6. Q-learning with linear function approximation

Bookmark this page

Project due May 10, 2023 08:59 -03 Completed

Since the state displayed to the agent is described in text, we have to choose a mecha descriptions into vector representations. A naive way is to create one unique index for we have done in previous part. However, such approach becomes infeasible when the huge. To tackle this challenge, we can design some representation generator that does textual state space. In particular, a representation generator $\psi_R(\cdot)$ reads raw text disp converts it to a vector representation $v_s = \psi_R(s)$. One approach is to use a bag-of-w derived from the text description.

In large games, it is often impractical to maintain the Q-value for all possible state-acti this problem is to approximate $Q(s, c)$ using a parametrized function $Q(s, c; \theta)$.

In this section we consider a linear parametric architecture:

$$Q(s, c; \theta) = \phi(s, c)^T \theta = \sum_{i=1}^{d} \phi_i(s, c)\, \theta_i,$$

where $\phi(s, c)$ is a fixed feature vector in $\mathbb{R}^d$ for state-action pair (s, c) with i-th comp , and $\theta \in \mathbb{R}^d$ is a parameter vector that is shared across state-action pairs. The key ch the feature vectors $\phi(s, c)$. Note that given a textual state s, we first translate it to a v using $\psi_R(s)$. So the question here is how to design a mapping function convert $(\psi_R($ representation in $\mathbb{R}^d$. Assume that the size of action space is d_C, and the dimension of state representation is d_R.

Feature engineering

1/1 point (graded)

Exercise: Consider the following feature engineering. Define a function $\psi_C : \mathcal{C} \to \mathbb{R}^{d_C}$ component $\psi_{C,j}(c)$ is given as follows:

$$\psi_{C,j}(c) = \begin{cases} 1 & \text{if } j = c \\ 0 & \text{else} \end{cases}$$

The feature vector is defined as

Alternatively, consider the following feature map: , where
, and for ,

$$\begin{bmatrix} \\ \vdots \\ \mathbf{0} \end{bmatrix}$$

You will implement this feature map in the next tab.

Computing theta update rule

1.0/1 point (graded)

The Q-learning approximation algorithm starts with an initial parameter estimate of . A
Q-learning, upon observing a data tuple , the target value for the Q
defined as the sampled version of the Bellman operator,

Then the parameter is simply updated by taking a gradient step with respect to the s

Previous Next

The negative gradient can be computed as follows:

(Enter your answer in terms of `y`, `Q(s, c, theta)`, and `phi(s, c)`.)

In the alternative feature map there seems to be a problem displaying a part of the text: 'Could not format H

Invalid Input: \'d\' not permitted in answer as a variable

Hi all, I get the messageInvalid Input: \'d\' not permitted in answer as a variable even when I have no 'd' in my

? What is the shape of the feature vector/matrix?

? Why is there no "Show answer" here?

I got the answer wrong for the first question here and while I think I understand now why it is wrong, a detail

? Specifics of the feature array phi

Is the structure of the feature array such that for each state description, only a range of the rows have nonze

y = R(s,c) + gamma * Q(s', c') ?

Connect

Blog
Contact Us
Help Center
Security
Media Kit

GET IT ON Google play

Download on the App Store

© 2023 edX LLC. All rights reserved.
深圳市恒宇博科技有限公司 粤ICP备17044299号-2