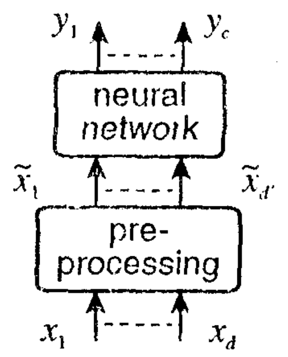

Figure 1.4. The majority of neural network applications require the original input variables $x_1, \ldots, x_d$ to be transformed by some form of pre-processing to give a new set of variables $\widetilde{x}_1, \ldots, \widetilde{x}_{d'}$. These are then treated as the inputs to the neural network, whose outputs are denoted by $y_1, \ldots, y_c$.

1.4 The curse of dimensionality

There is another important reason why pre-processing can have a profound effect on the performance of a pattern recognition system. To see this let us return again to the character recognition problem, where we saw that increasing the number of features from 1 to 2 could lead to an improvement in performance. This suggests that we might use an ever larger number of such features, or even dispense with feature extraction altogether and simply use all 65 536 pixel values directly as inputs to our neural network. In practice, however, we often find that, beyond a certain point, adding new features can actually lead to a *reduction* in the performance of the classification system. In order to understand this important effect, consider the following very simple technique (not recommended in practice) for modelling non-linear mappings from a set of input variables x_i to an output variable y on the basis of a set of training data.

We begin by dividing each of the input variables into a number of intervals, so that the value of a variable can be specified approximately by saying in which interval it lies. This leads to a division of the whole input space into a large number of boxes or cells as indicated in Figure 1.5. Each of the training examples corresponds to a point in one of the cells, and carries an associated value of the output variable y. If we are given a new point in the input space, we can determine a corresponding value for y by finding which cell the point falls in, and then returning the average value of y for all of the training points which lie in that cell. By increasing the number of divisions along each axis we could increase the precision with which the input variables can be specified. There is, however, a major problem. If each input variable is divided into M divisions, then the total number of cells is M^d and this grows *exponentially* with the dimensionality of the input space. Since each cell must contain at least one data point, this implies that the quantity of training data needed to specify the mapping also grows exponentially. This phenomenon has been termed the *curse of dimensionality*