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? □

Tabular methods store an independent value for each state. Therefore in order to mimic this behavior as a special case of the linear function approximation the feature vector must be a "one-hot", i.e. all zeros expect for the i-th element. This feature vector with one at that i-th position corresponds to state s_i. The learned weight w_i then matches the value function in the tabular case.

$$v(s_k, \vec{w}) = \vec{x}(s_k) \cdot \vec{w} = \sum_{i=1}^{n} x_i(s_k)\, w_i =$$

$$= \sum_{i=1}^{n} \delta_{ik} w_i = w_k$$

$\vec{x}(s_k)$ is one hot, i.e. $[0 \dots 1 \dots 0]$ ← k-th position

s_i ... i-th state
v ... value function
$\vec{w}$ ... weights
$\vec{x}$ ... features

Exercise 9.2 Why does (9.17) define $(n+1)^k$ distinct features for dimension k? □

Indexing of exponents starts at 0, therefore there i n+1 available choices for every dimension of the polynomial exponent.

Exercise 9.3 What n and $c_{i,j}$ produce the feature vectors $\mathbf{x}(s) = (1, s_1, s_2, s_1 s_2, s_1^2, s_2^2, s_1 s_2^2, s_1^2 s_2, s_1^2 s_2^2)^\top$? □

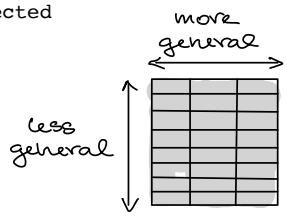
$$x_i(s) = \prod_{j=1}^{k} s_j^{c_{i,j}} \qquad i \in \{1, \dots, (n+1)^k\} \quad j \in \{1, \dots, k\}$$

n ... # features (=2)
k ... # states (=2)

$$c_{i,j} = \begin{array}{c} \\ j \\ \\ \\ \end{array} \begin{array}{c} \overset{1 \longleftarrow\ i\ \longrightarrow (n+1)^k}{} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 2 & 0 & 1 & 2 & 2 \\ 0 & 0 & 1 & 1 & 0 & 2 & 2 & 1 & 2 \end{bmatrix} \\ \\ \end{array} \begin{array}{l} 1 \\ \\ \\ k \end{array}$$

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? □

Asymmetrical blocks with much denser discretization along the suspected important dimension



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? □

Let's start with the value suggested in the paragraph above the exercise. Optimal learning rate is generally problem specific but this may give us a good initial value to tune later.

$$\alpha = \frac{1}{\tau \mathbb{E}[\vec{x}\cdot\vec{x}]} = \frac{1}{10 \cdot 20} = 1/200$$

$$\mathbb{E}[\vec{x}\cdot\vec{x}] = \sum_{i=1}^{98} \mathbb{E}[x_i^2] = \sum_{i=1}^{98} p(x_i)\, x_i^2 =$$

$$= \sum_{j=1}^{56} \underbrace{p(x_j)}_{8/56} \underbrace{x_j^2}_{1^2} + 2 \cdot \sum_{l=1}^{21} \underbrace{p(x_l)}_{6/21} \underbrace{x_l^2}_{1^2} =$$

For every x only 8 tilings have non-zero feature value

stripe tiles

pair-wise tiles

There's 6 non-zero pair-wise interaction for any x

$$= 8 + 12 = 20$$