Modern Artificial Intelligence: Homework 3 *

Deadline: 2021/05/26 00:00am

Compress all of your code and pdf files into a zip file. Then upload it to LMS system on time. Please cite references (e.g., websites or books) if you learn something from them.
NOTICE: DO NOT use any toolbox for this homework. If you use any library, you should make sure it is included correctly. If your code cannot independently run on Prof. Tseng's Matlab. You only got partial points.

1. Q-learning (40%):

As Fig.1 shows, there is a 4×3 world. Its goal is to move to the cell {4,3}. The environment is defined as follows:
$state : s \in S = \{1..4, 1..3\}$
$initial \quad state : s = \{1, 1\}$
$action : a \in A = \{left, up, right, down\}$
$cost : R(s) = -0.04$ when $s' \neq (4, 3)$ or $s' \neq (4, 2)$
$R(s) = +1$ when $s = (4, 3)$
$R(s) = -1$ when $s = (4, 2)$
$discount : \gamma = 0.9$
$terminal : s = \{4, 3\}$
If the robot hits the walls, it will be bounced.
The robot adopts epsilon greedy to explore the environment. The probability of exploitation is 70%. Please implement Q-learning to solve this problem. Assume the Q function is approximated by a linear function. The four features of the approximated Q function are as follows:
(a) constant.
(b) the x position of the robot.
(c) the y position of the robot.
(d) the distance between the goal and the robot.
Please answer the following questions:
1) Display a human-machine interface of the robot's actions and plot weighting values v.s. episode. (20%)
2) Display the Q-values and optimal actions at each episode. (20%)

Code Delivery:

Matlab code (.m file). The file name should be *Q_learning.m*. Compress all of your code into a zip file. This code should display: The optimal action and utility value in each state at each episode until converge. Please display at least 20 episodes.

*The homework and the programming assignment must NOT be the result of cooperative work. Each student must work individually in order to understand the material in depth. You CANNOT copy the homework or the

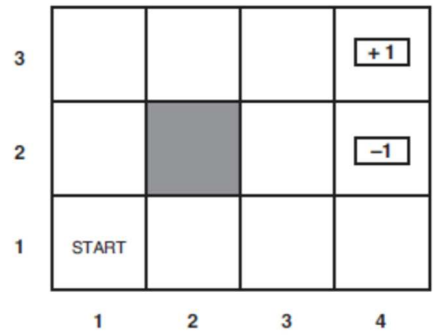

Figure 1: Illustration of 4×3 world.

2. Supervised learning (40%):

Given a leg dataset including laser data X and labels Y, the data information is as follows:
There are 720 laser data points at each time step. The laser data was divided into several segments S_t^i, where t denotes the time step and i denote the i-th segments. The data is recorded in 120 seconds. The first 60 data is training data and the other data is testing data. The 5 features of data is as follows:
(a). The number of points in each segment:
$n = |S_t^i|$
(b). The standard deviation of each segment:
$\sigma = \sqrt{\frac{1}{n}\sum_j ||\boldsymbol{x_j} - \bar{\boldsymbol{x}}||^2}$, where $\mathbf{x}$ is the x-y data.
(c). The width of each segment:
(d). The circularity (s_c) of each segment:
A circle is represented as $(x-x_c)^2+(y-y_c)^2 = r_c^2$. The unknown variable x' is $[x_c, y_c, (x_c^2+y_c^2-r_c^2)]$. x' can be solved by pseudo inverse $x' = (A^TA)^{-1}A^Tb$, where

$$A = \begin{bmatrix} -2x_1 & -2y_1 & 1 \\ -2x_2 & -2y_2 & 1 \\ \vdots & \vdots & \vdots \\ -2x_n & -2y_n & 1 \end{bmatrix}, b = \begin{bmatrix} -x_1^2 - y_1^2 \\ -x_2^2 - y_2^2 \\ \vdots \\ -x_n^2 - y_n^2 \end{bmatrix},$$

$s_c = \sum_{i=1}^n [r_c - \sqrt{(x_c - x_i)^2 + (y_c - y_i)^2}]^2$
(e). The radius (r_c) of each segment:
The r_c can be computed as (d) mentioned.

The goal is to detect a segment is a leg or not. Write four supervised learning algorithms to classify the training and testing data. Display the confusion table of training and test data for each algorithm. Plot cost function v.s. iteration for perceptron and logistic regression.
1) Naive Bayes. (8%)
2) Perceptron. (8%)
3) Logistic regression. (8%)
4) Adaboost. (8%)

---

programming assignment of somebody else.

5) Compare and explain each algorithm's performance. (8%)

Code Delivery:

Matlab code (.m file). The file name should be $NB.m$, $Perceptron.m$, $logistic.m$ and $Adaboost.m$. Compress all of your code into a zip file.

3. LRTA* and Q-learning (20%):

Prove LRTA* is a special case of Q-learning.

$H(s) = min[c(s, a, s') + h(s')]....LRTA^*$

$Q(s, a) = [R(s) + \gamma max_{a'} Q(s', a')]...Q - learning$

$c(s, a, s')$: cost from the state s and action a to the state state s'.

$R(s)$: reward at the state s.