**Temperature and Humidity control using a Reinforcement learning agent**

* Dwait Bhatt
* Rachal Agarwal

***MATLAB:***

The problem statement is climate control in a closed environment. Here we are controlling 2 parameters, temperature and humidity in a closed chamber. We have assumed two devices, bulb1 and bulb2 but more devices can be easily incorporated into the code. Details regarding this are mentioned towards the end of this section.

The main algorithm used is Prioritized Sweeping with each Q value update being a one-step tabular Q-learning update. The algorithm is described in section 8.4 of Sutton and Barto’s standard book on Reinforcement Learning (included in the same folder).

The state parameters taken are error in temperature from setpt and error in humidity from setpt, i.e. state = (temp\_err,hum\_err)

temp\_err = temp\_setpt – temp\_current

hum\_err = hum\_setpt – hum\_current

**List of parameters :**

|  |  |
| --- | --- |
| ***Parameter*** | ***Purpose*** |
| temp\_ext, hum\_ext | External Room temperature/humidity before training starts (initial temp/hum) |
| temp\_setpt, hum\_setpt | Temperature and Humidity setpoints (Desired values of temp and hum) |
| norandom | Set to ‘True’ to imply that no random actions are to be taken (training is sufficient if goal is reached with small number of steps with norandom on) |
| maxepisodes | Total number of episodes to train the model for |
| N,M | Size of the range (in degrees) over which the temperature and humidity will vary (ex: for setpt-15 to setpt+15 for temp and hum, N=M=30) |
| K | Coarseness - number of temp (or hum) values which are taken as same state parameter (ex: K=5 means 0-5, 5-10, 10-15 degrees are grouped together by the agent). This parameter is to reduce the state space for faster training. |
| maxsteps | Limit to the number of steps taken without reaching the goal. Current episode will be terminated after these many steps and the next one will begin. |
| alpha\_init | Learning rate used for Q-value update. Separate alpha value is maintained for each (s,a) pair (using Ns(s,a)) but all are initialized the same value, alpha\_init. |
| gamma | Discount factor – factor by which future reward is discounted (scaled down). |
| epsilon | Probability with which a random action is taken (0 when norandom is True). Reduces to 90% for its value each episode. |
| theta | S,A pairs for which TD error > theta are added to the Priority Queue |
| grafic | Set to False to turn off plotting any data recorded during an episode. |

**List of functions :**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Function name*** | ***Purpose*** | ***Note*** | ***Inputs*** |
| TempHumDemo | Governs number of episodes, parameter values (epsilon, alpha, theta, etc.) | This is the main function to be run | temp\_ext, temp\_setpt, hum\_ext, hum\_setpt, Q, norandom |
| BuildActionList | Sets the actions the agent can perform. | Depends on number and type of controllable devices | - |
| GetState | Get the state parameters from the sensor readings and setpts | - | temp\_ext, temp\_setpt, hum\_ext, hum\_setpt |
| Episode\_queue | Algorithm for one episode of Prioritized Sweeping goes here. Also takes care of I/O with the Arduino and plotting of the various graphs. | The actual “training” (updating Q-table) happens here | maxsteps, Q, Model, alpha, gamma, epsilon, theta, statelist, actionlist, grafic, maze, start, goal, temp\_setpt, hum\_setpt, temp\_ext, hum\_ext |
| DiscretizeState | State number (index) from the state parameters | Index of state x in statelist | x, statelist |
| e\_greedy\_selection | Selects a random action or best action according to epsilon greedy strategy | Random action with probability epsilon | Q, s, epsilon |
| GetBestAction | Choose best action to take from Q-table entries for that state | Highest Q(s,a) value gives best action for a given state | Q, s |
| PlotMaze | Plot a grid representing temp and hum on the x and y axes respectively. Current temperature and humidity are marked by the shaded block | The (temp\_setpt, hum\_setpt) pair is the center of the grid. 2 grids are plotted, the second is a zoomed in version of the first about the center. | x, maze, start, goal |
| DoActionOnFridge | Send inputs to the devices which affect temp and hum (here, bulb1 and bulb2) | Sends actions to Arduino via a text file (actions.txt) | action, bulb1\_ip, bulb2\_ip, t |
| ReadFromSensors or ReadRandom | Read temp and hum reading from the sensors or get random values in a range (for testing code in absence of sensors) | Reads the temp and hum from Arduino via text files (temp.txt, hum.txt) | - |
| GetReward | Get reward according to previous and current state | Need to change theta if rewards are changed. Goal is to get the temp\_err and hum\_err to be <5 for 5 time steps | xp, prev\_pos |
| UpdateModel | Update environmental Model which stores predictions of next state and reward | - | s, a, r, sp, Model |
| PrioritizedSweeping | Implements the core algorithm which pushes s,a pairs to the pqueue depending on the size of the update to the backup | Pseudocode in section 8.4 of Sutton and Barto’s book | Q, Model, s, a, r, sp, alpha, gamma, theta, pqueue |
| PlotMaze | Plots the grid depicting temp and hum on the x and y axes | Second plot is a zoomed in version of the first about the centre (temp\_setpt, hum\_setpt) | x,maze,start,goal |

RandomPlanning is a legacy function that we had used earlier to implement updates on previously visited states by selecting them randomly instead of prioritized sweeping. This is called DynaQ and is explained in section 8.2 of Sutton and Barto’s book.

Here the class for the implementation of Priority Queue data structure in MATLAB is Richard T. Guy’s work.

**Adding new devices :**

Functions to be changed :

* BuildActionList (This will change size of Q-table and Model accordingly),
* Episode\_queue (Need to introduce new variables for the inputs to new device in a similar fashion as bulb1\_ip and bulb2\_ip)
* DoActionOnFridge (Change the new variables and print them to actions.txt)

Also, add new plots in Episode\_queue corresponding to the new devices added

NOTE:

Due to the nature of the state parameters (errors in temp, hum from setpt), the agent will need to be trained again if the setpt values are changed (Considering that the environment reacts differently to same device output in different temperature/humidity ranges, ex: an increment in bulb input by 1 gives a different temperature increment from an initial temp of 5o and from 35o). However, changing the external temp, hum values (initial state) does not require training again.

***ARDUINO/PROCESSING :***

The Arduino code reads the sensor data and puts it on the Serial stream every 2 seconds. The Processing code gets this data from Serial and prints 5 such values to temp.txt and hum.txt which will subsequently be read by the MATLAB code.

The actuators or control devices can be controlled by a similar method. A text file, actions.txt will keep updating the inputs that are supposed to be given to the devices. This file can then be read by processing code and used further to control a relay or any other device connected to the Arduino.