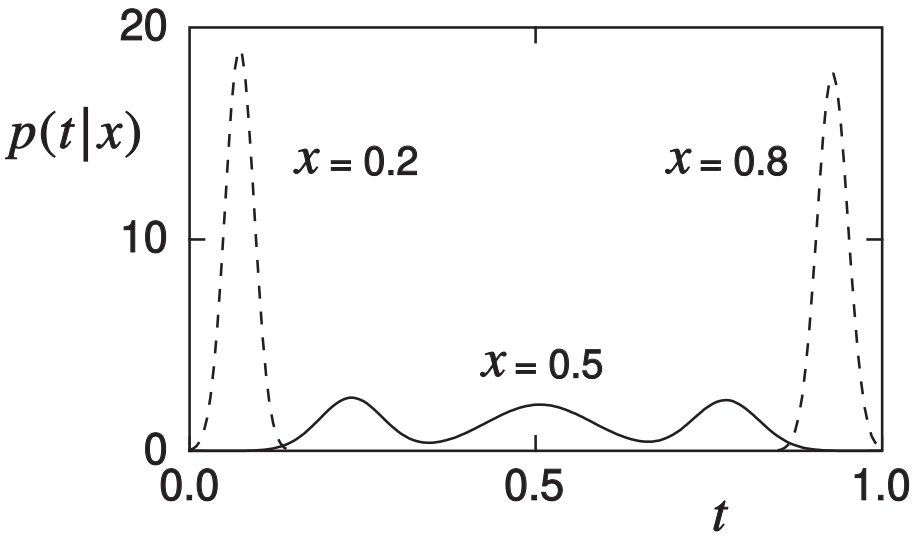

Figure 7: Plot of the conditional probability densities of the target data, for various values of x, obtained by taking vertical slices through the contours in Figure 6, for $x = 0.2$, $x = 0.5$ and $x = 0.8$. It is clear that the Mixture Density Network is able to capture correctly the multimodal nature of the target data density function at intermediate values of x.

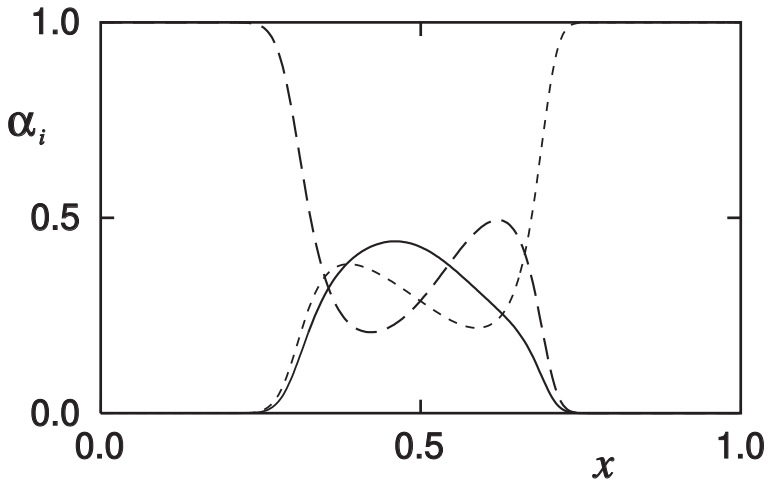

Figure 8: Plot of the priors $\alpha_i(x)$ as a function of x for the 3 kernel functions from the same Mixture Density Network as was used to plot Figure 6. At both small and large values of x, where the conditional probability density of the target data is unimodal, only one of the kernels has a prior probability which differs significantly from zero. At intermediate values of x, where the conditional density is tri-modal, the three kernels have comparable priors.

by (45), and the squared variance $s^2(x)$ of the target data around this mean, given by (47). Figure 9 shows a plot of $\langle t \mid x \rangle$ against x for the MDN used to plot Figure 6, together with plots of $\langle t \mid x \rangle \pm s(x)$. This representation corresponds to the assumption of a single Gaussian distribution for the target data, but with a variance parameter which is a function of x. While this is more general that the standard least-squares approach (which assumes a constant variance) it still gives a poor representation of the data in the multi-valued region. Notice that, in the regions where the data is single valued, the MDN gives a much smoother and more accurate representation of the conditional average of the target data than was obtained from the standard least-squares neural network as shown in Figure 5. This can be attributed to the fact that the standard network is having to make a single global fit to the whole data set, whereas the MDN uses different kernels for the different branches of the mapping.

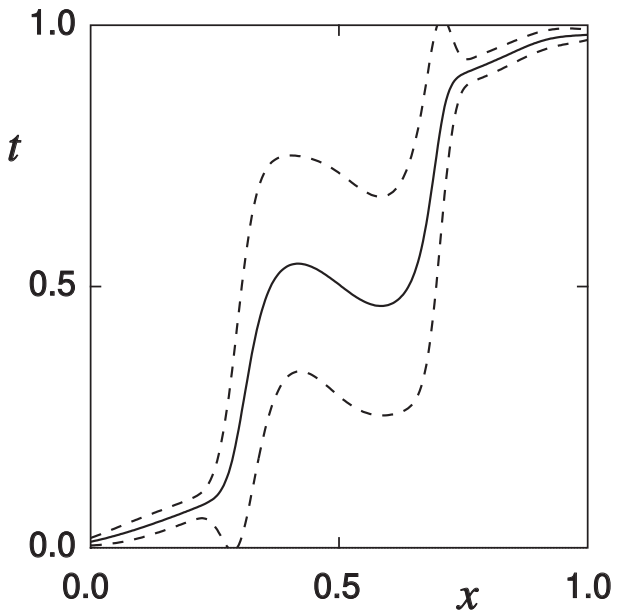

Figure 9: This shows a plot of $\langle t \mid x \rangle$ against x (solid curve) calculated from the MDN used to plot Figure 6, together with corresponding plots of $\langle t \mid x \rangle \pm s(x)$ (dashed curves). Notice that for small and large values of x, where the mapping is single-valued, the MDN actually gives a better representation of the conditional average than the standard least-squares approach, as can be seen by comparison with Figure 5. This can be attributed to the fact that the standard network is having to make a single global fit to the whole data set, whereas the MDN uses different kernels for the different branches of the mapping.

We can also consider the evaluation of the center of the most probable kernel according to (49), which gives the result shown in Figure 10. This now represents a discontinuous functional mapping from x to t, such that, at each value of x, the MDN make a good prediction for the value of t, which lies well within one of the branches of the data. It can be seen that the discontinuities correspond to the crossing points in Figure 8 which separate the regions in which different priors have the largest value. Comparison with the corresponding mapping obtained with the standard neural network approach, given in Figure 5, shows that the MDN gives substantially improved predictions for the inverse mapping.

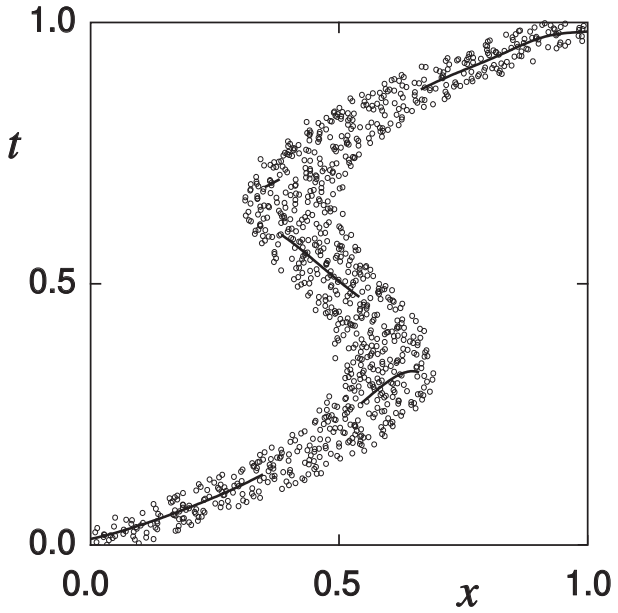

Figure 10: Plot of the central value of the most probable kernel as a function of x from the Mixture Density Network used to plot Figure 6. This gives a (discontinuous) functional mapping from x to t which at every value of x gives an accurate representation of the data. The diagram should be compared with the corresponding result obtained from a conventional neural network, shown in Figure 5.

6 Robot Kinematics

As our second application of Mixture Density Networks, we consider the kinematics of a simple 2-link robot arm, as shown in Figure 11. For given values of the joint angles (θ_1, θ_2), the end effector is moved to a position given by the cartesian coordinates

$$x_1 = L1\cos(\theta_1) - L2\cos(\theta_1 + \theta_2) \tag{56}$$

$$x_2 = L1\sin(\theta_1) - L2\sin(\theta_1 + \theta_2) \tag{57}$$

where $L1$ and $L2$ are the lengths of the two links of the robot arm. Here we consider a particular configuration of robot for which $L1 = 0.8$ and $L2 = 0.2$ and where θ_1 is restricted to the range $(0.3, 1.2)$ and θ_2 is restricted to the range $(\pi/2, 3\pi/2)$. The mapping from (θ_1, θ_2) to (x_1, x_2) is known as the *forward kinematics*, and is single-valued. However, for practical robot control, the end effector must be moved to prescribed locations and it is therefore necessary to find corresponding values for the joint angles. This is called the *inverse kinematics* and corresponds to the mapping from (x_1, x_2) to (θ_1, θ_2). In general, the inverse kinematics is not a single-valued mapping. This is illustrated in Figure 12, where we see that there are two configurations of the joint angles, known as 'elbow up' and 'elbow down', which both give rise to the same end effector position. The extent of the elbow up and elbow-down regions, for the particular configuration of robot arm considered here, is shown in Figure 13. We see that there are regions (A and C) which are accessible using only one of the two configurations, and for end effector positions in either of these regions, the inverse kinematics will be single valued. There is also a region (B) in which end effector positions can be accessed by both elbow-up and elbow-down configurations, and in this region the inverse kinematics is double-valued.

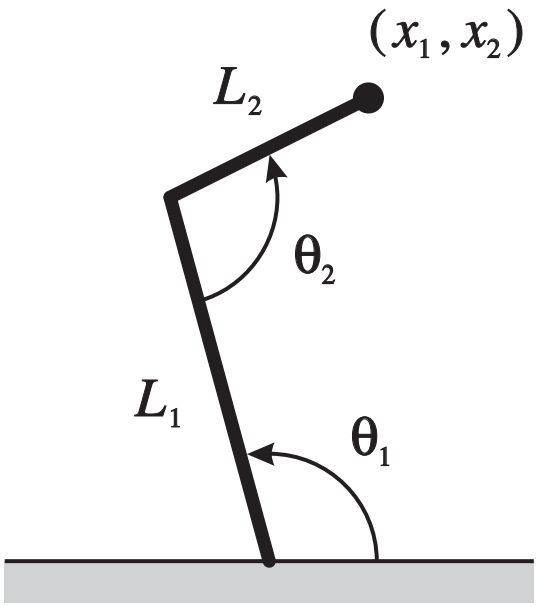

Figure 11: Schematic illustration of a two-link robot arm in two dimensions. For given values of the joint angles θ_1 and θ_2, the position of the end effector, described by the cartesian coordinates (x_1, x_2), is uniquely determined. In practice, control of the robot arm requires the solution of the *inverse kinematics* problem in which the end effector position is specified and the joint angles θ_1 and θ_2 must be found.

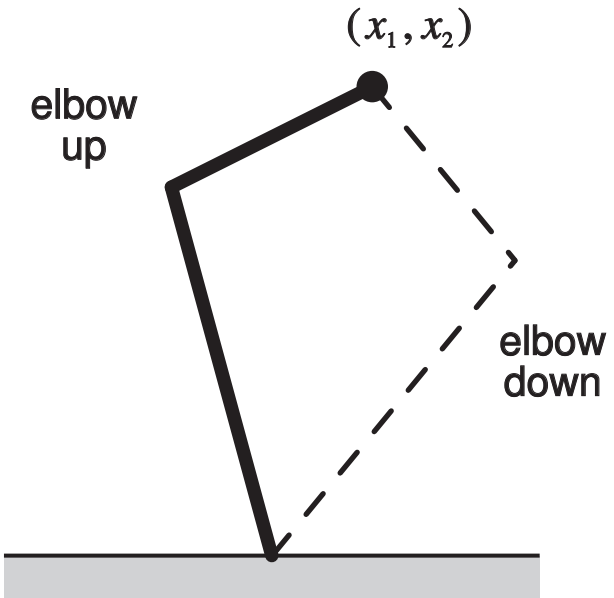

Figure 12: This diagram shows why the inverse kinematics mapping for the robot arm is multi-valued. For the given position (x_1, x_2) of the end effector, there are two solutions for the joint angles, corresponding to 'elbow up' and 'elbow down'.

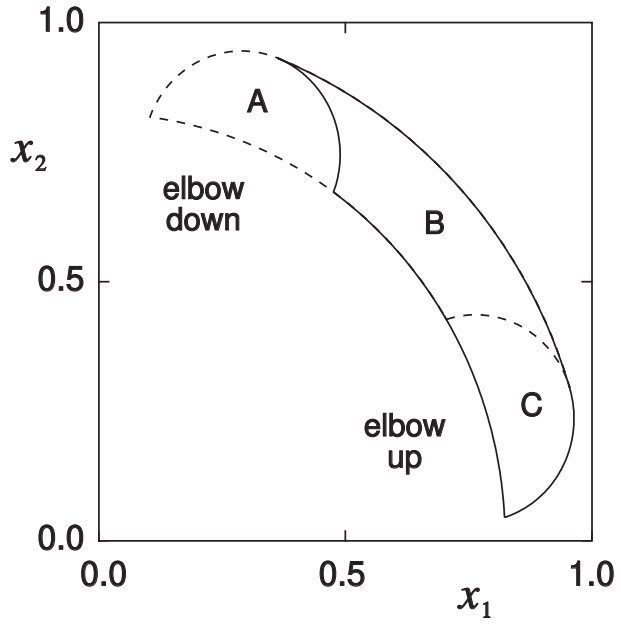

Figure 13: For the particular geometry of robot arm considered, the end effector is able to reach all points in regions A and B in the 'elbow down' configuration, and all points in regions B and C in the 'elbow up' configuration. Thus, the inverse kinematics will correspond to a single-valued mapping for positions in regions A and C, and to a double-valued mapping for positions in region B. The base of the robot arm is at $(0.0, 0.0)$.

We first consider the use of a standard neural network, trained by minimizing a sum-of-squares error function, to learn the inverse kinematics mapping. A training set of 1,000 points was generated by selecting pairs of joint angles at random with uniform distribution within the allowed limits, and computing the corresponding end effector coordinates using (56) and (57). A test set, also of 1,000 points, was generated in a similar way, but with a different random selection of joint angles. A standard multilayer perceptron network having 2 inputs, N hidden units with 'tanh' activation functions, and 2 linear output units was used. Here N was set to 5, 10, 15, 20, 25 and 30, and in each case the network was trained for 3,000 complete cycles of the BFGS algorithm. The performance of the network was assessed by presenting the test set end effector coordinates as inputs and using the corresponding values of joint angles predicted by the network to calculate the achieved end effector position using (56) and (57). The RMS Euclidean distance between the desired and achieved end effector positions is used as a measure of performance. This measure is evaluated using the test set, after every 10 cycles of training using the training set, and the network having the smallest RMS error is saved. All the networks gave very similar performance. Figure 14 shows the positioning errors achieved by the best network (20 hidden units) for all of the points in the test set. Comparison with Figure 13 shows that the positioning errors are largest in region B where the inverse kinematics mapping is double valued. In this region the end effector positions achieved by the robot lie at the outer boundary of the accessible region, corresponding to a value of $\theta_2 = \pi$. Examination of Figure 12 shows that this is indeed the result we would expect, since the average of the joint angles for an elbow-up configuration and the corresponding elbow-down configuration always gives this value for θ_2.

The same datasets were also used to train an MDN having two kernel functions in the

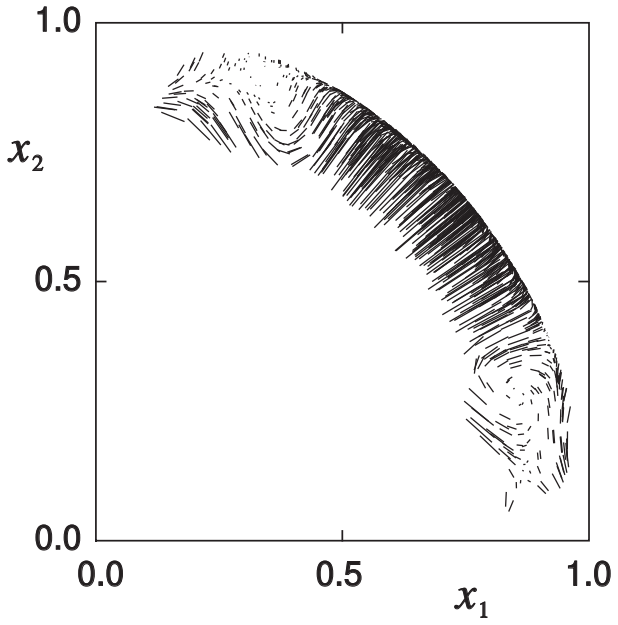

Figure 14: This shows the result of training a conventional neural network, using a sum-of-squares error function, on the inverse kinematics problem for the robot arm corresponding to Figure 13. For each of the 1,000 points in the test set, the positioning error of the end effector is shown as a straight line connecting the desired position (which forms the input to the network) to the actual position achieved by the robot when the joint angles are set to the values predicted by the outputs of the network. Note that the errors are everywhere large, but are smaller for positions corresponding to regions A and C in Figure 13 where the inverse kinematics is single valued, and larger for positions corresponding to the double valued region B.

mixture model. The network component of the MDN was a standard multilayer perceptron having two inputs, N hidden units and 8 output units, and the same optimization procedure was used as for the previous network trained by least squares. In this case the best network had 10 hidden units, and the corresponding position errors are plotted in Figure 15. It is clear that the positioning errors are reduced dramatically compared to the least-squares results shown in Figure 14. A comparison of the RMS positioning errors for the two approaches is given in Table 1, which shows that the MDN gives an order of magnitude reduction in RMS error compared to the least-squares approach.

Model	**RMS positioning error**
Least squares	0.0578
MDN	0.0053

Table 1: Comparison of RMS positioning errors for the robot end effector, measured using the test set, for a standard neural network trained by least-squares, and for a Mixture Density Network.

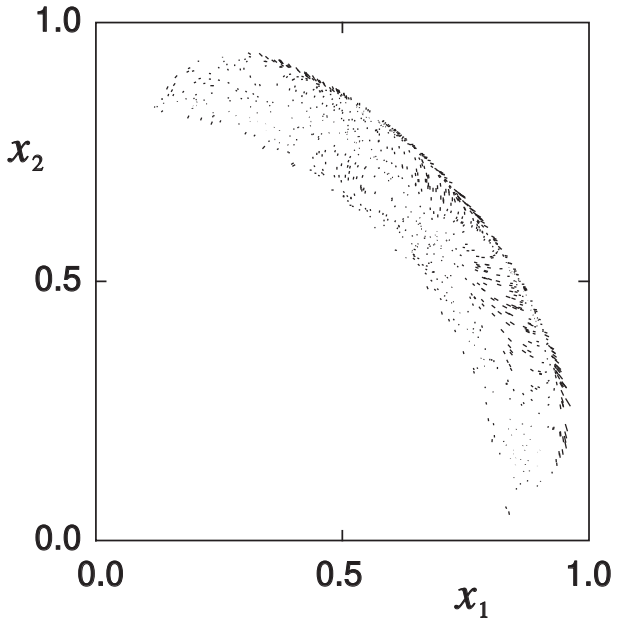

Figure 15: As in Figure 14, but showing the corresponding results obtained using a Mixture Density Network. The RMS error in positioning the end effector is reduced by an order of magnitude compared with the conventional network, and the errors remain small even in the region where the inverse kinematics is double-valued.

7 Discussion

In this paper we have introduced a new class of networks, called Mixture Density Networks, which can model general conditional probability densities. By contrast, the conventional network approach, involving the minimization of a sum-of-squares error function, only permits the determination of the conditional average of the target data, together with a single global variance parameter. We have illustrated the use of Mixture Density Networks for a simple 1-input 1-output mapping, and for a robot inverse kinematics problem. In both of these examples the required mapping is multi-valued and so is poorly represented by the conditional average.

There are many other approaches to dealing with the problem of learning multi-valued mappings from a set of example data. In general, however, these are concerned with generating one specific solution (i.e. one branch of the multivalued mapping). The Mixture Density Network, by contrast, is concerned with modelling the complete conditional density function of the output variables, and so gives a completely general description of the required mapping. From the conditional density, more specific information can be extracted. In particular, we have discussed methods for evaluating moments of the conditional density (such as the mean and variance), as well as for selecting a particular branch of a multi-valued mapping.

Implementation of Mixture Density Networks is straightforward, and corresponds to a modification of the error function, together with a different interpretation for the network outputs. One aspect of the MDN which is more complex than with standard models is the problem of model order selection. In applying neural networks to finite data sets,

the degree of complexity of the model must be optimized to give the best generalization performance. This might be done by varying the number of hidden units (and hence the number of adaptive parameters) as was done for the simulations in this paper. It could also be done through the use of regularization terms added to the error function, or through the use of 'early stopping' during training to limit the effective number of degrees of freedom in the network. The same problem of model complexity must also be addressed for MDNs. However, there is in addition the problem of selecting the appropriate number of kernel functions. Changes to the number of kernels leads to changes in the number of adaptive parameters in the network through changes to the number of output units (for a given number of hidden units), and so the two problems are somewhat interrelated. For problems involving discrete multi-valued mappings it is important that the number of kernel functions is at least equal to the maximum number of branches of the mapping. However, it is likely that the use of a greater number of kernel functions than this will have little ill effect, since the network always has the option either of 'switching off' redundant kernels by setting the corresponding priors to small values, or of 'combining' kernels by giving them similar $\boldsymbol{\mu}_i$ and σ_i parameters. Preliminary experiments on the problems discussed in this paper involving a surplus of kernels indicates that there is no significant reduction in network performance. Future research will be concerned with the automation of model order selection for Mixture Density Networks, as well as with the performance of these networks in a range of large-scale applications.

Acknowledgements

I would like to thank Pierre Baldi, David Lowe, Richard Rohwer and Andreas Weigend for providing helpful comments on an earlier draft of this report.

References

[1] H White, "Learning in artificial neural networks: a statistical perspective", *Neural Computation*, vol. 1, pp. 425–465, 1989.

[2] C M Bishop, "Novelty detection and neural network validation", in Proceedings ICANN-93, Ed. S Gielen and B Kappen, Springer-Verlag, pp. 789–794, 1993 (submitted to IEE Proceedings).

[3] S Geman, E Bienenstock and R Doursat, "Neural networks and the bias/variance dilemma", *Neural Computation*, vol. 4, pp. 1–58, 1992.

[4] C M Bishop, *Neural Networks for Statistical Pattern Recognition*, Oxford University Press, to appear, 1994.

[5] K Hornik, M Stinchecombe and H White, "Multilayer feedforward networks are universal approximators", *Neural Networks*, vol. 2, pp. 359–366, 1989.

[6] W H Press, B P Flannery, S A Teukolsky and W T Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, second edition, Cambridge University Press, 1992.

[7] G J McLachlan and K E Basford, *Mixture Models: Inference and Applications to Clustering*, Marcel Dekker, New York, 1988.

[8] J S Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition", In *Neurocomputing: Algorithms, Architectures and Applications*, Ed. F Fogelman Soulié and J Hérault, Springer-Verlag, 1990.

[9] R A Jacobs, M I Jordan, S J Nowlan and G E Hinton, "Adaptive mixtures of local experts", *Neural Computation*, vol. 3, pp. 79–87, 1991.

[10] S J Nowlan and G E Hinton, "Simplifying neural networks by soft weight sharing", *Neural Computation*, vol. 4, pp. 473–493, 1992.