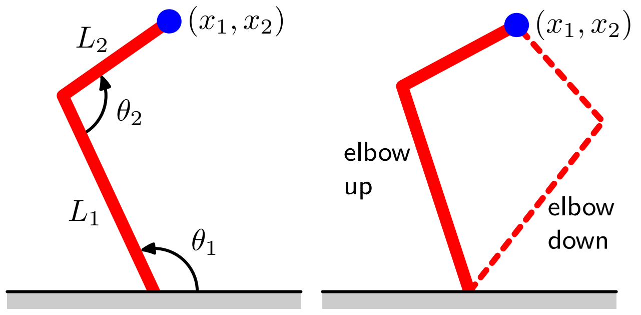

Figure 5.18 The left figure shows a two-link robot arm, in which the Cartesian coordinates (x_1, x_2) of the end effector are determined uniquely by the two joint angles θ_1 and θ_2 and the (fixed) lengths L_1 and L_2 of the arms. This is know as the *forward kinematics* of the arm. In practice, we have to find the joint angles that will give rise to a desired end effector position and, as shown in the right figure, this *inverse kinematics* has two solutions corresponding to 'elbow up' and 'elbow down'.

$$\frac{\partial \widetilde{E}}{\partial \eta_j} = \sum_i \{\pi_j - \gamma_j(w_i)\}. \tag{5.147}$$

We see that π_j is therefore driven towards the average posterior probability for component j.

5.6. Mixture Density Networks

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 5.33

As a simple example of an inverse problem, consider the kinematics of a robot arm, as illustrated in Figure 5.18. 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 5.18.

Forward problems often corresponds 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 pattern recognition, 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.

In the robotics example, the kinematics is defined by geometrical equations, and the multimodality is readily apparent. However, in many machine learning problems the presence of multimodality, particularly in problems involving spaces of high dimensionality, can be less obvious. For tutorial purposes, however, we shall consider a simple toy problem for which we can easily visualize the multimodality. Data for this problem is generated by sampling a variable x uniformly over the interval $(0, 1)$, to give a set of values $\{x_n\}$, and the corresponding target values t_n are obtained

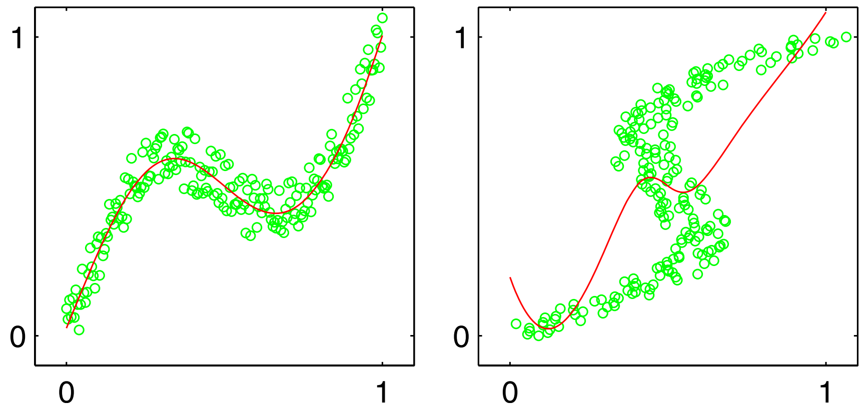

Figure 5.19 On the left is the data set for a simple 'forward problem' in which the red curve shows the result of fitting a two-layer neural network by minimizing the sum-of-squares error function. The corresponding inverse problem, shown on the right, is obtained by exchanging the roles of x and t. Here the same network trained again by minimizing the sum-of-squares error function gives a very poor fit to the data due to the multimodality of the data set.

by computing the function $x_n + 0.3\sin(2\pi x_n)$ and then adding uniform noise over the interval $(-0.1, 0.1)$. The inverse problem is then obtained by keeping the same data points but exchanging the roles of x and t. Figure 5.19 shows the data sets for the forward and inverse problems, along with the results of fitting two-layer neural networks having 6 hidden units and a single linear output unit by minimizing a sum-of-squares error function. Least squares corresponds to maximum likelihood under a Gaussian assumption. We see that this leads to a very poor model for the highly non-Gaussian inverse problem.

We therefore seek a general framework for modelling conditional probability distributions. This can be achieved by using a mixture model for $p(\mathbf{t}|\mathbf{x})$ in which both the mixing coefficients as well as the component densities are flexible functions of the input vector $\mathbf{x}$, giving rise to the *mixture density network*. For any given value of $\mathbf{x}$, the mixture model provides a general formalism for modelling an arbitrary conditional density function $p(\mathbf{t}|\mathbf{x})$. Provided we consider a sufficiently flexible network, we then have a framework for approximating arbitrary conditional distributions.

Here we shall develop the model explicitly for Gaussian components, so that

$$p(\mathbf{t}|\mathbf{x}) = \sum_{k=1}^{K} \pi_k(\mathbf{x}) \mathcal{N}\left(\mathbf{t}|\boldsymbol{\mu}_k(\mathbf{x}), \sigma_k^2(\mathbf{x})\right). \tag{5.148}$$

This is an example of a *heteroscedastic* model since the noise variance on the data is a function of the input vector $\mathbf{x}$. Instead of Gaussians, we can use other distributions for the components, such as Bernoulli distributions if the target variables are binary rather than continuous. We have also specialized to the case of isotropic covariances for the components, although the mixture density network can readily be extended to allow for general covariance matrices by representing the covariances using a Cholesky factorization (Williams, 1996). Even with isotropic components, the conditional distribution $p(\mathbf{t}|\mathbf{x})$ does not assume factorization with respect to the components of $\mathbf{t}$ (in contrast to the standard sum-of-squares regression model) as a consequence of the mixture distribution.

We now take the various parameters of the mixture model, namely the mixing coefficients $\pi_k(\mathbf{x})$, the means $\boldsymbol{\mu}_k(\mathbf{x})$, and the variances $\sigma_k^2(\mathbf{x})$, to be governed by

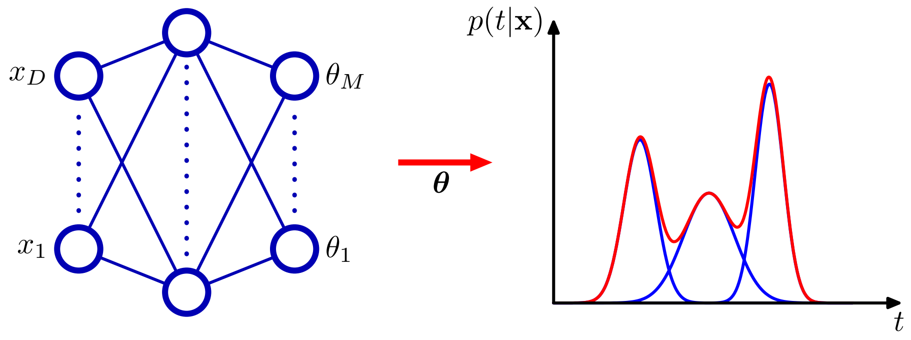

Figure 5.20 The *mixture density network* can represent general conditional probability densities $p(\mathbf{t}|\mathbf{x})$ by considering a parametric mixture model for the distribution of $\mathbf{t}$ whose parameters are determined by the outputs of a neural network that takes $\mathbf{x}$ as its input vector.

the outputs of a conventional neural network that takes $\mathbf{x}$ as its input. The structure of this mixture density network is illustrated in Figure 5.20. The mixture density network is closely related to the mixture of experts discussed in Section 14.5.3. The principle difference is that in the mixture density network the same function is used to predict the parameters of all of the component densities as well as the mixing coefficients, and so the nonlinear hidden units are shared amongst the input-dependent functions.

The neural network in Figure 5.20 can, for example, be a two-layer network having sigmoidal ('$\tanh$') hidden units. If there are L components in the mixture model (5.148), and if $\mathbf{t}$ has K components, then the network will have L output unit activations denoted by a_k^{π} that determine the mixing coefficients $\pi_k(\mathbf{x})$, K outputs denoted by a_k^{σ} that determine the kernel widths $\sigma_k(\mathbf{x})$, and $L \times K$ outputs denoted by a_{kj}^{μ} that determine the components $\mu_{kj}(\mathbf{x})$ of the kernel centres $\boldsymbol{\mu}_k(\mathbf{x})$. The total number of network outputs is given by $(K+2)L$, as compared with the usual K outputs for a network, which simply predicts the conditional means of the target variables.

The mixing coefficients must satisfy the constraints

$$\sum_{k=1}^{K} \pi_k(\mathbf{x}) = 1, \qquad 0 \leqslant \pi_k(\mathbf{x}) \leqslant 1 \tag{5.149}$$

which can be achieved using a set of softmax outputs

$$\pi_k(\mathbf{x}) = \frac{\exp(a_k^{\pi})}{\sum_{l=1}^{K} \exp(a_l^{\pi})}. \tag{5.150}$$

Similarly, the variances must satisfy $\sigma_k^2(\mathbf{x}) \geqslant 0$ and so can be represented in terms of the exponentials of the corresponding network activations using

$$\sigma_k(\mathbf{x}) = \exp(a_k^{\sigma}). \tag{5.151}$$

Finally, because the means $\boldsymbol{\mu}_k(\mathbf{x})$ have real components, they can be represented directly by the network output activations

$$\mu_{kj}(\mathbf{x}) = a^{\mu}_{kj}. \tag{5.152}$$

The adaptive parameters of the mixture density network comprise the vector $\mathbf{w}$ of weights and biases in the neural network, that can be set by maximum likelihood, or equivalently by minimizing an error function defined to be the negative logarithm of the likelihood. For independent data, this error function takes the form

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{k} \pi_k(\mathbf{x}_n, \mathbf{w}) \mathcal{N}\left(\mathbf{t}_n | \boldsymbol{\mu}_k(\mathbf{x}_n, \mathbf{w}), \sigma_k^2(\mathbf{x}_n, \mathbf{w})\right) \right\} \tag{5.153}$$

where we have made the dependencies on $\mathbf{w}$ explicit.

In order to minimize the error function, we need to calculate the derivatives of the error $E(\mathbf{w})$ with respect to the components of $\mathbf{w}$. These can be evaluated by using the standard backpropagation procedure, provided we obtain suitable expressions for the derivatives of the error with respect to the output-unit activations. These represent error signals δ for each pattern and for each output unit, and can be backpropagated to the hidden units and the error function derivatives evaluated in the usual way. Because the error function (5.153) is composed of a sum of terms, one for each training data point, we can consider the derivatives for a particular pattern n and then find the derivatives of E by summing over all patterns.

Because we are dealing with mixture distributions, it is convenient to view the mixing coefficients $\pi_k(\mathbf{x})$ as $\mathbf{x}$-dependent prior probabilities and to introduce the corresponding posterior probabilities given by

$$\gamma_k(\mathbf{t}|\mathbf{x}) = \frac{\pi_k \mathcal{N}_{nk}}{\sum_{l=1}^{K} \pi_l \mathcal{N}_{nl}} \tag{5.154}$$

where $\mathcal{N}_{nk}$ denotes $\mathcal{N}\left(\mathbf{t}_n | \boldsymbol{\mu}_k(\mathbf{x}_n), \sigma_k^2(\mathbf{x}_n)\right)$.

Exercise 5.34

The derivatives with respect to the network output activations governing the mixing coefficients are given by

$$\frac{\partial E_n}{\partial a_k^{\pi}} = \pi_k - \gamma_k. \tag{5.155}$$

Exercise 5.35

Similarly, the derivatives with respect to the output activations controlling the component means are given by

$$\frac{\partial E_n}{\partial a_{kl}^{\mu}} = \gamma_k \left\{ \frac{\mu_{kl} - t_l}{\sigma_k^2} \right\}. \tag{5.156}$$

Exercise 5.36

Finally, the derivatives with respect to the output activations controlling the component variances are given by

$$\frac{\partial E_n}{\partial a_k^{\sigma}} = -\gamma_k \left\{ \frac{\|\mathbf{t} - \boldsymbol{\mu}_k\|^2}{\sigma_k^3} - \frac{1}{\sigma_k} \right\}. \tag{5.157}$$

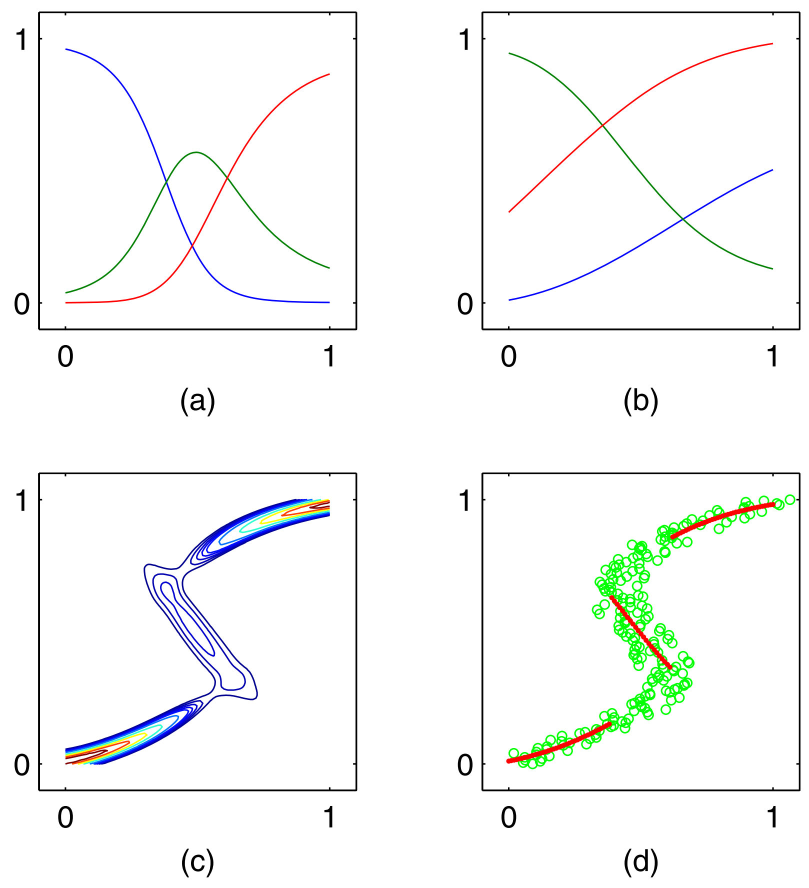

Figure 5.21 (a) Plot of the mixing coefficients $\pi_k(x)$ as a function of x for the three kernel functions in a mixture density network trained on the data shown in Figure 5.19. The model has three Gaussian components, and uses a two-layer multilayer perceptron with five '$\tanh$' sigmoidal units in the hidden layer, and nine outputs (corresponding to the 3 means and 3 variances of the Gaussian components and the 3 mixing coefficients). At both small and large values of x, where the conditional probability density of the target data is unimodal, only one of the kernels has a high value for its prior probability, while at intermediate values of x, where the conditional density is trimodal, the three mixing coefficients have comparable values. (b) Plots of the means $\mu_k(x)$ using the same colour coding as for the mixing coefficients. (c) Plot of the contours of the corresponding conditional probability density of the target data for the same mixture density network. (d) Plot of the approximate conditional mode, shown by the red points, of the conditional density.

We illustrate the use of a mixture density network by returning to the toy example of an inverse problem shown in Figure 5.19. Plots of the mixing coefficients $\pi_k(x)$, the means $\mu_k(x)$, and the conditional density contours corresponding to $p(t|x)$, are shown in Figure 5.21. The outputs of the neural network, and hence the parameters in the mixture model, are necessarily continuous single-valued functions of the input variables. However, we see from Figure 5.21(c) that the model is able to produce a conditional density that is unimodal for some values of x and trimodal for other values by modulating the amplitudes of the mixing components $\pi_k(\mathbf{x})$.

Once a mixture density network has been trained, it can predict the conditional density function of the target data for any given value of the input vector. This conditional density represents a complete description of the generator of the data, so far as the problem of predicting the value of the output vector is concerned. From this density function we can calculate more specific quantities that may be of interest in different applications. One of the simplest of these is the mean, corresponding to the conditional average of the target data, and is given by

$$\mathbb{E}[\mathbf{t}|\mathbf{x}] = \int \mathbf{t}p(\mathbf{t}|\mathbf{x})\,\mathrm{d}\mathbf{t} = \sum_{k=1}^{K} \pi_k(\mathbf{x})\boldsymbol{\mu}_k(\mathbf{x}) \tag{5.158}$$

where we have used (5.148). Because a standard network trained by least squares is approximating the conditional mean, we see that a mixture density network can reproduce the conventional least-squares result as a special case. Of course, as we have already noted, for a multimodal distribution the conditional mean is of limited value.

Exercise 5.37

We can similarly evaluate the variance of the density function about the conditional average, to give

$$\begin{aligned} s^2(\mathbf{x}) &= \mathbb{E}\left[\|\mathbf{t} - \mathbb{E}[\mathbf{t}|\mathbf{x}]\|^2 |\mathbf{x}\right] & (5.159) \\ &= \sum_{k=1}^{K} \pi_k(\mathbf{x}) \left\{ \sigma_k^2(\mathbf{x}) + \left\| \boldsymbol{\mu}_k(\mathbf{x}) - \sum_{l=1}^{K} \pi_l(\mathbf{x}) \boldsymbol{\mu}_l(\mathbf{x}) \right\|^2 \right\} & (5.160) \end{aligned}$$

where we have used (5.148) and (5.158). This is more general than the corresponding least-squares result because the variance is a function of $\mathbf{x}$.

We have seen that for multimodal distributions, the conditional mean can give a poor representation of the data. For instance, in controlling the simple robot arm shown in Figure 5.18, we need to pick one of the two possible joint angle settings in order to achieve the desired end-effector location, whereas the average of the two solutions is not itself a solution. In such cases, the conditional mode may be of more value. Because the conditional mode for the mixture density network does not have a simple analytical solution, this would require numerical iteration. A simple alternative is to take the mean of the most probable component (i.e., the one with the largest mixing coefficient) at each value of $\mathbf{x}$. This is shown for the toy data set in Figure 5.21(d).

5.7. Bayesian Neural Networks

So far, our discussion of neural networks has focussed on the use of maximum likelihood to determine the network parameters (weights and biases). Regularized maximum likelihood can be interpreted as a MAP (maximum posterior) approach in which the regularizer can be viewed as the logarithm of a prior parameter distribution. However, in a Bayesian treatment we need to marginalize over the distribution of parameters in order to make predictions.

In Section 3.3, we developed a Bayesian solution for a simple linear regression model under the assumption of Gaussian noise. We saw that the posterior distribution, which is Gaussian, could be evaluated exactly and that the predictive distribution could also be found in closed form. In the case of a multilayered network, the highly nonlinear dependence of the network function on the parameter values means that an exact Bayesian treatment can no longer be found. In fact, the log of the posterior distribution will be nonconvex, corresponding to the multiple local minima in the error function.

The technique of variational inference, to be discussed in Chapter 10, has been applied to Bayesian neural networks using a factorized Gaussian approximation