In a (3,1) repetition code, only 2 of the possible 8 three-bit data blocks are codewords. We can represent these bit patterns geometrically with the axes being bit positions in the data block. In the left plot, the filled circles represent the codewords [0 0 0] and [1 1 1], the only possible codewords. The unflled ones correspond to the transmission. The center plot shows that the distance between codewords is 3.

Because distance corresponds to flipping a bit, calculating the Hamming distance geometrically means following the axes rather than going "as the crow flies". The right plot shows the datawords that result when one error occurs as the codeword goes through the channel. The three datawords are unit distance from the original codeword. Note that the received dataword groups do not overlap, which means the code can correct all single-bit errors.

To perform decoding when errors occur, we want to fnd the codeword (one of the flled circles in [Figure 6.22](#_bookmark462)) that has the highest probability of occurring: the one closest to the one received. Note that if a dataword lies a distance of 1 from two codewords, it is **impossible** to determine which codeword was actu ally sent. This criterion means that if any two codewords are two bits apart, then the code **cannot** correct the channel-induced error. **Thus, to have a code that can correct all single- bit errors, codewords must have a minimum separation of three. Our repetition code has this property.**

Introducing code bits increases the probability that any bit arrives in error (because bit interval durations decrease). However, using a well-designed error-correcting code corrects bit reception errors. Do we win or lose by using an error-correcting code? The answer is that we can win if the code is well-designed. The (3,1) repetition code demonstrates that we can lose ([Block Channel Coding (Page 304)](#_bookmark459)). To develop good channel coding, we need to develop frst a general framework for channel codes and discover what it takes for a code to be maximally efcient: Correct as many errors as possible using the fewest error correction bits as possible (making the K efciency



as large as possible.) We also need a systematic way of fnding the codeword closest to any received dataword. A much better code than our (3,1) repetition code is the following (7,4) code.



where the generator matrix is

