## 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?*

## Exercise 9.2

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

For k state variables we have n + 1 possible values for $c_{i,j}$. Thus, the total combinations of states and values are $(n+1)^k$, which is the same as the number of distinct features possible to have.

## Exercise 9.3

*What n and ci,j produce the feature vectors x(s) = (...)*

```
n = 2 c = [0 0 1 0 0 1 1 1 2 0 0 2 2 1 1 2 2 2]
```

Where $c_{i,j}$ being 0 for both state k is 1 instead of 0.

## Exercise 9.4

*Suppose we believe that one of two state dimensions is more likely to have an e↵ect 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?*

Log tilings along the dimension believed to have more of an effect on the value function combined with regular square tilings.

This way we have more generalization along the desired dimension but still retain the ability to have seperate values for different combinations between the dimensions.

## 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 vˆ(s,w) ⇡ v⇡(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 ⇥ 8 = 56 tilings. In* *addition, in case there are some pairwise interactions between the dimensions, you also take all 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 ⇥ 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 ↵ should you use? Why?*

Since we want to learn near its asymptote in 10 presentations of the same feature vector, $\tau = 10$.

The expected value for $E[x^T x]$ is equivalent to the sum of all values in x that is expected.

Since we are doing tiling, we know that for each dimension of the feature vector, it will belong to at least one part of each tiling.

For the stripe tilings, each 7 dimensions will belong to one of 1/w stripes for 8 tiles where 1/w is how many stripes the dimension has been split up into.

Thus, we can always expect 7 * 8 = 49 one hot encodings of 1 and the rest to be 0.

Using the same logic, for the rectangular tilings we know we can expect a total value of 7 * 7 for each pairing * 2 because there are 2 tiles.

Thus $E[x^T x] = 98$ and according to our guidelines $\alpha = (\tau E[x^T x])^{-1} = (10 * 98)^{-1} = \frac{1}{980}$

## Exercise 9.6

*If $\tau$ = 1, prove that (9.19) together with (9.7) results in the error being reduced to zero in one update.*

## 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.*

## 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?*