In summary, there is a natural choice of both output-unit activation function and matching error function according to the type of problem being solved. For regression, we use linear outputs and a sum-of-squares error, for multiple independent binary classifications, we use logistic sigmoid outputs and a cross-entropy error function, and for multi-class classification, we use softmax outputs with the corresponding multi-class cross-entropy error function. For classification problems involving two classes, we can use a single logistic sigmoid output, or alternatively, we can use a network with two outputs having a softmax output activation function.

This procedure is quite general, and by considering other forms of conditional distribution, we can derive the associated error functions as the corresponding negative log likelihood. We will see an example of this in the next section when we consider multimodal network outputs.

6.5. Mixture Density Networks

So far in this chapter we have discussed neural networks whose outputs represent simple probability distributions comprising either a Gaussian for continuous variables or a binary distribution for discrete variables. We close the chapter by showing how a neural network can represent more general conditional probabilities by treating the outputs of the network as the parameters of a more complex distribution, in this case a Gaussian mixture model. This is known as a *mixture density network*, and we will see how to define the associated error function and the corresponding output-unit activation functions.

6.5.1 Robot kinematics example

The goal of supervised learning is to model a conditional distribution $p(\mathbf{t}|\mathbf{x})$, which for many simple regression problems is chosen to be Gaussian. However, practical machine learning problems can often have significantly non-Gaussian distributions. These can arise, for example, with *inverse problems* in which the distribution can be multimodal, in which case the Gaussian assumption can lead to very poor predictions.

Exercise 6.16

As a simple illustration of an inverse problem, consider the kinematics of a robot arm, as illustrated in Figure 6.16. The *forward problem* involves finding the end effector position given the joint angles and has a unique solution. However, in practice we wish to move the end effector of the robot to a specific position, and to do this we must set appropriate joint angles. We therefore need to solve the inverse problem, which has two solutions, as seen in Figure 6.16.

Forward problems often correspond to causality in a physical system and generally have a unique solution. For instance, a specific pattern of symptoms in the human body may be caused by the presence of a particular disease. In machine learning, however, we typically have to solve an inverse problem, such as trying to predict the presence of a disease given a set of symptoms. If the forward problem involves a many-to-one mapping, then the inverse problem will have multiple solutions. For instance, several different diseases may result in the same symptoms.