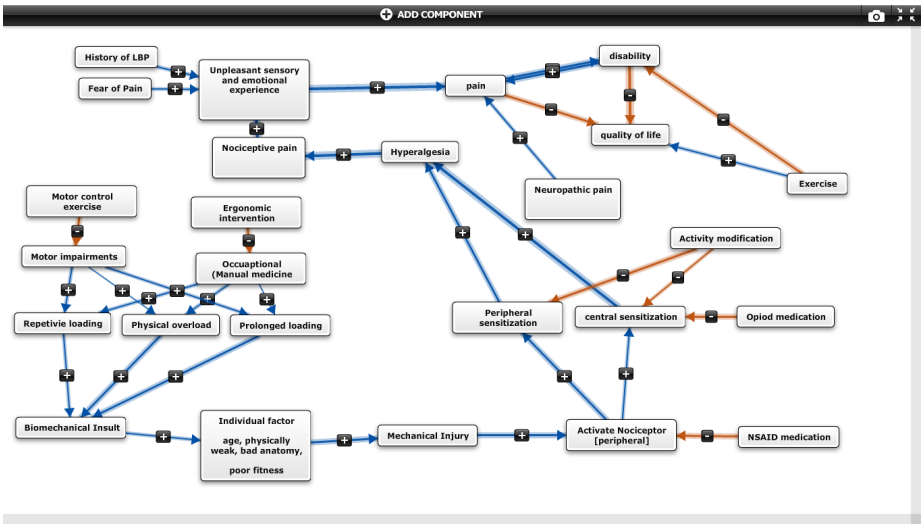


Figure 1: Mental Modeler for back pain

different conditions to determine how the system would react under a range of changes. The relationship between the components is given by an adjacency matrix that we use for our state transition.
The Mental Modeler that we have used for this project is shown in the figure 1.

3 Environment

We try to model the given mental modeler(MM) as an MDP. The forward simulation of the model is straightforward by matrix multiplication using the adjacency matrix. Each state can take values between -1 and 1. As shown by the [2], all inputs to a given component are multiplied by the weights of connections and then summed. This result is then input to an activation function, tanh, which caps the values of each node between -1 and 1.

The given model has 25 states which are-
pain, disability, quality of life, nociceptive pain, neuropathic pain, hyperalgesia, central sensitization, peripheral sensitization, activate nociceptor [peripheral], mechanical injury, unpleasant sensory and emotional experience, motor control exercise, biomechanical insult, repetivie loading, physical overload, prolonged loading, individual factor age, physically weak, bad anatomy, poor fitness, occuaptional (manual medicine, motor impairments, history of lbp, fear of pain, ergonomic intervention, nsaid medication, activity modification, opiod medication

Most of these states are hidden. From the expert's information, we have the following information for the initial state distribution:

- 50% people suffer from history of backpain
- 20% people have fear of backpain
- 10% people suffer from neuropathic pain

- 10% people suffer from central sensitization and peripheral sensitization

I have used these values as a mean to sample for these 5 states and calculate values for all other states.
Out of the 25 states, these five states are our control states- motor control exercise, ergonomic intervention, nsaid medication, activity modification, opiod medication. Various values and combinations of these five states are used for different experiments.
The adjacency matrix for the mental modeler is used for our state transition function, where the values propagate through each node based on the adjacency matrix.
The agent gets a reward of -1 for each time step until the terminal state is reached. It gets a reward of 0 for reaching the terminal state. The terminal state is also something that I have experimented with. Usually, it depends on the values of the states - Quality of life, pain and disability.
For this environment, I have chosen γ to be 1. Our aim is to give the patient the best advice over time, and we should not penalize actions taking at earlier stages.

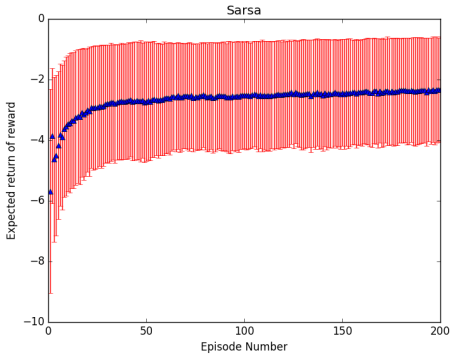
4 Experiments and Results

For each experiment, I have defined a state vector for size 25 for all the mentioned components. I have sampled the 5 initial states from a normal distribution with the mean as specified by the expert, and a standard deviation of 0.01. To update the value of states, I multiply the adjacency matrix with the state vector, and update the state vector. I keep the initial states fixed, and at each step apply the activation function *tanh*. To get the initial state vector, I multiply the adjacency vector for 6 steps.
For each of the 5 actions, I have defined the number of steps that the multiplication would take to propagate to the hidden nodes.
These parameters worked well and were used for all experiments:
$\alpha = 0.05$ and $\epsilon = 0.05$
After that I run Q-learning and SARSA on various conditions.

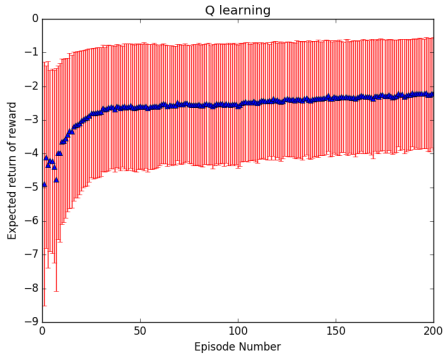
For the first experiment, I allow the agent to take only 1 action and give it a value of +1.
To reach the terminal state, the value of the component quality of life should be more than 0.7 Both SARSA and Q-learning converse to around -2 rewards.
I have run the algorithms for 5000 trials, 200 episodes per trial. Table 1 shows the plots for SARSA and Q-learning which show the average return per trial per episode, along with the standard deviation.

Because we have many nodes, the state space is huge. It does not make sense to do a Fourier transformation. So one change I did was to round off the state values to the 2 decimal points to check if it helped in learning by reducing the values the states can take. I have used the same terminal condition and same action selections. From here on, I have done this for all experiments.
Their results for 5000 trials and 200 episodes are as showed in Table 2

Next, I tried for different values for terminal state. Interestingly, if I keep the threshold for quality of life as 0.75 instead of 0.7, the method does not

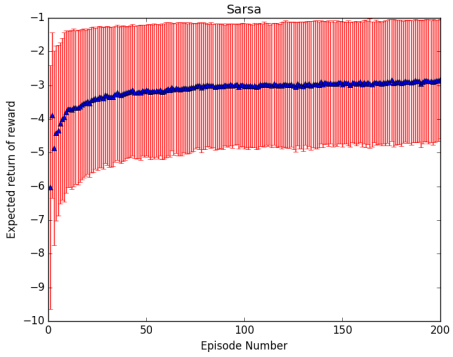


(a)

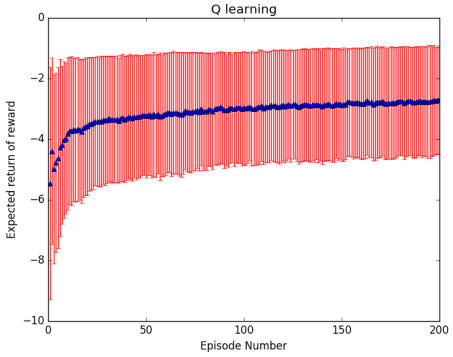


(b)

Table 1: Experiments with one action selected



(a)



(b)

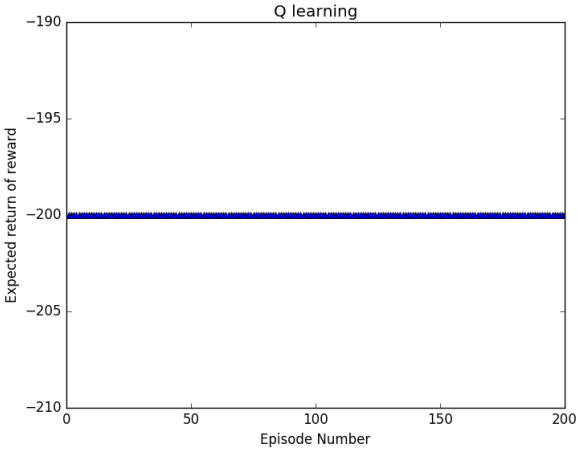
Table 2: Experiments with state values rounded off

converge. It was run for 100 trials, 200 episodes, with 200 as maximum time-step per episode, as shown in Table 3.

I tried to used the components pain, disability and quality of life to determine the terminal state as follows: quality of life should be greater than 0.7, pain should be less than 0.1 and disability should be less than 0.1. Their results for 5000 trials and 200 episodes are as showed in Table 4.

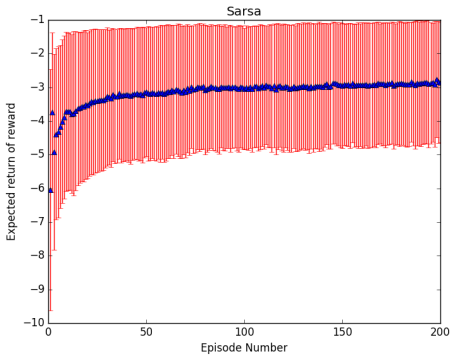
In the actual user-scenario, it does not make sense to suggest just one action to the user. For example, any user can be assigned both, medication and exercise. To account for this, I used a combination of actions. Hence there were a total of $2^5 = 32$ actions available. All the selected actions components are given a value of $+1$. The time step is selected as the maximum time-step of all the chosen actions. Their results for 5000 trials and 200 episodes are as showed in Table 5.

At times it might not be feasible to give the action values for 1. For example you cannot give a patient full dosage of a medication, it has to be given in parts. For this, I experimented with different values to set for the selected actions. In

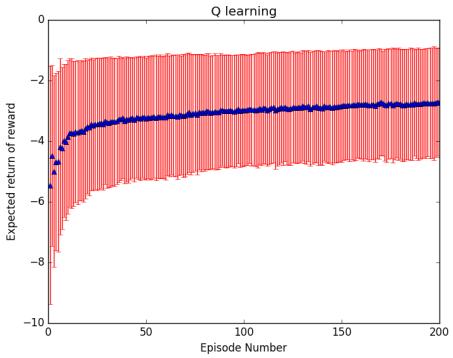


(a)

Table 3: Experiments with threshold for quality of life threshold more than 0.75



(a)



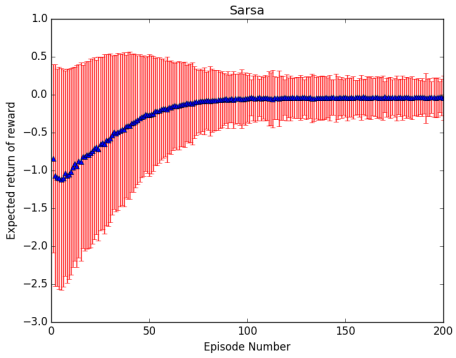
(b)

Table 4: Experiments with different terminal state

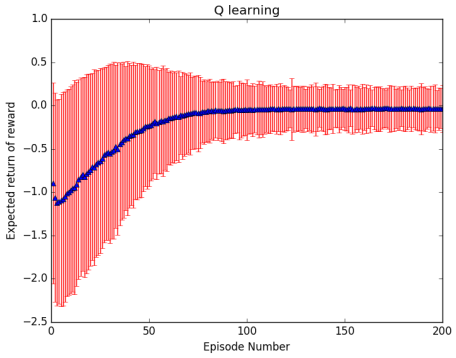
this experiment, I select just one action, but that action can take values either 0.25, 0.5, 0.75 or 1. Hence, we have a total number of (5*4=20) actions. This might take more time to learn, but is a more realistic approach to the problem in hand. Their results for 5000 trials and 200 episodes are as showed in Table 6.

5 Conclusion and Future Work

The fact that the algorithms have converged give us some hope that we might be able to simulate this model. One obvious extension is to test with all different values and combinations of the control nodes. We will have to deal with continuous action states for that. I plan to try Monte Carlo as I feel we can evaluate such an agent after an episode ends. I also try to use other learning

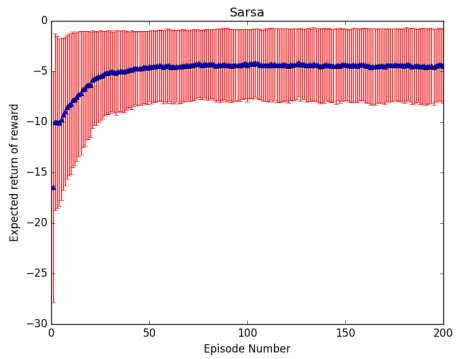


(a)

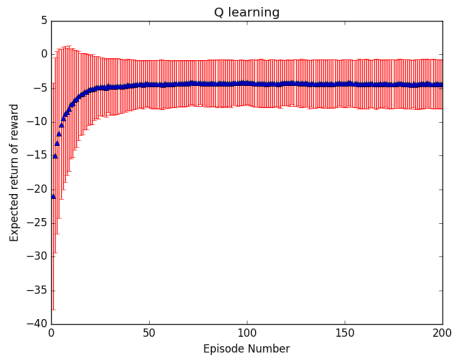


(b)

Table 5: Experiments with multiple actions selected



(a)



(b)

Table 6: Experiments with different values of action selected

algorithms to see how they perform.

References

http://www.mentalmodeler.org
http://levis.sggw.pl/ rew/scenes/pdf/Ozesmi.pdf