CS 237B: Principles of Robot Autonomy II

Problem Set 01

Name: Ricardo Paschoeto
SUID: rp304154

Problem 1: Markovian Drone

(ii) Heatmap of the optimal value function:

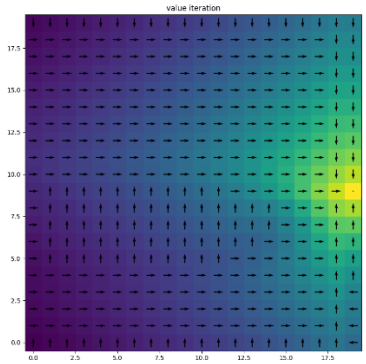

Figure 1 - Optimal Value Function

(iv) Heatmap policy and drone trajectory (Used a customized function from scratch) the highlighted squares indicate the drone trajectory:

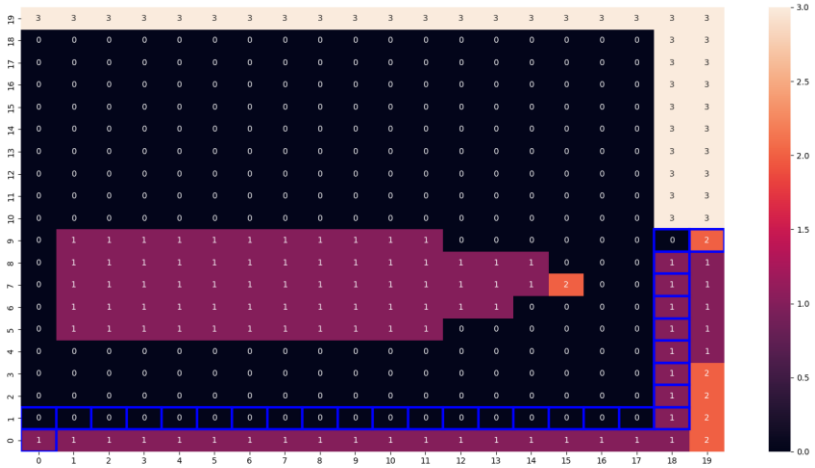

Figure 2 - Policy heatmap.

The policy evaluates the trajectory to the goal point maximizing the reward, avoiding the storm, taking the shortest path possible in that way.

(vi) Bellman Equation of the optimal Q-function (Q-network optimization):

$$Q(x,u) = R(x,u) + (1 - is\_terminal(x)) * gamma * Q(x',u)$$

Where:

- $is\_terminal(x)$: Boolean (1) if x is a terminal state, reducing the equation to - $Q(x,u) = R(x,u)$.

(ix) My example will be given in the Nuclear area: In Nuclear Power Plant we have a group of variables to control the Reactor – Coolant average Temperature, Axial power distributions, Pressure on secondary side. The behavior of these set of variables generates control commands to move the control bar in/out of the Reactor Vessel. We have a historical set of these variable behaviors and actions, so we could train a Q-network with these historical samples and evaluate the results in a nuclear reactor control.

(x)

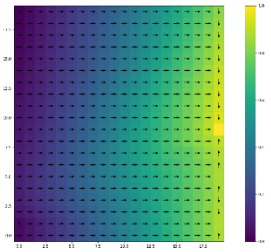

Figure 3 – learning rate = 1e-4.

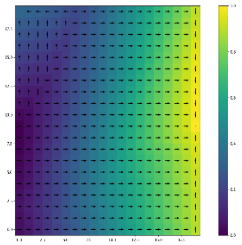

Figure 4 - learning rate = 1e-3.

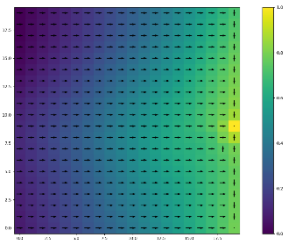

Figure 5 - learning rate = 1e-2.

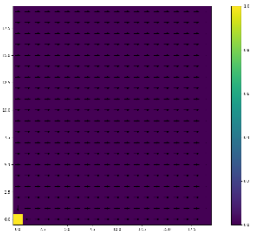

Figure 6 - learning rate = 1e-1.

Problem 2: Classification and Sliding Window Detection

Image Classification

(iv) Tensor Board Model Structure:

- dimension of each “bottleneck” image: [?, 2048]
- parameters (weights + biases) are we optimizing: 15

(v) We could have problem with overfitting when evaluate the training process. And probably need to use a set of dropout layer to avoid it.

Object Detection and Localization

(viii) Detection plot:

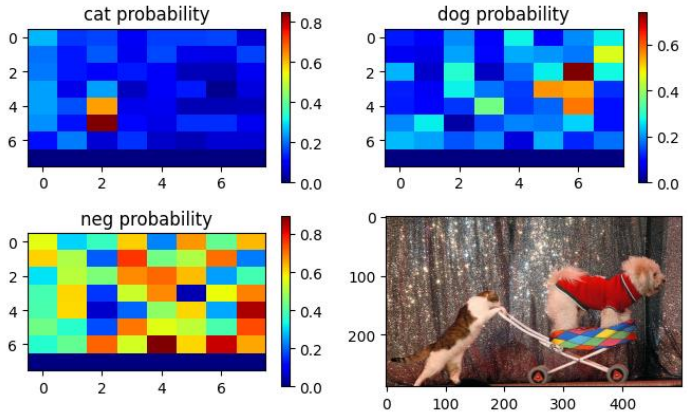

Figure 7 - Sliding Window Classification

(ix) Convolutional layer (mixed_10)

Operation – Convolutional 2D sliding.

Feature vector calculation process – Applying convolution in K x K window image and extract features by classification.

(xiii) Results:

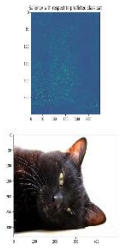

Figure 8 - Saliency for class 'cat'.

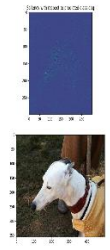

Figure 9 - Saliency for class 'dog'.

Figure 10 - Saliency for class 'neg'.