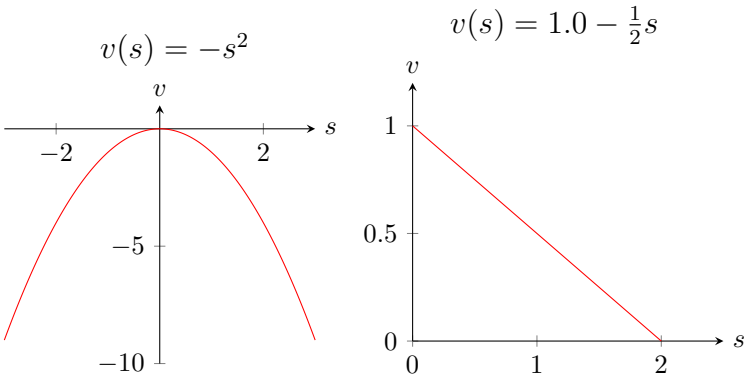
CMPUT 365: Introduction to Reinforcement Learning, Winter 2023
Worksheet #10: Constructing Features for Prediction

Manuscript version: *#6537ce* - 2023-04-06 19:16:21-06:00

Question 1. Consider the following two functions.



1. Design features for each function, to approximate them as a linear function of these features. Can you design features to make the approximation exact?
2. Can you design one set of features, that allows you to represent both functions?

Question 2. Consider a (one hidden layer) neural network, where the input *column* vector $\mathbf{s}$ is mapped by the input-weight matrix $\mathbf{B}$ and a (differentiable) activation function g to the "feature vector"

$$\mathbf{x} := g(\boldsymbol{\psi}), \qquad \text{where } \boldsymbol{\psi} := \mathbf{B}\mathbf{s}.$$

(We assume, that the function g is applied elementwise to the vector $\boldsymbol{\psi}$, i.e. the above equation is equivalent to saying $x_i = g(\psi_i)$ for all i.) Finally, the feature vector is mapped by the output-weight vector $\mathbf{w}$ linearly to the value estimate

$$\hat{v} := \mathbf{w}^\top \mathbf{x}.$$

(Here, we use c_i to denote the ith element of the vector $\mathbf{c}$, and use a_{ij} to denote the element at the ith row and the jth column of a matrix $\mathbf{A}$.)

Now using chain rule, we can compute the following derivatives (how?):

$$\begin{aligned}
\frac{\partial \hat{v}}{\partial w_j} &= x_j, \\
\frac{\partial \hat{v}}{\partial B_{ij}} &= w_i \frac{\partial x_i}{\partial B_{ij}} = w_i \frac{\partial g(\psi_i)}{\partial \psi_i} s_j,
\end{aligned}$$

where in the first equality on the second line, we used the fact that x_k for $k \neq i$ would be independent of B_{ij} (why?)

For the following sub-questions, let the activation function be ReLU, which for $x \in \mathbb{R}$ is defined as

$$f(x) = \begin{cases} x & \text{if } x \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

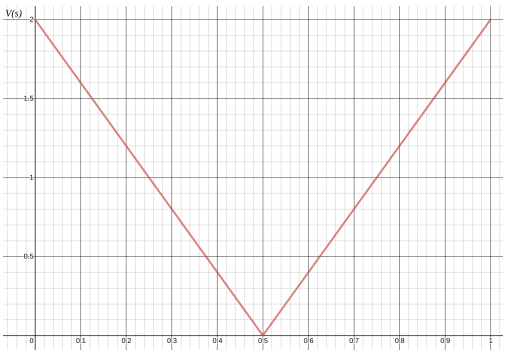
1. What is the derivative of the ReLU activation function g (this is a trick question)? When are the derivatives well-defined?

2. Imagine that all the weights (i.e. $\mathbf{B}$ and $\mathbf{w}$) are initialized to zero. Would this be a problem? If so, why?

Question 3. Consider a problem with the state space, $\mathcal{S} = \{0, 0.01, 0.02, \ldots, 0.99, 1\}$. Assume that the true value function is

$$v_\pi(s) = 4|s - 0.5|$$

which is visualized below.



Let us now approximate v_π using state aggregation. We choose to aggregate the states into two bins: $[0, 0.5]$ and $(0.5, 1]$. (This means that the value function learns, say, $w_1 \in \mathbb{R}$ for all $s \in [0, 0.5]$ and learns, say, $w_2 \in \mathbb{R}$ for all $s \in [0.5, 1]$. Also let $\mathbf{w} = (w_1, w_2)^\top$.) Then

1. Give one set of possible features $\mathbf{x}$, such that a linear approximation to v_π (i.e. $v_{\mathbf{w}}(s) = \mathbf{w}^\top \mathbf{x}(s)$) is equivalent to the value estimate learned using the (two bin) state aggregation technique described above.

2. Imagine that you minimize the (weighted) mean squared value error

$$\overline{\text{VE}}(\mathbf{w}) := \sum_{s \in \mathcal{S}} d(s)(v_\pi(s) - \hat{v}(s, \mathbf{w}))^2,$$

with a uniform weighting $d(s) = \frac{1}{101}$ for all $s \in \mathcal{S}$. Here $\hat{v}$ represents the value function estimate under the state-aggregation scheme described before. Then compute

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^2} \overline{\text{VE}}(\mathbf{w}).$$

3. Now if the agent puts all of the weighting on the range $[0, 0.25]$ (i.e. $d(s) = 0$ for all $s \in (0.25, 1]$ and $d(s)$ is uniform otherwise), then what vector $\mathbf{w}^*$ is found by minimizing $\overline{\text{VE}}$?

Question 4. Consider the following general SGD update rule with a general target U_t

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \left[U_t - \hat{v}(S_t, \mathbf{w}_t)\right] \nabla \hat{v}(S_t, \mathbf{w_t}).$$

Assume that we are using linear function approximation, i.e. $\hat{v}(S, \mathbf{w}) = x(S)^\top \mathbf{w}$.

1. What happens to the update, if we scale the features by a constant. In particular, we use the new features $\tilde{x}(S) = 2x(S)$. Will this pose any problem? Why or why not?

2. In general, we want a stepsize that is invariant to the magnitude of the feature vector $x(S)$, where the magnitude is measured by the inner product $x(S)^\top x(S)$. The book suggests using the stepsize

$$\alpha = \frac{1}{\tau x(S)^\top x(S)}.$$

   What is $x(S)^\top x(S)$ when using tile coding with 10 tilings? Suppose $\tau = 1000$. What is α if we use tile coding with 10 tilings?

Question 5. A student is tasked to program an agent for a 100×100 grid-world problem, with the states denoted by (x, y) and $x, y \in \{0, 1, \ldots, 99\}$. The goal of this problem is to reach the state $(99, 99)$, located at the top-right corner. In this grid-world problem, rewards are -1 at each time step, except for the terminal state where the reward is 0 and the episode ends. The student first considers a tabular feature representation. However, since the grid-world has a large number of states, the student is worried about generalization.

1. What is the dimension of the tabular representation? Suppose the student chooses to represent a state by its (x, y) coordinates instead. What is the dimension of this new representation? How does it differ from the tabular representation?

2. Design a function $f(x, y)$ as features for linear Temporal Difference (TD) learning that, where (x, y) are the state coordinates, can improve the agent's performance in this fixed grid-world problem.

3. Suppose that the student designs a new feature representation $\mathbf{x}'(s) = (x, y, z)$, where the first two components are the row and column indices, and the last component is binary: $z = 1$ when the agent is 2 squares away from the goal and $z = 0$ otherwise. What is the dimension of feature vector in this case? How much will this help with generalization instead of only using $\mathbf{x}(s) = (x, y)$?

Question 6. (*Exercise 9.2 S&B; modification of*) Consider an MDP where each state is a k-dimensional vector $(s_1, s_2, \ldots, s_k)^\top \in \mathbb{R}^k$. For this k-dimensional state space, an order-n polynomial-basis feature $x_i : \mathcal{S} \to \mathbb{R}$ can be written as

$$x_i(s) = \prod_{j=1}^{k} s_j^{c_{ij}},$$

where $c_{ij} \in \{0, 1, \ldots, n\}$ and $0 \leq \sum_{j=1}^{k} c_{ij} \leq n$. (As an example, if $c_{ij} = 0$ for all j, then the corresponding feature would be $x_i(s) = 1$. Another feature could be $x_i(s) = s_1$, if we let $c_{i1} = 1$ and $c_{ij} = 0$ for $j \neq 1$.)

These features make up the order-n polynomial basis for dimension k. Can you count the total number of distinct features x_is we can create by varying $(c_{ij})_{ij}$s?

(For a simple example, let $k = 2$ and $n = 3$. Then we have 10 distinct features:

$$(1, \quad s_1, \quad s_2, \quad s_1 s_2, \quad s_1^2, \quad s_2^2, \quad s_1^2 s_2, \quad s_1 s_2^2, \quad s_1^3, \quad s_2^3)^\top.$$

Note that this definition of order-n differs from the definition used by the book. Can you figure out the difference?)

Question 7. (*Exercise 9.3 S&B; modification of*) What n and c_{ij} produce the below feature vector?

$$\mathbf{x}(s) = (1, s_1, s_2, s_1 s_2, s_1^2, s_2^2, s_1 s_2^2, s_1^2 s_2, s_1^2 s_2^2)$$

Question 8. (*Exercise 9.4 S&B*) Suppose we believe that one of two state dimensions is more likely to have an effect on the value function than is the other, that generalization should be primarily across this dimension rather than along it. What kind of tilings could be used to take advantage of this prior knowledge?
