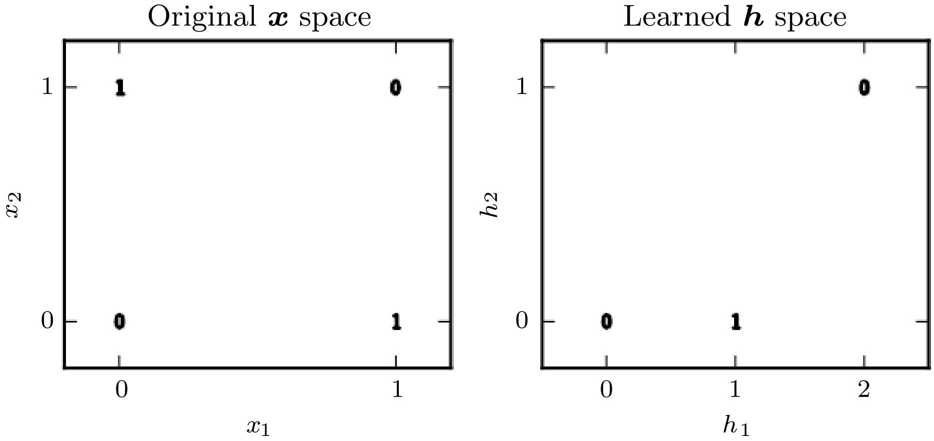

Figure 6.1: Solving the XOR problem by learning a representation. The bold numbers printed on the plot indicate the value that the learned function must output at each point. *(Left)*A linear model applied directly to the original input cannot implement the XOR function. When $x_1 = 0$, the model's output must increase as x_2 increases. When $x_1 = 1$, the model's output must decrease as x_2 increases. A linear model must apply a fixed coefficient w_2 to x_2. The linear model therefore cannot use the value of x_1 to change the coefficient on x_2 and cannot solve this problem. *(Right)*In the transformed space represented by the features extracted by a neural network, a linear model can now solve the problem. In our example solution, the two points that must have output 1 have been collapsed into a single point in feature space. In other words, the nonlinear features have mapped both $\boldsymbol{x} = [1, 0]^\top$ and $\boldsymbol{x} = [0, 1]^\top$ to a single point in feature space, $\boldsymbol{h} = [1, 0]^\top$. The linear model can now describe the function as increasing in h_1 and decreasing in h_2. In this example, the motivation for learning the feature space is only to make the model capacity greater so that it can fit the training set. In more realistic applications, learned representations can also help the model to generalize.