Networks for Key-finding

5.1 Key-Finding

5.1.1 Key-Finding Outlined

As was noted earlier in Chapter 1, one of the most important discoveries about music cognition is that listeners mentally represent music using a tonal hierarchy (Krumhansl, 1979, 1990a; Krumhansl, Bharucha, & Kessler, 1982; Krumhansl & Shepard, 1979). The tonal hierarchy organizes the pitches of a particular key to reflect the fact that some are more important than others are, and that other tones must be considered in relation to these stable pitches. For instance, if a musical context establishes a specific musical key, then the best-fitting note is the tonic of that key (i.e., the pitch in position 1 of the scale). For example, in the musical key of C major the note C is the most stable. The next best tones are those in either the 3 or 5 positions of the key’s scale. In the key of C major, these are the notes E or G. Less apt than these two notes are any of the set of remaining notes that belong to the scale. In the context of C major, these are the notes D, F, A, or B. Finally, the least stable tones are the set of five pitch-classes that do not belong to the context’s scale. For C major, these are C♯, D♯, F♯, G♯, and A♯.

Krumhansl (1990a) employs the tonal hierarchy as the basis for a cognitive theory of music perception. Her central claim is that music perception does not simply depend upon music’s acoustic properties but also depends upon the mental organization of musical sounds. Listeners have precise knowledge about the structure of tonal music, and use this knowledge to understand music. One consequence of Krumhansl’s theory is that listeners must possess a fundamental ability to identify the tonal centre of music. If one cannot identify a musical key, then one cannot use tonal hierarchies to organize music. Human listeners are indeed able to infer musical key on the basis of very little musical evidence (Butler, 1989).

Given the importance of the tonal hierarchy in musical cognition, and the consequent importance of identifying tonal centres, it is not surprising that many
researchers propose theories about how listeners identify musical keys. We call these theories about key-finding or tonal implication. When one hears a musical selection in the key of C major, what procedure does one use to infer that the piece is indeed in that key, to permit using the C major tonal hierarchy to understand the selection’s tonal structure? A key-finding theory aims to answer this question.

One influential theory of key-finding is proposed by Krumhansl and Schmuckler, and described in detail in Krumhansl’s seminal work *Cognitive Foundations of Musical Pitch* (Krumhansl, 1990a). This algorithm, described in more detail later in this chapter, compares a musical input to templates associated with different musical keys. These templates are derived from the probe-tone method. The template that provides the best match to the musical input is used to assign a key to the input. Variations of this algorithm have also been proposed; they maintain its general character but include technical modifications to improve performance even further (Albrecht & Shanahan, 2013; Shmulevich & Yli-Harja, 2000; Temperley, 1999, 2007). The model also has status as a cognitive theory of key-finding with support from psychological experiments (Frankland & Cohen, 1996; Schmuckler & Tomovski, 2005).

Other key-finding theories also exist. Some propose that listeners detect the presence of rare musical intervals like the tritone (Brown & Butler, 1981; Browne, 1981; Butler, 1989). Others describe algorithms that match musical inputs to particular musical patterns, such as specific key-implying pitch sequences (Handelman & Sigler, 2013; Holtzman, 1977), positions in a geometric space defined by pitch-class coordinates (Albrecht & Shanahan, 2013), or pitch-classes that belong to particular major or minor scales (Longuet-Higgins & Steedman, 1971; Vos & Van Geenen, 1996). This chapter explores still other approaches to key-finding: algorithms based upon artificial neural networks.

5.1.2 One Network for Key-Finding

A key-finding procedure must accomplish two different tasks. First, when presented a musical stimulus, it must judge the tonic of the stimulus’s musical key. Second, it must judge the mode (major vs. minor) of the key. The perceptron described in Chapter 3 identifies the tonic of a presented musical scale but does not identify mode. The multilayer perceptron described in Chapter 4 identifies the mode of a presented musical scale but does not identify tonic. Thus, neither network is a candidate for a key-finding algorithm.

One could merge these two networks together, unaltered, to create a system capable of identifying both tonic and mode. This merging is possible because both
networks use identical input representations and are trained on identical patterns; but the tonic judgment of the perceptron is accomplished by a set of connection weights and output units that are completely independent of the connection weights, hidden units, and output unit of the multilayer perceptron.

However, it is not typical for connectionist researchers to train different networks on task components, and then sew them together into a working whole like Frankenstein’s monster. Instead, a more typical approach is to train a single artificial neural network to accomplish the entire task at once. This chapter explores two such networks for key-finding.

5.2 Key-Finding with Multilayered Perceptrons

Figure 5-1  A multilayer perceptron that uses four hidden units to detect both the mode and the tonic of presented major or harmonic minor scales.
To begin our exploration of key-finding with artificial neural networks, let us continue to train a network to generate responses when using musical scales as stimuli (i.e., the sets of pitch-classes from Table 3-1). In Chapter 3, a perceptron learned to output the tonic for each of these stimuli, while in Chapter 4 a multilayer perceptron learned to output the mode. The first network to be described in the current chapter is a multilayer perceptron (Figure 5-1) that learns to respond with both the mode and the tonic of each of these scales.

5.2.1 Task

The goal is to train a single artificial neural network to identify both the tonic and the mode when presented either a major scale or a harmonic minor scale. Given a main result of Chapter 3—that a multilayer perceptron was required to detect scale mode—the network of interest in this section uses a layer of hidden units to solve this problem.

5.2.2 Network Architecture

Figure 5-1 illustrates a multilayer perceptron for the scale-based version of the key-finding problem. It uses 13 output units to represent scale mode and tonic and uses 12 input units to represent the pitch-classes that belong either to a major or to a harmonic minor scale. Pilot simulations revealed that this network requires four hidden units to solve this problem. All of these hidden units, and all of the output units, are value units.

5.2.3 Training Set

The training set consists of 12 different major and 12 different harmonic scales represented in pitch-class format (i.e., each input stimulus is one of the rows of numbers presented in Chapter 3 as Table 3-1). The representation of inputs is identical to that described earlier in Chapters 3 and 4. The representation of outputs uses 13 value units to combine the representations used by the Chapter 3 perceptron and by the Chapter 4 multilayer perceptron. One output unit is trained to turn on if a presented scale is major, and to turn off otherwise. The remaining output units each represent a possible scale tonic; the network is trained to turn the unit representing the correct tonic on and to turn the other tonic units off.

5.2.4 Training

The network is trained with the generalized delta rule developed for networks of value units (Dawson & Schopflocher, 1992) using the Rumelhart software program.
(Dawson, 2005). All connection weights in the network are set to random values between −0.1 and 0.1 before training begins. The µs of the output and hidden units are all initialized to zero, but are modified as training proceeds. A learning rate of 0.01 is employed. Training occurs until the network generates a “hit” for every output unit for each of the patterns in the training set. Again, a “hit” is defined as activity of 0.9 or higher when the desired response is one or as activity of 0.1 or lower when the desired response is zero. The multilayer perceptron described in more detail in the next section learns to solve the problem after 6801 epochs of training.

5.3 Interpreting the Network

5.3.1 Carving Hidden Unit Space

In general, multilayer perceptrons solve classification problems by using output units to carve a hidden unit space into various decision regions. All of the output units in the Figure 5-1 network are value units. A value unit carves a hidden unit space into three areas using two parallel straight cuts. Patterns that fall between the two cuts (which are very close together) turn the unit “on,” and patterns that fall outside the cuts turn the unit “off.” In order for the multilayer perceptron to solve the key-finding problem, its hidden units must arrange patterns in a hidden unit space so that the output unit for mode can separate any major scale from all of the harmonic minor scales. This arrangement must also permit a tonic output unit to separate the two scales that have its tonic as a root from all of the other scales.

It is easy to imagine hypothetical hidden unit spaces that permit the output units to operate in this fashion. Figure 5-2 illustrates a hypothetical two-dimensional space. It arranges the scales in order by root, so that scales that have the same root are side by side. It also arranges the scales by type, so that all the major scales line up on the left, and all the harmonic minor scales line up on the right. As illustrated in Figure 5-2, this arrangement can easily be carved to identify C major by having the mode output unit “carve out” the major scales vertically, and by having the tonic output unit for C carve the two C scales out horizontally.

In order to solve the problem with the hypothetical space illustrated in Figure 5-2, network training must accomplish two things. First, the connection weights between the input units and the hidden units have to take on values that permit the hidden units to position the scales in appropriate locations in the space. Second, the connection weights between the hidden units and the output units have to take on values that permit the output units to carve the space appropriately.
A two-dimensional space (e.g., the one illustrated in Figure 5-2) does not provide the only possible solution for the key-finding problem. The fact that four hidden value units are required to permit the multilayer perceptron of Figure 5-1 to converge to a solution indicates that it needs a four-dimensional hidden unit space to classify the different scales. Let us take a moment to consider the properties of the four-dimensional hidden unit space obtained from the trained multilayer perceptron. This hidden unit space contains 24 different points, one for each stimulus scale. The four-dimensional coordinates of each point are the four activities produced in the network’s hidden units by a particular stimulus.

The problem with a four-dimensional hidden unit space is that it is impossible to visualize. However, we can understand a portion of its structure by considering lower-dimensional visualizations. For example, Figure 5-3 presents a three-dimensional visualization of part of this hidden unit space by plotting the positions of scales in a cube using activities of Hidden Units 1, 2, and 4 to provide the coordinates of each scale in the space.

In the discussion of Figure 5-2, I observed that it was necessary to arrange the major scales so that they are separable from all of the harmonic minor scales. Figure 5-3 indicates that the hidden unit space for the multilayer perceptron achieves this
goal. Note that all of the major scales arrange themselves in a group near the front of the cube. The dashed lines suggest where this cube could be carved to separate these scales, which turn the output unit on, from all of the harmonic minor scales that are spread elsewhere in the space and which turn the unit off.

Figure 5-3  A three-dimensional projection of the four-dimensional hidden unit space for the key-finding multilayer perceptron.

In the earlier discussion of Figure 5-2, it was noted that a major scale and a harmonic minor scale with the same root must be positioned in the hidden unit space so that a single unit (the output unit for their tonic) could separate both from all of the other scales. Figure 5-4 presents a second three-dimensional projection of the four-dimensional hidden unit space that attempts to show how it arranges different scales based upon the same root. It uses Hidden Unit 1, 2, and 3 activities to provide the coordinates of each scale in the cube. An examination of Figure 5-4 indicates that different scales with the same tonic are aligned in the cube in such a way that they can be separated from the others by the two parallel cuts of a value unit. The dashed lines in Figure 5-4 provide a sense of how this might be accomplished for the tonic F♯. Importantly, there is no requirement that one tonic output unit in the multilayer perceptron carve the hidden unit space in a fashion that is related to the
carving made by other output units (e.g., making cuts at different positions but in the same direction). Thus from the perspective illustrated in Figure 5-4 there is no compelling arrangement of all of the scales (e.g., around a circle of minor seconds). All that is required in this space is that one hidden unit can separate two locations (the locations of the major and minor scale with the same tonic) from all of the others.

**Figure 5-4** A different three-dimensional projection of the four-dimensional hidden unit space for the key-finding multilayer perceptron.

### 5.4 Coarse Codes for Key-Finding

The foregoing discussion emphasized hidden unit space regularities. In particular, it focused on the arrangement of the different scales in a four-dimensional hidden unit space in order to permit output units to isolate particular scales for generating correct responses for the key-finding task. We now turn to a related topic: What musical features does each of the four hidden units detect?

Some of the seminal advances in neuroscience were made possible by presenting stimuli to the sensory systems of animals while recording responses from individual neurons (Hubel & Wiesel, 1959; Lettvin, Maturana, McCulloch, & Pitts, 1959). With this technique, it is possible to describe a neuron as being sensitive to a trigger feature, a particular pattern which, when presented, produces maximum activity...
in the cell. This success, in turn, led some researchers to propose a neuron doctrine for research on perception (Barlow, 1972, 1995). Adherents to the neuron doctrine believed that a complete theory of perception would result from knowing the trigger features of each perceptual neuron. One can attempt to apply the neuron doctrine to the interpretation of artificial neural networks by identifying those stimuli that produce maximum activity in each of a network’s hidden units (Dawson, 2004, 2013).

In order to explore the trigger features of the four hidden units of Figure 5-1, one can wiretap them by recording the activity produced in each hidden unit by each pattern in the training set. Given the nature of the key-finding problem, one might expect that some hidden units are dedicated to detecting properties related to scale tonics, while others specialize in detecting properties related to scale modes.

Figure 5-5 presents the wiretapping results. Each panel of the figure presents the results for one hidden unit. The length of each horizontal bar provides the activity produced by one stimulus. The stimuli are grouped by tonic, with the major scale on the left and the minor scale on the right. Note that activity ranges from zero to one in both directions across the figure.

Figure 5-5 indicates that the hypothesis that the four different hidden units are specialized is mistaken. There is no indication that some hidden units detect scale mode and that others detect scale tonic. Instead, all four hidden units respond to a subset of tonics, and generate strong responses to a subset of scales that includes both majors and harmonic minors. In other words, the wiretapping does not reveal any illuminating trigger features.

Consider Hidden Unit 1, the upper left panel of Figure 5-5. It prefers harmonic minor scales, because there seems to be more black displayed by the bars on the right than there is grey displayed by bars on the left. However, it does not respond to every minor scale: it produces no activity to Bm, and produces minimal activity to A♯m, C♯m, and F♯m. Furthermore, it generates strong activity to some of the major scales (D♯, G, and G♯). In short, if this hidden unit is a scale mode specialist, then it is not a very accurate one.

It is also the case that Hidden Unit 1 does not appear to be successful in the specialized role of tonic detector. It generates strong responses to four tonics (C, D♯, G, and G♯) regardless of scale mode and generates very weak responses to several other tonics (B, C♯, and F♯). In some instances (F) it responds to the major scale with this tonic, but hardly responds at all to the harmonic minor scale with the same tonic.

An examination of Figure 5-5 indicates that similar stories are true of the other three hidden units as well. They tend to have a general preference for one scale mode or the other, but can generate strong responses to either. They tend to have high
activations to some tonics and not others, but sometimes activate to a tonic in one scale mode but fail to activate to the same tonic in a different scale mode. Individually, none of the hidden units seems eminently useful for solving the key-finding problem. However, collectively they are all important, because the multilayer perceptron uses these hidden units to converge to a solution. How is it possible for hidden units to be individually poor at representing a problem’s solution, but collectively successful?

Figure 5-5 The results of wiretapping each hidden unit, using the 24 input patterns as stimuli.
Figure 5-5 suggests one answer to this question. Let us consider the four hidden units in terms of tonic identification. Imagine a scale stimulus that produces little activity in Hidden Unit 1, moderate activity in Hidden Units 2 and 3, and high activity in Hidden Unit 4. What is the tonic and mode of this scale? For Hidden Unit 1, several scales produce activity of 0.1 or lower, such as: C♯, Bm, F♯, B, F, A♯, and F♯m. For Hidden Unit 2, a different subset of stimuli generates moderate activity between 0.29 and 0.61: Am, C♯m, Gm, E, F♯m, A♯, and B. Activity in this same middle range for Hidden Unit 3 is produced by yet another subset of patterns: F, F♯m, F♯, Bm, and D♯. Finally, high activity (0.75 or greater) is produced in Hidden Unit 4 by F♯m, Dm, C♯m, Am, G♯m, F♯, Cm, and D.

What do all of these four different subsets have in common? One can take this question literally by examining the four different subsets of scales provided in the preceding paragraph: [C♯, Bm, F♯, B, F, A♯, F♯m], [Am, C♯m, Gm, E, F♯m, A♯, B], [F, F♯m, F♯, Bm, D♯], and [F♯m, Dm, C♯m, Am, G♯m, F♯, Cm, D]. To ask, “what do these sets have in common?” is to ask, “what is the intersection of the four subsets taken together?” By examining the four subsets, we discover that there is only one scale present in all four: F♯m. In other words, a stimulus that produces low activity in Hidden Unit 1, medium activity in Hidden Units 2 and 3, and high activity in Hidden Unit 4 must be a scale that has F♯ as its tonic and is of minor mode.

This account of how the four inaccurate hidden units can successfully isolate a single scale is an example of coarse coding. Coarse coding is an example of a distributed representation, which is one of the more interesting and important contributions of connectionism to cognitive science (Hinton, McClelland, & Rumelhart, 1986; Van Gelder, 1991). Coarse coding involves individual processors that are inaccurate detectors of some property or feature. For instance, a hidden unit might be inaccurate because it is broadly tuned, and can be activated either by a wide range of features or by wide range of levels of a specific feature (Churchland & Sejnowski, 1992; Hinton et al., 1986). Alone, such processors are not adept at solving classification problems.

However, if one combines the responses of many such poor detectors, then overall accuracy can be markedly enhanced. Typically, this requires that these poor detectors each have a slightly different view of the problem. That is, in coarse coding each processor is expected to evaluate different but overlapping ranges of stimuli. Each processor’s response will be a combination of signal (the correct answer) and noise (additional incorrect responses). Processors with different perspectives on the problem will capture the same signal but will also likely capture different noise. When responses are pooled together, the different patterns of noise will cancel each
other out, leaving only the correct answer. The intersection-of-subsets example above illustrates this general principle.

5.4.1 Implications

Our discussion of Figure 5-5 indicates that the hidden units transform the input pattern space via coarse coding. Discovering coarse coding in a network causes difficulties that were absent from the network interpretations we saw in previous chapters. A consequence of coarse coding is that it may be difficult, if not impossible, to relate network structure to formal music theory. It is very difficult to simply examine the patterns of bars in Figure 5-5 and easily discover musical structure. The reason for this is that coarse codes seem to be the antithesis of formal musical entities: they are fuzzy, messy, noisy, and poorly defined. Coarse coding is in fact one property of artificial neural networks that connectionists use to champion connectionist cognitive science as an alternative to classical cognitive science.

In the 1950s, classical cognitive science arose as a reaction against radical behaviourism (Bruner, 1990; Miller, 2003; Sperry, 1993). Behaviourism emphasized the study of the environment and of observable behaviour. In contrast, cognitivism argued that full accounts of psychological phenomena must appeal to mental representations and to the processes that manipulated them. The central claim of classical cognitive science is that cognition is computation (Pylyshyn, 1984).

Classical cognitive science has a particular view of computation: it is the kind of activity performed by a physical symbol system (Newell, 1980), a device like a modern digital computer. As a result, the mind is assumed to hold symbolic expressions that can be manipulated by the formal rules of a programming language called the language of thought (Fodor, 1975). Classical cognitive science is the current form of the logicist tradition (Boole, 1854/2003) that equated the laws of thought with the laws of formal logic. Given this view, it is far from surprising that a deeply formal system—music—has been extensively studied by cognitivists (Berkowitz, 2010; Deutsch, 1999; Howell et al., 1985; Krumhansl, 1990a; Sloboda, 1985; Temperley, 2001).

However, not all cognitive scientists agree that thinking is the rule-governed manipulation of mental representations (Dawson, 2013). While classical cognitive science believes that cognition is formal or symbolic, connectionist cognitive scientists disagree. For example, Paul Smolensky has argued that, in contrast to symbolic theories, artificial neural networks are subsymbolic (Smolensky, 1988; Smolensky & Legendre, 2006).
To say that a network is subsymbolic is to say that its individual hidden units do not detect interpretable features that could be represented as individual symbols. Instead, hidden units detect microfeatures. Individually, a microfeature is unintelligible, because its “interpretation” depends crucially upon its context (i.e., the set of other microfeatures that are simultaneously present [Clark, 1993]). However, a collection of microfeatures represented by a number of different hidden units can represent a concept that corresponds to a symbol in a classical model. In other words, the symbolic vocabulary of classical cognitive science is an approximate description of what emerges from finer-level mechanisms involving microfeatures, processor activities, and the like.

Clearly, coarse coding relates to Smolensky’s notions of subsymbolic processes and microfeatures. However, if our analysis reveals coarse coding, then this likely means that we are discovering subsymbolic features that are much more difficult to relate to traditional music theory. This is a problem if our goal is to use network structure to provide insights about the theory of music, as we have been attempting in Chapters 3 and 4.

This is not to say that the coarse coding revealed in Figure 5-5 is not musical. The coarse coding of the hidden units supports musical classifications, and there are interesting musical properties revealed by the hidden unit space. For instance, an investigation of the distances between scales in the four-dimensional space (Figures 5-3 and 5-4) will reveal that different scales based upon the same tonic are near one another in this space. This is similar to the principle of placing a major key near its parallel minor in a Tonnetz (Schoenberg, 1969).

However, at times researchers might be tempted to emphasize the differences between subsymbolic musical networks and traditional music theory. That is, the extent to which the coarse codes of a network cannot be related to formal music may very well be the extent to which the network captures new regularities that cannot be expressed in formal theory. Importantly, the next chapter explores the possibility of an interesting compromise between hidden unit representations and formal music theory: we will explore networks that clearly exploit coarse coding, but these coarse codes are also related to formal music theory.

Before we move on to that material, though, let us consider a second approach to key-finding with neural networks. In the remainder of this chapter, we explore how one might create simpler neural networks that are variants of theories like the Krumhansl-Schmuckler key-finding algorithm (Krumhansl, 1990a). The purpose of these networks is to determine the keys of musical stimuli that are more complex than the simple scales that have been the focus, up to this point, of Chapter 5.
5.5 Key-Finding with Perceptrons

5.5.1 Key-Finding with Tonal Hierarchies

A critical component of listening to music is identifying its musical key. Human listeners are able to make such judgments very rapidly (Butler, 1989). Not surprisingly there is considerable interest in proposing procedures for musical key-finding, both to contribute to theories of music perception and to develop computer algorithms for automatically asserting the keys of musical stimuli (Albrecht & Shanahan, 2013; Frankland & Cohen, 1996; Handelman & Sigler, 2013; Holtzman, 1977; Longuet-Higgins & Steedman, 1971; Sapp, 2005; Shmulevich & Yli-Harja, 2000; Temperley, 1999; Temperley & Marvin, 2008; Vos & Van Geenen, 1996). An important feature of these theories is that they are intended for general musical stimuli, not just for the musical scales that were presented to the multilayer perceptron discussed in the preceding sections of Chapter 5.

In Section 5.1.1, we noted that the tonal hierarchy is one of the major findings in the study of musical cognition (Krumhansl, 1979, 1990a; Krumhansl & Shepard, 1979). Tonal hierarchies provide the foundation for one influential theory of key-finding proposed by Krumhansl and Schmuckler (Krumhansl, 1990a). The theory begins by recognizing that there is a different tonal hierarchy associated with each musical key. It uses these tonal hierarchies to create a set of key profiles, one for each of the 12 major and for each of the 12 minor musical keys. In the Krumhansl/Schmuckler key-finding algorithm, a to-be-analyzed musical stimulus is also represented as a key profile. This is accomplished by determining the total duration of each pitch-class in the stimulus. That is, one tabulates the total number of beats in the stimulus that involve hearing the pitch-class A, the total number of beats of the pitch-class A♯, and so on. Once the stimulus is represented in this fashion, correlations are computed between the stimulus’s profile and each of the 24 standardized key profiles. The algorithm identifies the standardized profile that produces the highest correlation, and asserts that this is the key of the musical stimulus. Krumhansl (1990a) reports that this algorithm performs very well. It may also serve as a model of the cognitive processes involved when human listeners establish tonal centres (Frankland & Cohen, 1996; Schmuckler & Tomovski, 2005). For instance, Schmuckler and Tomovski used the algorithm to predict listeners’ experience of tonality for preludes by both Bach and Chopin.

Although influential and successful, the Krumhansl-Schmuckler algorithm is not problem-free. First, its performance is not perfect. For example, when examining the performance of the algorithm on a test set of 492 selections of classical
compositions, Albrecht and Shanahan (2013) note that this procedure is only 74.2% accurate. Second, the performance of the algorithm varies depending upon whether it is presented stimuli in major or in minor keys. In Albrecht and Shanahan’s test set, the Krumhansl-Schmuckler algorithm generated 69.0% accuracy for major key compositions, while it was 83.2% accurate in assigning minor keys.

Some researchers have investigated variations of the Krumhansl-Schmuckler algorithm in an attempt to improve key-finding performance. In general, these variations explore two different avenues. The first involves replacing the Krumhansl-Schmuckler tone profiles with alternative profiles derived from large corpuses of music pieces (Albrecht & Shanahan, 2013; Temperley, 2004). These alternative tone profiles are provided in Table 5-1 along with those used in the Krumhansl-Schmuckler algorithm.

Table 5-1 The three sets of key profiles used in key-finding algorithms.

<table>
<thead>
<tr>
<th>Scale degree of pitch-class</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Krumhansl-Schmuckler</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Major</td>
<td>6.35</td>
<td>2.23</td>
<td>3.48</td>
<td>2.33</td>
<td>4.38</td>
<td>4.09</td>
<td>2.52</td>
<td>5.19</td>
<td>2.39</td>
<td>3.66</td>
<td>2.29</td>
<td>2.88</td>
</tr>
<tr>
<td>Minor</td>
<td>6.33</td>
<td>2.68</td>
<td>3.52</td>
<td>5.38</td>
<td>2.60</td>
<td>3.53</td>
<td>2.54</td>
<td>4.75</td>
<td>2.98</td>
<td>2.69</td>
<td>3.34</td>
<td>3.17</td>
</tr>
<tr>
<td>Temperley</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Major</td>
<td>0.75</td>
<td>0.06</td>
<td>0.49</td>
<td>0.08</td>
<td>0.67</td>
<td>0.46</td>
<td>0.10</td>
<td>0.72</td>
<td>0.10</td>
<td>0.37</td>
<td>0.06</td>
<td>0.40</td>
</tr>
<tr>
<td>Minor</td>
<td>0.71</td>
<td>0.08</td>
<td>0.48</td>
<td>0.62</td>
<td>0.05</td>
<td>0.46</td>
<td>0.11</td>
<td>0.75</td>
<td>0.40</td>
<td>0.07</td>
<td>0.13</td>
<td>0.33</td>
</tr>
<tr>
<td>Albrecht-Shanahan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Major</td>
<td>0.24</td>
<td>0.01</td>
<td>0.11</td>
<td>0.01</td>
<td>0.14</td>
<td>0.09</td>
<td>0.02</td>
<td>0.21</td>
<td>0.01</td>
<td>0.08</td>
<td>0.01</td>
<td>0.08</td>
</tr>
<tr>
<td>Minor</td>
<td>0.22</td>
<td>0.01</td>
<td>0.10</td>
<td>0.12</td>
<td>0.02</td>
<td>0.10</td>
<td>0.01</td>
<td>0.21</td>
<td>0.06</td>
<td>0.02</td>
<td>0.06</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Note. The major and minor profiles from Krumhansl (1990a), the major and minor profiles from Temperley (2004), and the major and minor profiles from Albrecht and Shanahan (2013). Scale degree 6 is assumed to be the tonic pitch, etc.

The second avenue for exploring variations of the Krumhansl-Schmuckler algorithm involves comparing inputs to standardized key profiles using some method other than correlation. Temperley (2004) uses the Bayesian probability equation, while
Albrecht and Shanahan (2013) assume that the standardized key profiles, and the to-be-classified stimulus, are all points located in 12-dimensional space; they then use the distance between pairs of points in this space to determine musical key.

### 5.5.2 A Key-Finding Perceptron

Let us now explore another variation of algorithms that use key profiles for key-finding. In this new approach, an artificial neural network learns to map different key profiles to outputs that represent musical key. Then we present the network a variety of different musical stimuli in order to test its ability to assert their musical keys.

Pilot studies reveal that a perceptron can accomplish this task (Figure 5-6). This perceptron uses 12 input units to represent key profiles, and uses 24 output units. Each of these output units represents a different musical key. Unlike the networks discussed up to this point, these output units are also integration devices that use the sigmoid-shaped logistic activation function instead of the Gaussian. I adopt these properties for two reasons. First, the output unit representation makes the behaviour of the network analogous to the behaviour of a correlation-based algorithm like the Krumhansl-Schmuckler. Second, using the logistic activation function permits us to interpret output unit activity as a probability such as the Bayesian probability employed by Temperley (1999).

![Figure 5-6](image)

**Figure 5-6** A perceptron that can be used to map key profiles onto musical key.
5.5.3 Perceptron Types

Three different types of key-finding perceptrons are trained. The first is trained on key profiles taken from the Krumhansl-Schmuckler algorithm (Krumhansl, 1990a), the second on key profiles taken from the Temperley (2004) algorithm, and the third on key profiles taken from the Albrecht and Shanahan (2013) algorithm.

Each of these three types of perceptrons represents variations of the original Krumhansl-Schmuckler correlational algorithm. Consider a perceptron trained on the Krumhansl-Schmuckler key profiles. It differs from the original Krumhansl-Schmuckler algorithm in two ways. First, it does not directly match key profiles to musical keys. Instead, it uses its connection weights to transform key profiles before their signals reach the output units. Second, this network does not use correlations to match stimuli with desired musical keys. Instead, the output units take the signals from (transformed) input profiles and then further transform them by applying the logistic equation. It is therefore possible that a perceptron trained on the Krumhansl-Schmuckler key profiles will respond differently to the same stimuli in comparison to the Krumhansl-Schmuckler algorithm itself.

5.5.4 Training Sets

One advantage that the Krumhansl-Schmuckler algorithm has via its use of correlations to compare stimuli to key profiles is that this operation is not affected by stimulus magnitude. That is, when a long musical selection is summarized as a pitch-class profile, the magnitude of each value in its 12-dimensional vector is expected to be larger than would be observed when a shorter musical selection is summarized in the same way. This is simply because on average one would expect to find more instances of each pitch-class in a longer piece than in a shorter one. However, the correlation equation is not sensitive to stimulus magnitude. Instead, it in essence computes the similarity of two patterns by only considering their relative directions (when they are considered as vectors pointing in a multidimensional space).

As perceptrons do not use correlations, their outputs can be affected by differences in stimulus magnitude. For this reason, I first mean-centre and then normalize each key profile from Table 5-1. The resulting normalized key profiles associated with each of these three different algorithms are provided in Table 5-2.
Table 5-2. The three sets of mean-centred normalized key profiles used to train different key-finding perceptrons. These are the same profiles that were presented earlier in Table 5-1, but each has been processed as described in the text.

<table>
<thead>
<tr>
<th>Scale degree of pitch-class</th>
<th>Krumhansl-Schmuckler</th>
<th>Temperley</th>
<th>Albrecht-Shanahan</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Major</td>
<td>Minor</td>
<td>Major</td>
</tr>
<tr>
<td>0</td>
<td>0.655</td>
<td>0.668</td>
<td>0.575</td>
</tr>
<tr>
<td>1</td>
<td>−0.286</td>
<td>−0.234</td>
<td>−0.330</td>
</tr>
<tr>
<td>2</td>
<td>−0.001</td>
<td>0.151</td>
<td>0.103</td>
</tr>
<tr>
<td>3</td>
<td>−0.263</td>
<td>−0.305</td>
<td>0.199</td>
</tr>
<tr>
<td>4</td>
<td>0.205</td>
<td>0.355</td>
<td>0.040</td>
</tr>
<tr>
<td>5</td>
<td>0.139</td>
<td>0.119</td>
<td>0.250</td>
</tr>
<tr>
<td>6</td>
<td>−0.220</td>
<td>−0.289</td>
<td>0.485</td>
</tr>
<tr>
<td>7</td>
<td>0.390</td>
<td>0.405</td>
<td>0.276</td>
</tr>
<tr>
<td>8</td>
<td>−0.250</td>
<td>−0.280</td>
<td>−0.012</td>
</tr>
<tr>
<td>9</td>
<td>0.041</td>
<td>0.014</td>
<td>−0.280</td>
</tr>
<tr>
<td>10</td>
<td>−0.273</td>
<td>−0.333</td>
<td>−0.037</td>
</tr>
<tr>
<td>11</td>
<td>−0.138</td>
<td>0.052</td>
<td>−0.009</td>
</tr>
</tbody>
</table>

To build a training set for a perceptron trained on the Krumhansl-Schmuckler profiles, the mean-centred normalized major key profile in the first row of Table 5-2 is used to generate a key profile for each of the 12 major keys by associating the value given in Table 5-2 with the appropriate input unit. That is, for the key of C major the value of 0.655 is presented to the input unit representing the pitch-class C, the value of −0.286 is presented to the input unit representing the pitch-class C♯, and so on. Similarly, the stimulus for the key of C♯ major involves presenting the value of 0.655 to the input unit representing the pitch-class C♯, the value of −0.286 to the input unit representing the pitch-class D, and so on. A similar procedure creates input stimuli for the 12 different minor keys using the normalized minor key profile in the second row of Table 5-2. As a result, the training set consists of 24 different input patterns, one for each musical key. A second set of 24 training patterns is created by applying this method with the two Temperley profiles from Table 5-2, and a third set of 24 training patterns is created by applying this method using the two Albrecht-Shanahan profiles from Table 5-2.
5.5.5 Training

Each network is a perceptron of the type illustrated in Figure 5-6, and is trained using the Rosenblatt software program (Dawson, 2005). Before training begins, all connection weights in a network are set to random values between −0.1 and 0.1. I initialize all the biases ($\theta$) of the output units to zero, but I then modify them using the learning rule during training. I employ a learning rate of 0.50. Networks are trained epoch by epoch, where each of the 24 training patterns is presented once each epoch. The order of pattern presentation is randomized each epoch. Training continues until the network generates a “hit” for every output unit for each of the 24 patterns in the training set. A hit is defined as an activity of 0.90 or higher when the desired output is one or as an activity of 0.10 or lower when the desired output is zero.

Each network is trained from a random configuration of a small initial set of connection weights, it is possible that different networks will achieve qualitatively different end states via training. For this reason, I train 10 different perceptrons on each of the three sets of training patterns, resulting in 30 different perceptrons. Each of these perceptrons is a different “subject” in a simulation experiment.

All 30 perceptrons successfully learned to map mean-centred normalized tone profiles onto musical keys. On average the 10 perceptrons trained on the Krumhansl-Schmuckler profiles converge after 1354.3 epochs of training ($SD = 0.823$). On average the 10 perceptrons trained on the Temperley profiles converge after 1453.4 epochs of training ($SD = 1.17$). On average the 10 perceptrons trained on the Albrecht-Shanahan profiles converge after 1864.0 epochs of training ($SD = 1.25$).

As 10 different versions of each perceptron were trained, and as each of these began from a different set of small randomly selected initial weights, are there any qualitative differences between the solutions reached by different versions of the same perceptron type? Interestingly, it appears that each perceptron trained on the same set of profiles reaches essentially the same solution (i.e., the same set of connection weights, as is detailed later) and generates essentially the same responses to stimuli.

5.5.6 Testing

After successfully completing the training phase, a network is tested on its ability to assert the musical key of 296 different musical selections. These test stimuli represent four different sources. The first is the collection of 48 preludes and 48 fugues from both books of J.S. Bach’s Well-Tempered Clavier. These compositions are a typical test bed for key-finding algorithms (Albrecht & Shanahan, 2013; Temperley,
The second is a collection of 24 preludes composed by Frederic Chopin as his Opus 28. One prelude was written in each musical key. These too are often used to test the accuracy of key-finding theories; for instance, they pose some difficulty for the Krumhansl-Schmuckler algorithm (Krumhansl, 1990a). The third test set is the 24 Preludes, Opus 67, composed by Johann Hummel. As is the case for the Chopin preludes, Hummel composed one prelude in each musical key. The fourth test set, the only selection of music not from the classical genre, consists of 152 Nova Scotia folk songs from *Songs and Ballads from Nova Scotia* (Creighton, 1932).

Each of these four collections of musical selections is available in the kern file format at http://kern.humdrum.org/. As a result, they can be analyzed using the Humdrum toolkit (Huron, 1999). Humdrum’s key command is used to represent each of the 296 test stimuli in the format required by the Krumhansl-Schmuckler algorithm. That is, the total duration of each of the 12 Western pitch-classes is computed for each stimulus, producing a 12-entry vector representation. Each of these vectors is then normalized to unit length prior to being presented to a trained network. This normalization renders the test stimuli into a format that is identical to the one used to represent the training stimuli. It also ensures that the varying lengths of each of these test patterns do not affect network performance.

During the test phase, network performance is assessed using a procedure analogous to the Krumhansl-Schmuckler algorithm. That is, when a test stimulus is presented to a network it produces activity in 24 different output units, each one representing a different musical key. The output unit that generates the highest activity is taken to indicate the musical key being asserted by the network. The accuracy of this assertion is then compared to the key in which the stimulus was actually composed (information that is provided as part of the test stimulus’s kern file).

### 5.5.7 Perceptron Performance

The first row of Table 5-3 provides the average accuracy of key assertion for perceptrons trained on the Krumhansl-Schmuckler profiles for the four different sets of test materials. For each set, accuracy is given as the percent correct for the entire set of stimuli; accuracy is then provided for only the major key stimuli and for only the minor key stimuli. The next two rows of Table 5-3 then provide the performance for perceptrons trained with the Temperley profiles and for perceptrons trained with the Albrecht-Shanahan profiles.

In order to have some reference point for assessing perceptron performance, the Krumhansl-Schmuckler algorithm is also employed. The fourth row of Table 5-3 provides key-finding accuracy when correlations between test stimuli and the
normalized Krumhansl-Schmuckler profiles are used (where key is asserted by finding the highest correlation).

Table 5-3: The average percent accuracy of classification of the three perceptrons trained on three different mean-centred and normalized key profiles.

<table>
<thead>
<tr>
<th>Key-finding method</th>
<th>Bach Well-Tempered Clavier</th>
<th>Chopin preludes</th>
<th>Hummel preludes</th>
<th>Nova Scotia folk songs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All</td>
<td>Major</td>
<td>Minor</td>
<td>All</td>
</tr>
<tr>
<td>KSP</td>
<td>89.6</td>
<td>89.6</td>
<td>89.6</td>
<td>54.2</td>
</tr>
<tr>
<td>TP</td>
<td>95.8</td>
<td>97.9</td>
<td>93.8</td>
<td>95.8</td>
</tr>
<tr>
<td>ASP</td>
<td>96.9</td>
<td>93.8</td>
<td>100.0</td>
<td>83.3</td>
</tr>
<tr>
<td>KS Corr</td>
<td>93.8</td>
<td>87.5</td>
<td>100.0</td>
<td>87.5</td>
</tr>
</tbody>
</table>

Note. KSP indicates the perceptron trained on the Krumhansl-Schmuckler profiles; TP indicates the perceptron trained on the Temperley profiles, and ASP indicates the perceptron trained on the Albrecht-Shanahan profiles. The final row (KS Corr) provides the performance of the Krumhansl-Schmuckler correlation algorithm for purposes of comparison.

One observation to make from Table 5-3 is that perceptrons trained on tone profiles demonstrate different key-finding performance for stimuli in major or minor keys. In some instances, a perceptron is better on minors than on majors, while in other instances the reverse is true. Interestingly, the perceptron that performs best on minor key stimuli is the one trained on the Albrecht-Shanahan profiles; for classical genre patterns, it is 100% accurate. One of the motivations that Albrecht and Shanahan (2013) provided for their profiles was the goal of improving key-finding for minor key stimuli.

Another observation to make from Table 5-3 is that performance on classical genre stimuli is much better—for both perceptrons and for the correlation algorithm—than it is for the Nova Scotia folk songs. This may be due to a variety of factors. For instance, the folk songs are generally short univocal selections, while the classical pieces are generally longer and include harmony. As a result, there may be more reliable information about key in the classical selections than in the folk songs. Table 5-3 indicates that particular sets of tone profiles for key-finding might have more success for some genres, or for at least some subsets of stimuli, than for others.
One final observation to make concerning Table 5-3 is that when the same set of profiles is used, but processed differently, the same performance is not produced. In particular, the Krumhansl-Schmuckler perceptron generates significantly different levels of accuracy than the Krumhansl-Schmuckler correlation algorithm, even though both of these methods use the same profiles. This shows that the perceptron is processing the profiles in a different fashion than when correlation is used.

The results of this study indicate that a very simple artificial neural network, the perceptron, is capable of mapping key profiles onto musical keys, and can then use this learned ability to perform respectably when given the task of asserting the keys of novel stimuli. In addition, it is clear that the outputs of a perceptron trained on key profiles are not identical to the judgments of an algorithm that asserts musical key using correlations with the same key profiles. This difference provides further support for the position that key-finding performance can be altered by changing the method used to compare the key profile of a stimulus with the key profiles associated with different musical keys. Finally, perceptron performance is clearly affected by which key profiles are used for training. Being trained on the normalized Temperley profiles leads to better performance when presented novel stimuli than being trained on the normalized Krumhansl-Schmuckler profiles.

5.6 Network Interpretation

5.6.1 Interpreting Perceptron Weights

The key-finding perceptrons that I have been discussing differ from the previous networks in the use of the logistic activation function in their output units. Earlier I noted that, in general, networks that use value units are easier to interpret than networks that use integration devices. However, when the network has no hidden units, a quantitative analysis of an integration device network’s structure can be carried out very easily. This is because a perceptron that uses integration devices as outputs is functionally equivalent to a system that performs logistic regression (Schumacher, Rossner, & Vach, 1996). This, in turn, means that a network’s weights can be interpreted as being natural logarithms of odds ratios, and that the size of each weight provides a measure of the importance of each input with respect to activating an output unit (Dawson & Gupta, 2017). When a perceptron uses integration devices as outputs, its connection weights literally reflect the effect size of each input signal—the degree to which each part of a stimulus is responsible for turning an input unit on or off.
In Chapter 3, instead of viewing input units in terms of the pitch-class they represented, I considered them in terms of their degree within a scale related to an output unit (coding the input unit representing the scale’s tonic pitch-class as 0, coding the input unit representing the pitch-class a semitone higher than the tonic as 1, and so on). This is how I transformed Table 3-2 into Table 3-3 when interpreting the structure of the scale tonic perceptron. I found that, with this recoding, each output unit had essentially the same set of connection weights feeding into it.

This is also the case when this recoding is performed for each of the three key-finding perceptrons. That is, for each perceptron, each of its output units that represents a major key has the same set of input weights feeding into it; a different pattern of weights is found for each of the network’s output units that represents a minor key. Table 5-4 provides the connection weights that I discovered for each perceptron, and for each key type, when input units were considered in terms of a pitch’s relation to the key (with respect to degree) instead of absolute pitch-class.

Table 5-4  Weights from input units to typical output units of the three perceptrons.

<table>
<thead>
<tr>
<th></th>
<th>Weight between input unit (coded in terms of key degree) and output unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Krumhansl-Schmuckler</td>
<td></td>
</tr>
<tr>
<td>Major</td>
<td>6.68</td>
</tr>
<tr>
<td>Minor</td>
<td>6.46</td>
</tr>
<tr>
<td>Temperley</td>
<td></td>
</tr>
<tr>
<td>Major</td>
<td>3.67</td>
</tr>
<tr>
<td>Minor</td>
<td>3.13</td>
</tr>
<tr>
<td>Albrecht-Shanahan</td>
<td></td>
</tr>
<tr>
<td>Major</td>
<td>7.27</td>
</tr>
<tr>
<td>Minor</td>
<td>5.73</td>
</tr>
</tbody>
</table>

Note. Input units are coded in terms of key degree; biases are provided in the column labelled θ. ASP indicates the perceptron trained on the Albrecht-Shanahan profiles, TP indicates the perceptron trained on the Temperley profiles, and KSP indicates the perceptron trained on the Krumhansl-Schmuckler profiles. Unit type indicates whether the output unit represents a major key or a minor key.
In all of the key-finding algorithms that use key profiles (Albrecht & Shanahan, 2013; Krumhansl, 1990a; Temperley, 2004), there is a direct comparison of the profile representing a musical stimulus to the profile of each musical key. That is, the tacit assumption underlying these algorithms is that the key profile itself (Table 5-1) provides enough information to identify key. If the different values of a key profile were themselves sufficient to select the correct key, then a perceptron would not have to weight these input values when learning to map these profiles into musical keys. That is, each of its connection weights would equal 1, because the perceptron would not have to alter its stimulus information. However, Table 5-4 reveals that this is clearly not the case: Table 5-4 contains connection weights ranging from over 7 to about −6. This indicates that different components of stimulus profile have different degrees of importance in determining musical key.

It is important to realize that these Table 5-4 weightings of input unit signals are in addition to the different-sized signals that are reflected in the profiles themselves. For instance, the mean-centred normalized Albrecht-Shanahan major key profile itself indicates that the most common component in the profile is degree 0, because its profile value is 0.575. However, the perceptron further amplifies this value by multiplying it by 7.29, indicating that this high profile value is itself extremely informative relative to the other profile values. Similarly, in the Albrecht-Shanahan major profile the values of degree 1 and degree 3 both equal −0.29. However, the perceptron weights these two values quite differently, multiplying the degree 1 signal by a weight of 0.44, while multiplying the degree 3 signal by a weight of −5.04. This indicates that the perceptron has learned that decreased occurrence of degree 3 pitch-classes is far more important for establishing the major key than is decreased occurrence of degree 1 pitch-classes. Similar observations can be made for the various signals contributing to the identification of either major or minor keys, and these observations can be made for all three types of perceptron.

5.6.2 Implications

Table 5-4 indicates that when key profiles are used to train a key-finding perceptron, it learns that not all components of the profile are equally important. Perceptrons adjust their weights to modify the input signals to emphasize the information provided by important profile elements, and to de-emphasize the information provided by less important elements. If this were not the case, then all of the weights in Table 5-4 would equal 1. One interesting implication of this finding is that the structure of a trained perceptron suggests possible variations of other key-finding algorithms that do not employ artificial neural networks.
For example, the Albrecht-Shanahan algorithm positions each tone profile as a point in a 12-dimensional space, positions a stimulus profile in the same space, and assigns the stimulus the key that is represented by the point the stimulus is closest to (Albrecht & Shanahan, 2013). However, this model assumes that each dimension in the space has the same importance. The Albrecht-Shanahan perceptron weights (Table 5-4) indicate that the space in which the Albrecht and Shanahan algorithm measures distances could be distorted, with some dimensions being stretched out (i.e., those associated with important profile components), and with others being shortened in size (i.e., those associated with less important profile components). The weights of the perceptron provide magnitudes for distorting the Albrecht-Shanahan space before points are plotted and distances are measured; it would be interesting to see what effect such distortions would have on the algorithm’s performance. A similar approach could be taken to incorporate weights into other algorithms that are based on tone profiles by using them to scale profile components. Of course, the weights in Table 5-4 would likely not be the precise ones to use, because they arise from networks trained on mean-centred normalized tone profiles. However, a perceptron trained to map musical keys to tone profiles that have not been preprocessed (i.e., those presented earlier in Table 5-2) could provide weights that could be incorporated into another algorithm.

5.7 Summary and Implications

5.7.1 Summary

This chapter examined key-finding from two different perspectives. I began by exploring a multilayer perceptron for identifying the tonic and mode of scales. I then discovered that it developed a different kind of representation, a coarse code, for determining scale tonics and modes.

For the second perspective, we trained a perceptron to key-find using key profiles taken from established theories of key-finding. After this training, I tested the perceptron using a variety of different compositions after summarizing them in the way required by the original Krumhansl-Schmuckler algorithm. In general, perceptrons trained on key profiles are plausible algorithms for asserting musical key. Furthermore, an examination of the weights of these trained networks indicates that some components of a key profile are more important than others with respect to performing key-finding. Let us now consider some of the implications of the results observed in this chapter.
5.7.2 Coarse Coding

One of the major themes of this book is that artificial neural networks provide a medium for discovering new mappings between inputs and outputs. This theme was illustrated in our interpretations of a tonic-identifying perceptron in Chapter 3, and in our interpretations of a mode-identifying multilayer perceptron in Chapter 4. The first half of Chapter 5 demonstrated that when a multilayer perceptron learns to accomplish both of these tasks at the same time it discovers a new method that utilizes coarse coding.

The discovery of coarse coding has some interesting implications for the study of musical cognition. The general perspective of this field of study is that mental representations are used to organize and understand musical stimuli. However, the nature of these mental representations is an open question. One possibility is that geometric or spatial representations used to model musical regularities might also serve as accounts of mental musical representations (Krumhansl, 1990a, 2005).

Coarse coding provides an alternative representational possibility. Networks that employ coarse coding should exhibit particular patterns of learning (i.e., learning some instances faster than they learn others), should make particular patterns of responses to novel stimuli, and should generate definite errors when presented noisy stimuli. One might ask whether human listeners use coarse codes to represent music, and search for an answer to this question by comparing patterns of human responses under various conditions to related patterns observed in networks that are known to coarse code.

Coarse coding has less obvious, but still definite, implications for formal music theory. Our interpretation of Figure 5-5 focused on a formal process (intersections of subsets) that could use the coarse code to identify a scale's tonic and mode. This formal process is carried out when an output unit's activation function carves decision regions in a hidden unit space.

Our interpretation of each hidden unit's response (Figure 5-5) was cursory, and did not explore particular musical features. I simply noted that the patterns of activity depicted in that figure summarize "subsymbolic" properties. However, such properties, though subsymbolic, are almost certainly going to be musical in nature. A more intensive (and likely more challenging) examination of hidden unit responses could identify the musical properties that cause a hidden unit to generate high responses to some stimuli but not to others. These properties would likely represent an alien music theory. They would also be important for informing representational theories of music cognition.
5.7.3 Asserting Musical Key

One of the important results of the second half of Chapter 5 is that perceptrons trained on key profiles from other theories can perform quite well when identifying the keys of novel stimuli. These results are very encouraging, because they indicate that artificial neural networks demonstrate a great deal of promise as key-finding mechanisms.

These results are also surprising. First, the fact that simple perceptrons are capable of this high degree of performance is unexpected. Second, this high degree of performance is produced when a very sparse representation of input stimuli is employed. For instance, given the various modulations of key that characterize the pieces that make up the Well-Tempered Clavier (Bruhn, 2014; Temperley, 1999), it is surprising that a 12-number summary of each piece permits a perceptron to judge their intended key (as per Bach’s titling) at nearly 98% accuracy. If anything, the success of these simple networks provides converging validation about the fundamental importance of Krumhansl’s (1990a) tonal hierarchy.

The three different perceptrons also point to even further avenues of exploration. First, the three sets of tone profiles (Albrecht & Shanahan, 2013; Krumhansl, 1990a; Temperley, 1999) used to train the perceptrons are derived from different origins, are typically employed using different algorithms, and involve different magnitudes of values (as can be seen by inspecting Table 5-1). Given these differences, it is perhaps not surprising that no one has attempted to improve key-finding by combining all three sets of profiles together in a single algorithm.

However, from the perspective of artificial neural networks, combining different profiles together is a natural next step. Indeed this notion follows naturally from the notion of coarse coding, if we consider coarse coding not in terms of hidden units but instead in terms of perceptron responses. Table 5-3 indicates that the responses of different types of perceptrons differ from one another on the various test stimuli. This suggests that each captures slightly different properties. From the coarse coding perspective of neural networks, this in turn suggests that key-finding performance could be improved in an algorithm that combines the three different sets of tone profiles together.

There is no reason that a single perceptron cannot be trained to key-find by learning about all three different types of key profiles in a single training set. Second, and more in the spirit of coarse coding, one could assert musical key after combining the responses of different perceptrons together. Such an architecture is called a committee of networks; committees of networks have been shown to be superior to single networks in a variety of pattern classification tasks (Buus et al., 2003; Das,
Reddy, & Narayanan, 2001; Guo & Luh, 2004; Marwala, 2000; Medler & Dawson, 1994; Zhao, Huang, & Sun, 2004). It would be interesting to see how a committee of networks performs on the key-finding task in comparison to the perceptrons that have been described in this chapter.