Exercises - Chapter 9

Carl Fredriksson, c@msp.se

Exercise 9.1

Show that tabular methods such as presented in Part I of this book are a special case of linear function approximation. What would the feature vectors be?

My answer:

The general SGD update for linear function approximation:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \Big[U_t - \hat{v}(S_t, \mathbf{w}_t) \Big] \mathbf{x}(S_t)$$

We can take state aggregation to the extreme, with one estimated value (one component of the weight vector $\mathbf{w}$) for each state: $\mathbf{w}_i = V(s_i)$. The feature vectors would also have one element per state, with $\mathbf{x}(s_i)$ being a one-hot vector with a 1 at component i and 0 everywhere else. We have

$$\hat{v}(S_t, \mathbf{w}_t) = \mathbf{w}_t^\top \mathbf{x} = V(S_t)$$

and thus the update becomes

$$V(S_t) \leftarrow V(S_t) + \alpha \Big[U_t - V(S_t) \Big]$$

which is the general form of the update for tabular methods.

Exercise 9.2

Why does (9.17) define $(n+1)^k$ distinct features for dimension k?

My answer:

$n = 0, k = 1 : (s_1^0) = (1)$

$n = 1, k = 1 : (s_1^0, s_1^1) = (1, s_1)$

$n = 2, k = 1 : (s_1^0, s_1^1, s_1^2) = (1, s_1, s_1^2)$

$n = 0, k = 2 : (s_1^0 s_2^0) = (1)$

$n = 1, k = 2 : (s_1^0 s_2^0, s_1^0 s_2^1, s_1^1 s_2^0, s_1^1 s_2^1) = (1, s_2, s_1, s_1 s_2)$

$n = 2, k = 2 : (s_1^0 s_2^0, s_1^0 s_2^1, s_1^0 s_2^2, s_1^1 s_2^0, s_1^1 s_2^1, s_1^1 s_2^2, s_1^2 s_2^0, s_1^2 s_2^1, s_1^2 s_2^2) =$
$(1, s_2, s_2^2, s_1, s_1 s_2, s_1 s_2^2, s_1^2, s_1^2 s_2, s_1^2 s_2^2)$

We have k numbers s_i that all have $n+1$ possible exponents $0, 1, \ldots, n$. Combining all of the numbers and possible exponents we get $(n+1)^k$ permutations.

Exercise 9.3

What n and $c_{i,j}$ produce the feature vectors $\mathbf{x}(s) = (1, s_1, s_2, s_1 s_2, s_1^2, s_2^2, s_1 s_2^2, s_1^2 s_2, s_1^2 s_2^2)^\top$

My answer:

$n = 2, k = 2$

$c_{0,1} = 0, c_{0,2} = 0, c_{1,1} = 1, c_{1,2} = 0, c_{2,1} = 0, c_{2,2} = 1, c_{3,1} = 1, c_{3,2} = 1, c_{4,1} = 2, c_{4,2} = 0, c_{5,1} = 0, c_{5,2} = 2, c_{6,1} = 1, c_{6,2} = 2, c_{7,1} = 2, c_{7,2} = 1, c_{8,1} = 2, c_{8,2} = 2$

Gives us

$x_0(s) = \prod_{j=1}^{k} s_j^{c_{i,j}} = s_1^{c_{0,1}} s_2^{c_{0,2}} = s_1^0 s_2^0 = 1$

$x_1(s) = s_1^{c_{1,1}} s_2^{c_{1,2}} = s_1^1 s_2^0 = s_1$

$x_2(s) = s_1^{c_{2,1}} s_2^{c_{2,2}} = s_1^0 s_2^1 = s_2$

$x_3(s) = s_1^{c_{3,1}} s_2^{c_{3,2}} = s_1^1 s_2^1 = s_1 s_2$

$x_4(s) = s_1^{c_{4,1}} s_2^{c_{4,2}} = s_1^2 s_2^0 = s_1^2$

$x_5(s) = s_1^{c_{5,1}} s_2^{c_{5,2}} = s_1^0 s_2^2 = s_2^2$

$x_6(s) = s_1^{c_{6,1}} s_2^{c_{6,2}} = s_1^1 s_2^2 = s_1 s_2^2$

$x_7(s) = s_1^{c_{7,1}} s_2^{c_{7,2}} = s_1^2 s_2^1 = s_1^2 s_2$

$x_8(s) = s_1^{c_{8,1}} s_2^{c_{8,2}} = s_1^2 s_2^2 = s_1^2 s_2^2$

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

Exercise 9.4

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?

My answer:

Tilings with tiles that are elongated across the state dimension that is more likely to have an effect on the value function. For example stripes or rectangular tiles.

Exercise 9.5

Suppose you are using tile coding to transform a seven-dimensional continuous state space into binary feature vectors to estimate a state value function $\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$. You believe that the dimensions do not interact strongly, so you decide to use eight tilings of each dimension separately (stripe tilings), for $7 \times 8 = 56$ tilings. In addition, in case there are some pairwise interactions between the dimensions, you also take all $\binom{7}{2} = 21$ pairs of dimensions and tile each pair conjunctively with rectangular tiles. You make two tilings for each pair of dimensions, making a grand total of $21 \times 2 + 56 = 98$ tilings. Given these feature vectors, you suspect that you still have to average out some noise, so you decide that you want learning to be gradual, taking about 10 presentations with the same feature vector before learning nears its asymptote. What step-size parameter α should you use? Why?

My answer:

With tile coding, exactly one feature (corresponding to one tile in the tiling) is active (=1) in a tiling at one time, and all other features are inactive (=0). Thus the number of active features is always the same as the number of tilings. We have:

$$\alpha \doteq (\tau \mathbb{E}[\mathbf{x}^\top \mathbf{x}])^{-1} = (10 \times 98)^{-1} = 980^{-1} \approx 0.001$$

Exercise 9.6

If $\tau = 1$ and $\mathbf{x}(S_t)^\top \mathbf{x}(S_t) = \mathbb{E}[\mathbf{x}^\top \mathbf{x}]$, prove that (9.19) together with (9.7) and linear function approximation results in the error being reduced to zero in one update.

My answer:

We have

$$\alpha \doteq (\tau \mathbb{E}[\mathbf{x}^\top \mathbf{x}])^{-1} = \frac{1}{\mathbf{x}(S_t)^\top \mathbf{x}(S_t)}$$

and

$$\begin{aligned}
\mathbf{w}_{t+1} &\doteq \mathbf{w}_t + \alpha \big[U_t - \hat{v}(S_t, \mathbf{w}_t)\big] \nabla \hat{v}(S_t, \mathbf{w}_t) \\
&= \mathbf{w}_t + \frac{1}{\mathbf{x}(S_t)^\top \mathbf{x}(S_t)} \big[U_t - \hat{v}(S_t, \mathbf{w}_t)\big] \mathbf{x}(S_t)
\end{aligned}$$

We are trying to prove

$$U_t - \hat{v}(S_t, \mathbf{w}_{t+1}) = 0 \iff \hat{v}(S_t, \mathbf{w}_{t+1}) = U_t$$

Proof:

$$\begin{aligned}
\hat{v}(S_t, \mathbf{w}_{t+1}) &\doteq \mathbf{w}_{t+1}^\top \mathbf{x}(S_t) \\
&= \left(\mathbf{w}_t + \frac{1}{\mathbf{x}(S_t)^\top \mathbf{x}(S_t)} \Big[U_t - \hat{v}(S_t, \mathbf{w}_t) \Big] \mathbf{x}(S_t) \right)^\top \mathbf{x}(S_t) \\
&= \mathbf{w}_t^\top \mathbf{x}(S_t) + \frac{1}{\mathbf{x}(S_t)^\top \mathbf{x}(S_t)} \Big[U_t - \hat{v}(S_t, \mathbf{w}_t) \Big] \mathbf{x}(S_t)^\top \mathbf{x}(S_t) \\
&= \hat{v}(S_t, \mathbf{w}_t) + U_t - \hat{v}(S_t, \mathbf{w}_t) \\
&= U_t
\end{aligned}$$

Exercise 9.7

One of the simplest artificial neural networks consists of a single semi-linear unit with a logistic nonlinearity. The need to handle approximate value functions of this form is common in games that end with either a win or a loss, in which case the value of a state can be interpreted as the probability of winning. Derive the learning algorithm for this case, from (9.7), such that no gradient notation appears.

My answer:

Let $z = \mathbf{w}^\top \mathbf{x}(s)$, then we have

$$\hat{v}(s, \mathbf{w}) \doteq \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}(s)}} = \frac{1}{1 + e^{-z}}$$

We can compute the partial derivatives of $\hat{v}(s, \mathbf{w})$ with respect to the components w_i of $\mathbf{w}$

$$\begin{aligned}
\frac{\delta \hat{v}(s, \mathbf{w})}{\delta w_i} &= \frac{\delta}{\delta w_i}\left(\frac{1}{1+e^{-z}}\right) \\
&= \frac{\delta}{\delta z}\left(\frac{1}{1+e^{-z}}\right)\frac{\delta z}{\delta w_i} \\
&= \frac{\delta}{\delta z}\left(\frac{1}{1+e^{-z}}\right)\frac{\delta}{\delta w_i}\left(\mathbf{w}^\top \mathbf{x}(s)\right) \\
&= \frac{e^{-z}}{(1+e^{-z})^2} x_i(s)
\end{aligned}$$

Thus we have

$$\nabla \hat{v}(S_t, \mathbf{w}_t) = \frac{e^{-z}}{(1+e^{-z})^2}\mathbf{x}(S_t)$$

which gives us

$$\begin{aligned}
\mathbf{w}_{t+1} &\doteq \mathbf{w}_t + \alpha\Big[U_t - \hat{v}(S_t, \mathbf{w}_t)\Big]\nabla \hat{v}(S_t, \mathbf{w}_t) \\
&= \mathbf{w}_t + \alpha\Big[U_t - \hat{v}(S_t, \mathbf{w}_t)\Big]\frac{e^{-z}}{(1+e^{-z})^2}\mathbf{x}(s) \\
&= \mathbf{w}_t + \alpha\Big[U_t - \hat{v}(S_t, \mathbf{w}_t)\Big]\frac{e^{-\mathbf{w}^\top \mathbf{x}(s)}}{(1+e^{-\mathbf{w}^\top \mathbf{x}(s)})^2}\mathbf{x}(s)
\end{aligned}$$

Exercise 9.8

Arguably, the squared error used to derive (9.7) is inappropriate for the case treated in the preceding exercise, and the right error measure is the cross-entropy loss (which you can find on Wikipedia). Repeat the derivation in Section 9.3, using the cross-entropy loss instead of the squared error in (9.4), all the way to an explicit form with no gradient or logarithm notation in it. Is your final form more complex, or simpler, than that you obtained in the preceding exercise?

My answer:

Since we have two outcomes (win and lose), we can write the cross-entropy loss for state s as

$$f(s, \mathbf{w}) \doteq -\Big(v_\pi(s)\log\big(\hat{v}(s, \mathbf{w})\big) + \big(1 - v_\pi(s)\big)\log\big(1 - \hat{v}(s, \mathbf{w})\big)\Big)$$

Let $z = \mathbf{w}^\top \mathbf{x}(s)$, and let's denote $v_\pi(s)$ by v_π and $\hat{v}(s, \mathbf{w})$ by $\hat{v}$ for brevity. We can then compute the partial derivatives with respect to the components w_i of $\mathbf{w}$ as follows

$$
\begin{aligned}
\frac{\delta f(s, \mathbf{w})}{\delta w_i} &= -\frac{\delta}{\delta w_i}\Big(v_\pi \log(\hat{v}) + (1 - v_\pi)\log(1 - \hat{v})\Big) \\
&= -\left(v_\pi \frac{\delta \log(\hat{v})}{\delta w_i} + (1 - v_\pi)\frac{\delta \log(1 - \hat{v})}{\delta w_i}\right) \\
&= -\left(v_\pi \frac{\delta \log(\hat{v})}{\delta \hat{v}}\frac{\delta \hat{v}}{\delta w_i} + (1 - v_\pi)\frac{\delta \log(1 - \hat{v})}{\delta \hat{v}}\frac{\delta \hat{v}}{\delta w_i}\right) \\
&= -\left(v_\pi \frac{1}{\hat{v}}\frac{\delta \hat{v}}{\delta w_i} + (1 - v_\pi)\frac{-1}{1 - \hat{v}}\frac{\delta \hat{v}}{\delta w_i}\right) \\
&= -\left(\frac{v_\pi}{\hat{v}} - \frac{1 - v_\pi}{1 - \hat{v}}\right)\frac{\delta \hat{v}}{\delta w_i} \\
&= -\frac{v_\pi(s)(1 - \hat{v}) - \hat{v}(1 - v_\pi)}{\hat{v}(1 - \hat{v})}\frac{\delta \hat{v}}{\delta w_i} \\
&= \frac{\hat{v} - v_\pi}{\hat{v}(1 - \hat{v})}\frac{\delta \hat{v}}{\delta w_i} \\
&= \frac{\hat{v} - v_\pi}{\left(\frac{1}{1+e^{-z}}\big(1 - \frac{1}{1+e^{-z}}\big)\right)}\frac{e^{-z}}{(1 + e^{-z})^2}x_i(s) \\
&= \frac{(\hat{v} - v_\pi)e^{-z}x_i(s)}{1 + e^{-z} - 1} \\
&= (\hat{v} - v_\pi)x_i(s)
\end{aligned}
$$

Thus we have

$$
\nabla f(s, \mathbf{w}) = \Big[\hat{v}(s, \mathbf{w}) - v_\pi(s)\Big]\mathbf{x}(s)
$$

Which gives us the stochastic gradient-descent update

$$
\begin{aligned}
\mathbf{w}_{t+1} &\doteq \mathbf{w}_t - \alpha \nabla f(S_t, \mathbf{w}) \\
&= \mathbf{w}_t - \alpha\Big[\hat{v}(S_t, \mathbf{w}_t) - v_\pi(S_t)\Big]\mathbf{x}(S_t) \\
&= \mathbf{w}_t + \alpha\Big[v_\pi(S_t) - \hat{v}(S_t, \mathbf{w}_t)\Big]\mathbf{x}(S_t) \\
&= \mathbf{w}_t + \alpha\Big[U_t - \hat{v}(S_t, \mathbf{w}_t)\Big]\mathbf{x}(S_t)
\end{aligned}
$$

This final form is simpler than what I obtained in the previous exercise and can be recognized from the linear case (Section 9.4).