To appear in: *Neural Computation, 7:3*, 1995.

Competition and Multiple Cause Models

Peter Dayan
Department of Computer Science
University of Toronto
6 King's College Road
Toronto, Ontario M5S 1A4
Canada
dayan@cs.toronto.edu

Richard S Zemel
CNL
The Salk Institute
PO Box 85800
San Diego, CA 92186-5800
USA
zemel@salk.edu

Abstract

If different causes can interact on any occasion to generate a set of patterns, then systems modelling the generation have to model the interaction too. We discuss a way of combining multiple causes that is based on the Integrated Segmentation and Recognition architecture of Keeler, Rumelhart and Leow (1991). It is more co-operative than the scheme embodied in the mixture of experts architecture, which insists that just one cause generate each output, and more competitive than the noisy-or combination function which was recently suggested by Saund (1994a;b). Simulations confirm its efficacy.

1 Introduction

Many learning techniques are derived from a *generative* view. In this, inputs are seen as random samples drawn from some particular distribution, which it is then the goal of learning to unearth. One popular class of distributions has a hierarchical structure – one random process chooses which of a set of high-level *causes* will be responsible for generating some particular sample, and then another random process, whose nature depends on this choice (and which itself could involve further hierarchical steps), is used to generate the actual sample. Self-supervised learning methods attempt to invert this process to extract the parameters governing generation – a popular choice has the high-level causes as multivariate Gaussian distributions, and the random choice between them to be a pick from a multinomial distribution. Individual input samples are attributed more or less strongly to the estimated high-level Gaussians, and the parameters of those Gaussians are iteratively adjusted to reflect the inputs for which they are held responsible. In the supervised case, a method such as the mixture of experts (Jacobs *et al*, 1991) has 'expert' modules as the high-level causes and divides responsibility for the input examples amongst them.

In these methods, the high-level generators typically compete so that a single winner accounts for each input example (one Gaussian or one expert). In many cases, however, it is desirable for more than one cause or expert to account for a single example. For instance, an input scene composed of several objects might be more efficiently described using a different generator for each object rather than just one generator for the whole input, if different objects occur somewhat independently of each other. An added advantage of such *multiple cause* models is that a few causes may be applied combinatorially to generate a large set of possible examples.

The goal in a multiple cause learning model is therefore to discover a vocabulary of independent causes or generators such that each input can be completely accounted for by the cooperative action of a few of these possible generators (which are typically represented in connectionist networks by the activation of hidden units). This is closely related to the sparse distributed form of representation advocated by Barlow (1961), who suggested representing an input as a combination of non-redundant binary features, each of which is a collection of highly correlated properties. For the autoencoder networks which we treat here, in which the network is trained to reconstruct the input on its output units, the goal of learning the underlying distribution can be viewed in terms of learning a set of priors and conditional priors to minimise the description length of a set of examples drawn from that distribution (Zemel, 1993; Hinton & Zemel, 1994).

Learning multiple causes is challenging, since cooperation (the use of several causes per input) has to be balanced against competition (the separation of the independent components in the input). Standard networks tend to err on the side of cooperation, with widely distributed patterns of activity. One approach that has been tried to counter this is to add terms to the objective function encouraging the hidden units to be independent and binary, *eg* Barlow *et al* (1989) and Schmidhuber (1992). Another approach is to encourage sparsity in the activities of the hidden units, *eg* Földiák (1990) and Zemel (1993).

Saund (1994a;b) advocated a third approach. He considered a form of autoencoder network in which the hidden units signal features and the hidden-output weights describe the way in which features generate predictions of the inputs. He suggested replacing the conventional sigmoid at the output layer with a *noisy-or* activation function (*eg* Pearl 1986), which allows multiple causes to cooperate in a probabilistically justified manner to activate the output units and hence reconstruct the input. While the noisy-or function allows multiple causes to account for a given example, it does not particularly encourage these causes to account for different parts of the input.

In this paper, we use the probabilistic theory which underlies Keeler, Rumelhart and Leow's (1991) Integrated Segmentation and Recognition architecture to suggest a way for multiple causes to interact that is more competitive than the noisy-or and more cooperative

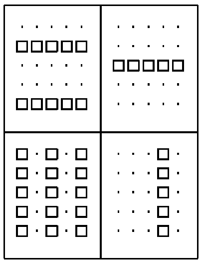

Figure 1: Bar Patterns. Two horizontal and two vertical bar patterns on a 5×5 pixel grid taken from the set used for training. The input values for the dots are 0 and those for the white boxes are 1.

than the unsupervised and supervised schemes, such as the mixture of experts, which assume that each example is generated by just a single cause. We propose an activation function which handles the common situation in which several causes combine to generate an input, but the value along a single output dimension (such as a single pixel in an image) can always be specified as coming from just one cause (even if there are many active causes that *could* have specified it). This discourages two causes from sharing partial responsibility for an output rather than taking full credit or blame. Sharing hinders the extraction of the independent elements in the input. We demonstrate that this new approach can learn appropriate representations.

2 The Bars

A simple example which motivated the model and the need for competition is one of extracting a number of independent horizontal and vertical bars on an input pixel grid (Földiák, 1990; Saund, 1994b; Zemel, 1993). Four examples of patterns are shown in figure 1 for a 5×5 grid. They were generated in a three stage process. First the direction, horizontal or vertical, was chosen (in our case, each being equiprobable). Then each of the five bars in that direction was independently chosen with some probability ($p = 0.2$). Finally the pixels corresponding to the chosen bars were turned from off (black; shown with lines) to on (white) deterministically. In general noise could be introduced at this stage too (Saund, 1994b). Previous uses of the bars omitted the first stage and allowed both horizontal and vertical bars in the same image.

We trained an autoencoder network with a single hidden layer to capture the structure in 500 of these patterns using the sigmoid and the noisy-or activation functions at the output layer and employing a cross-entropy error to judge the reconstructions. Zemel (1993) described how such autoencoder networks can be seen as generalisations of almost all existing self-supervised learning algorithms and architectures, provided that probabilistic priors over the activations of the hidden units are appropriately set and the deviations of these activations from their priors are penalised along with the errors in reconstruction. This amounts to using an error measure which is the description length of the set of inputs using as a code the activation of the hidden units. Minimising this error measure amounts to the use of an (approximate) minimum description length (MDL) strategy. We employed such an error measure, in this case setting the priors on the hidden unit activations commensurate with the actual generative model we used, assuming that the hidden units would come to code for the independent bars. These priors do not force this as a solution, however, as is evident in the sub-optimal weights in figure 2. [1]

Figure 2 shows the weights learned using the sigmoid and noisy-or output activation schemes, which clearly reveal the generative model they embody. Only 10 hidden units were allowed, which is the minimum number possible in this case. The sigmoidal scheme fails to capture the separate generators, and indeed reconstructs the inputs quite poorly (it never succeeded in extracting the generators, *ie* the bars, in 100 trials from different random starting weights). The noisy-or does much better, pulling out all the bars. However, 73% of the time (73 trials out of 100) it gets stuck at a local minimum in which one or more bars do not have individual generators (the figure shows one example). These local minima are significantly sub-optimal in terms of the coding cost. On the same problem, the more competitive rule described in the next section gets caught in a local minimum 31% of the time (31 trials out of 100). Figure 3 shows an example of the weights that this rule produced, and the individual generators are evident.

Although it might seem like a toy problem, the 5×5 bar task with only 10 hidden units turns out to be quite hard for all the algorithms we discuss. The coding cost of making an error in one bar goes up linearly with the size of the grid, so at least one aspect of the problem gets *easier* with large grids. The competitive scheme also worked better than the noisy-or when horizontal and vertical bars were mixed in the same input example, although it does fail slightly more often than in the earlier case.[2] With appropriate weights, the imaging model can be correct for all the three schemes, and it is hard to extract from suboptimal learning behaviour why different tasks have different failure rates. Both the noisy-or and the competitive activation rules worked well when more than 10 hidden units were used, but the sigmoid rule consistently failed.

Saund (1994a;b) did not use a set of input-hidden weights to generate the activities of the hidden units. Instead, he used an iterative inner optimisation loop, which might be

[1] Zemel (1993) judiciously set the value for this prior probability as a means of encouraging sparsity, *ie* discouraging the system from finding solutions in which single hidden units each generate more than one bar. Here the prior is appropriate to the generative scheme (modulo a lower order effect from the incapacity of the architecture to capture the correlations between the hidden units that generate bars in the same direction).

[2] The noisy-or activation rule failed to extract the bars on 75 out of 100 random trials, the competitive activation rule failed in 39 of 100 trials.

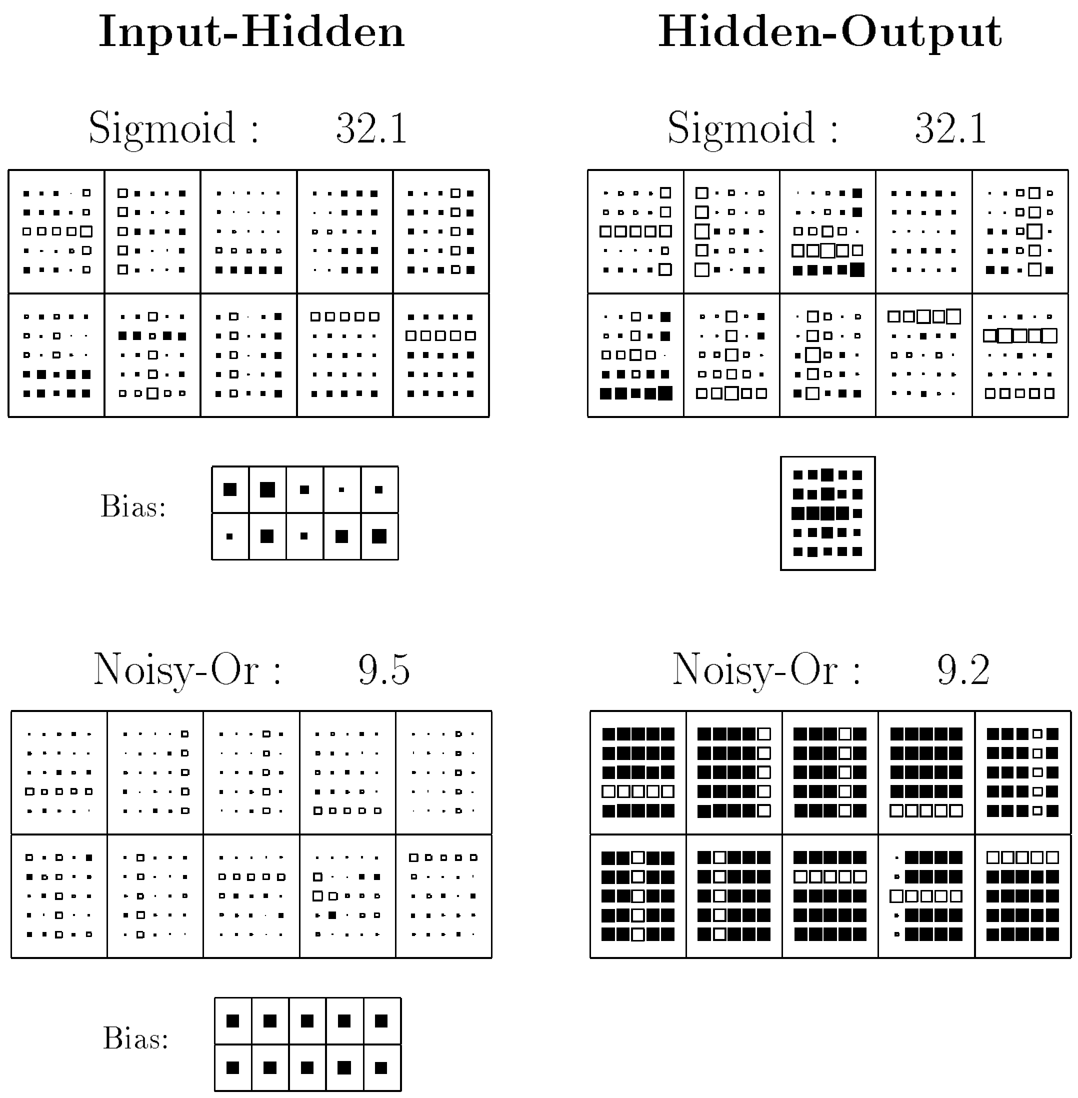

Figure 2: Bar weights. The input-hidden, hidden-output, and hidden unit and output bias weights based on learning from 250 horizontal and 250 vertical bar patterns, with each bar coming on with probability 0.2 in patterns of its direction. The top two rows show the case for sigmoidal output activation – only a few of the underlying generators are visible in the hidden-output weights and reconstruction is poor. Only the sigmoid activation function employs biases for the output units. The bottom two rows show the improvement using the noisy-or (note that hidden-output weights for the noisy-or should be probabilities and the ones shown are passed through a sigmoid before being used). However when the conjugate gradient minimisation procedure gave up, one of the hidden units takes responsibility for more than one bar, and the magnitude of the weights made recalcitrant this suboptimal solution. Black weights are negative, white positive, and the scale for each group (indicated by the number in each figure) is the magnitude of the largest weight.

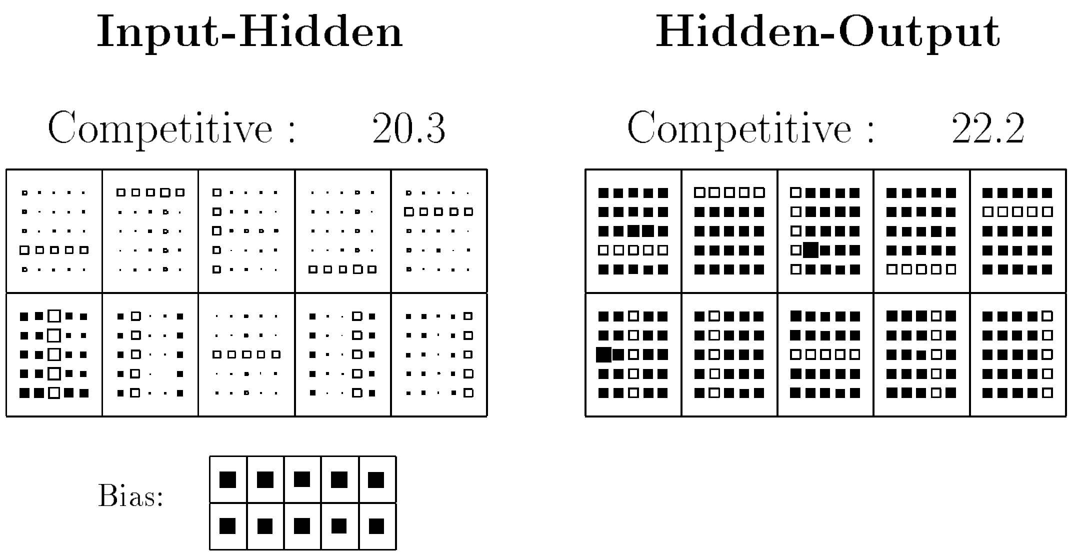

Figure 3: Bar weights using the competitive activation function described in section 3 (in this case hidden-output weights for this scheme represent odds, and the values shown are passed through the exponential function before being used). These weights exactly capture the generative scheme underlying patterns as there are individual generators for each bar.

---

expected to be more powerful for both the noisy-or and the competitive rule. We did not use such an inner loop because we are interested in hierarchical unsupervised learning (Dayan *et al*, 1994). The error surface for the activations of units in multiple layers has multiple modes, and these are computationally expensive to explore.

3 A competitive activation function

For simplicity, we describe the model for the self-supervised learning case, but it applies more generally. The noisy-or activation function comes from a particular form of stochastic generative model. Our competitive activation function comes from a different model which we now describe. The starting point for both models is the same – a set of binary representation units s_i whose activations are independent choices from binomials, with $\mathcal{P}[s_i = 1] = p_i$ (pattern indices are omitted for clarity). An overall pattern is generated by picking a set of these to be active (like picking a set of bars in the example above) and then using this set to generate the probability that the activity y_j of binary output unit j is 1. Since the output units are binary, a cross-entropy error measure is used.

The bars example (figure 1) naturally fits a *write-white* model in which a pixel j is generally black ($y_j = 0$) unless one of the causes seeks to turn it white. Given binary activities s_i, Saund (1994) recommended the use of the noisy-or (NO) combination function to calculate the probability that outputs should be white. If c_{ij} is the probability that $y_j = 1$ given the presence of cause s_i, then

$$p_j^{NO} \equiv \mathcal{P}^{NO}[y_j = 1] = 1 - \prod_i (1 - s_i c_{ij}) \tag{1}$$

since just one of the causes has to turn the pixel on for it to be white. A trouble with this is that if $c_{i_1 j} < 1$ for some potential cause i_1, then the other causes that are active are encouraged, using the noisy-or, to have $c_{ij} > 0$ to increase the overall value of p_j^{NO}. In the same way that (Nowlan, 1990; Jacobs *et al.*, 1991) showed that learning for the mixtures of experts is much more straightforward using their competitive rule than it was for the more cooperative rule used in (Jacobs, Jordan & Barto 1991), we might expect that having the system infer the independent causes would require a measure of competition.

Our generative model uses a more competitive procedure (C) for generating a pixel that forces at most one cause to take responsibility on any occasion. Define $c_{ij} < 1$ to be the probability that cause s_i seeks to turn pixel j white. The easiest way to describe the model involves a set of responsibility flags f_{ij}, which are chosen to be 0 or 1 according to:

$$\mathcal{P}[f_{ij} = 1] = \begin{cases} c_{ij} & \text{if } s_i = 1 \\ 0 & \text{if } s_i = 0 \end{cases}$$

If $f_{ij} = 0$ for all i, then we set $y_j = 0$; if $f_{ij} = 1$ for exactly one i, we set $y_j = 1$; and otherwise we pick a new set of f_{ij} from the distribution above and look again. It is clear that just one cause will take responsibility for generating pixel j on any occasion – this is the required competition. The f_{ij} do not appear explicitly in the calculations below, however they are responsible for the resulting conditional probabilities.

This makes the overall probability that $y_j = 1$

$$\begin{aligned} p_j^C &\equiv \mathcal{P}^C[y_j = 1] \\ &= \mathcal{P}[y_j = 1 \mid \text{at most one cause turns } j \text{ white}] \\ &= \frac{\mathcal{P}[y_j = 1 \text{ \& at most one cause turns } j \text{ white}]}{\mathcal{P}[\text{at most one cause turns } j \text{ white}]} \\ &= \frac{\mathcal{P}[\text{a single cause turns } j \text{ white}]}{\mathcal{P}[\text{no cause turns } j \text{ white}] + \mathcal{P}[\text{a single cause turns } j \text{ white}]} && (2) \\ &= \sum_i \frac{\mathcal{P}[\text{only cause } i \text{ turns } j \text{ white}]}{\mathcal{P}[\text{no cause turns } j \text{ white}] + \sum_k \mathcal{P}[\text{only cause } k \text{ turns } j \text{ white}]} && (3) \end{aligned}$$

More quantitatively, the probability that *only* cause i turns j white is,

$$s_i c_{ij} \prod_{l \neq i} (1 - s_l c_{lj}), \tag{4}$$

the likelihood that *no* cause turns j white is the complement of the noisy-or,

$$\prod_l (1 - s_l c_{lj}) \tag{5}$$

and substituting these into equation 3, we get

$$p_j^C = \sum_i \frac{s_i c_{ij} \prod_{l \neq i}(1 - s_l c_{lj})}{\prod_l (1 - s_l c_{lj}) + \sum_k s_k c_{kj} \prod_{l \neq k}(1 - s_l c_{lj})} \tag{6}$$

$$= 1 - \frac{1}{1 + \sum_k \frac{s_k c_{kj}}{1 - c_{kj}}} \tag{7}$$

using the facts that the ratio of equations 4 and 5 is just the odds $\frac{s_i c_{ij}}{1 - c_{ij}}$ that cause i generates j, and s_i is either 0 or 1. The sum of the odds in the denominator of equation 7 plays an equivalent role in the Integrated Segmentation and Recognition (ISR) system. We return to this point below.

An alternative way of looking at this conditional probability is that whereas for noisy-or

$$\mathcal{P}^{NO}[y_j = 1] = 1 - \mathcal{P}[\text{no cause turns j white}], \qquad \text{here,}$$

$$\mathcal{P}^{C}[y_j = 1] = 1 - \frac{\mathcal{P}[\text{no cause turns j white}]}{\mathcal{P}[\text{at most one cause turns j white}]}$$

Both of these schemes are monotonic: if a single model increases its probability of turning a pixel white, then the probability that that pixel is white also increases. The competitive scheme, however, has a different behaviour from the noisy-or for a fixed probability $\mathcal{P}$[no cause turns *j* white], in that distributing the probability that a cause turns j white among various causes decreases the probability that j will be white. Consider the difference between having one cause whose $c_{1j} = 0.75$ and two causes whose $c_{ij} = 0.5$ each. $\mathcal{P}$[no cause turns *j* white] $= 0.25$ in both cases. For the noisy-or, $\mathcal{P}^{NO}[y_j = 1] = 0.75$ in both cases, while in the competitive scheme, $\mathcal{P}^{C}[y_j = 1] = 0.75$ for the first case but only 0.67 in the second case.

An alternative way of comparing these two functions is shown in figure 4. When the first of two causes ($s_1 = s_2 = 1$) is not keen to turn pixel j white ($c_{1j} = 0.1$), the probability that pixel j is white depends directly on the value of c_{2j} for both the noisy-or and the competitive functions. However, when the first cause is keen to take responsibility for j ($c_{1j} = 0.9$), then the two functions have different behavior: in order to increase p_j, the noisy-or attempts to increase c_{2j}, while for the competitive scheme, p_j is largely independent of c_{2j}, at least until $c_{2j} \sim c_{1j}$.

Equation 7 is exactly the generative version of the forward model Keeler, Rumelhart and Leow (1991) used for their Integrated Segmentation and Recognition (ISR) architecture. They wanted to train networks to perform the segmentation and recognition necessary to

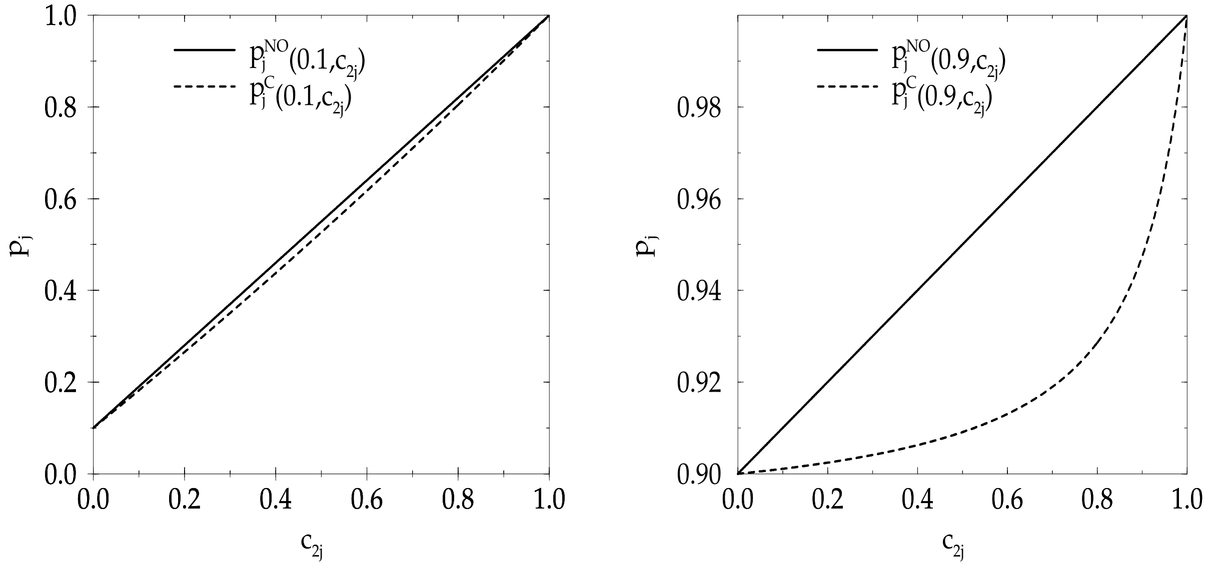

Figure 4: Noisy-or versus competitive scheme. The probability p_j that pixel j is white is plotted as a function of c_{2j}, the responsibility that cause 2 takes for turning pixel j white. The plot on the left shows that the noisy-or and competitive activation functions have similar behavior when the other cause is unlikely to take responsibility for pixel j ($c_{1j} = 0.1$). The plot on the right shows that when this cause is likely to turn j white ($c_{1j} = 0.9$), the noisy-or can still increase p_j by increasing c_{2j}, whereas the competitive scheme largely ignores c_{2j} until $c_{2j} \sim c_{1j}$ since the first cause will already largely take responsibility for j. Note the difference in the scale of p_j.

extract the 5 digits in a zip-code. During training, they only specified whether or not a digit was present in a particular image, and the network had to work out how to assign credit at the different spatial positions in an input to recognisers for the different digits. Weights were shared for the recognisers for each digit across all locations. They regarded the output of the digit recognisers as being the equivalent of c_{ij}, the probability that digit j is at position i, and, using a constraint that each digit should appear either no times or just once in any image, calculated the overall outputs of the network as the sums of the odds over all positions in the image (so $s_i = 1, \forall i$), just as in equation 7. Of course, c_{ij} in the competitive scheme (equation 7) are learned weights rather than activities.

There is also an interesting relationship between this activation function and that in the mixture of experts architecture (Jacobs *et al*, 1991). In the mixture of experts, the output of each expert module is gated by its responsibility for the input example. The competitive scheme computes a similar quantity. For this simple write-white example, we take the output of each cause, or expert module, to be 1 for pixel j, and also use a null cause with output 0 to account for the case that no cause takes responsibility for j. Equation 7 sums across the active causes, where the responsibility that cause i bears for the input is normalised across the other causes k and the null cause.

This competitive scheme therefore introduces an unorthodox form of competition. Here the units are not competing for activity, but instead are competing over responsibility for the individual output units.

4 Error function and mean-field approximation

It is convenient to use the odds $b_{ij} = \frac{c_{ij}}{1-c_{ij}}$ as the underlying adaptive parameter. Then, given a set of binary s_i, the function in Equation 7 resembles the positive part of a *tanh* activation function.

We use a cross-entropy error measure for pixel j:

$$-E_j^C = t_j \log p_j^C + (1 - t_j)\log(1 - p_j^C)$$

where t_j is the true probability that pixel j is on (which is usually 0 or 1), we have

$$\frac{\partial E_j^C}{\partial b_{ij}} = \frac{1 - p_j^C}{p_j^C}(t_j - p_j^C)s_i$$

Were gradient descent to be used, this would be just a modification of the delta rule (itself exactly what the sigmoid activation function would give), only weight changes are magnified if $p_j^C < 0.5$ and shrunk if $p_j^C > 0.5$. The equivalent for the noisy-or has

$$\frac{\partial E_j^{NO}}{\partial b_{ij}} = \frac{1}{p_j^{NO}}(t_j - p_j^{NO})s_i$$

which lacks the reduction in the gradient as $p_j^{NO} \to 1$.

In the case that the s_i are themselves stochastic choices from underlying independent binomials, we need an estimate of the expected cost under the cross-entropy error measure, namely

$$-\mathcal{E}_{\{s_i\}}[E_j] = \mathcal{E}_{\{s_i\}}[t_j \log p_j^C + (1 - t_j)\log(1 - p_j^C)]$$

One way to do this would be to collect samples of the $\{s_i\}$. Another way, which is a rather crude approximation, but which has worked, is to use

$$t_j \log \bar{p}_j^C + (1 - t_j)\log(1 - \bar{p}_j^C)$$

where

$$\bar{p}_j^C = \left(1 - \frac{1}{1 + \sum_i p_i b_{ij}}\right)\left(1 - \prod_i (1 - p_i \frac{b_{ij}}{1 + b_{ij}})\right) \tag{8}$$

The term on the left is just a mean field approximation to the activation function from equation 7 (using p_i in place of s_i). The extra term on the right takes partial account of the possibility that none of the s_i are on – this is underestimated in the term $\sum_i p_i b_{ij}$ which is insensitive to the generative priority of the p_i in that the s_i are *first* generated from the p_i before the f_{ij} are picked. For this, we employ just the noisy-or, written in terms of the odds b_{ij}. We used this mean field approximation to generate the results in figure 3.

Figure 5 shows how both the approximation in equation 8 and the simpler approximation $\hat{p}_j^C = 1 - 1/(1 + \sum_i p_i b_{ij})$ compare to the true value of p_j^C in a case like the one before of two causes, where $p_1 = 1$, $c_{1j} = 0.5$ and across different values of p_2 and c_{2j}. An anonymous referee pointed out the substantial difference between the true $p_j^C = 0.67$ and $\bar{p}_j^C = 0.5$ for $p_2 = 1$ and $c_{2j} = 0.5$. From our experiments, the important case seems to be as $c_{2j} \to 1$, and we can see that $\bar{p}$ is better than $\hat{p}$ in this limit.

5 Discussion

We have addressed the problem of how multiple causes can jointly specify an image, in the somewhat special case in which they interact at most weakly. We used this last constraint in the form of a generative model in which the probability distribution of the value of each pixel is specified on any occasion by just one cause (or a null or bias cause). This is the generative form of Keeler, Rumelhart and Leow's summing forward model in their Integrated Segmentation and Recognition architecture. The model is more competitive than previous schemes, such as the noisy-or, linear combination, or combination using a sigmoid activation function, and has application outside the self-supervised autoencoding examples that have motivated our work. For instance one could use a function based on this for the supervised learning in Nowlan and Sejnowski's (1993) model of motion

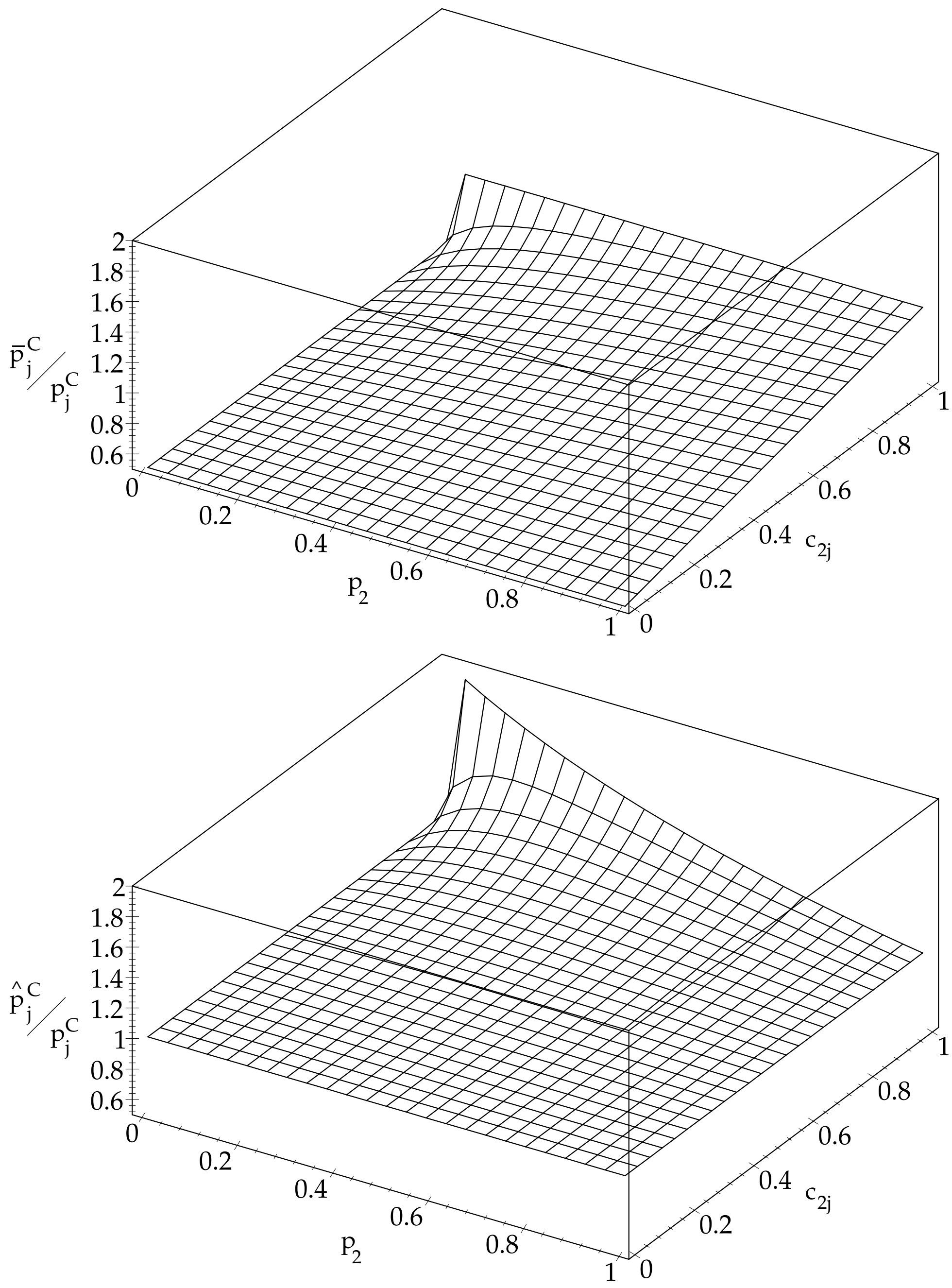

Figure 5: Mean-field approximations to p_j^C. The graphs show the ratios of $\bar{p}_j^C$ and $\hat{p}_j^C$ to p_j^C for the case of two causes, where $p_1 = 1$ and $c_{1j} = 0.5$. The behaviour of $\hat{p}_j^C$ at $c_{2j} = 1$ and small p_2 exhibits the insensitivity mentioned in the text.

segmentation, in which each local region in an image is assumed to support at most one image velocity.

There is a natural theoretical extension of this model to the case of generating grey-values for pixels rather than black or white ones. This uses the same notion of competition as above—at most one cause is responsible for generating the value of a pixel—but allows different causes to maintain different probabilities t_{ijk} of setting $y_j = k$, where k corresponds to a real-valued activation of the pixel. The b_{ij} odds again determine the amount of responsibility generator i takes for setting the value j, and the t_{ijk} would determine what i would do with the pixel if it is given the opportunity. This scheme also requires a bias t_{*jk} which is the probability that $y_j = k$ if none of the causes wins in the f_{ij} competition.

This makes:

$$p^C_{jk} = \mathcal{P}^C[y_j = k] = \frac{t_{*jk} + \sum_i s_i b_{ij} t_{ijk}}{1 + \sum_l s_l b_{lj}} \tag{9}$$

for the case of binary s_i. Note that equation 7 is a simple case of equation 9 where $t_{ij1} = 1$ for each cause and the bias is zero.

Once again, we can sample from the distribution generating the s_i to calculate the expected cost of coding y_j using this as the prior. We have considered the case where k can be black (0) or white (1) as a way of formalising a *write white-and-black* imaging model (Saund, 1994a;b). Unfortunately a mean field version of equation 9 which combines p^C_{j0} and p^C_{j1} in a manner analogous to equation 8 yields a poor approximation. Causes with b_{ij} very large, p_i moderate and $t_{ij0} = 1$ can outweigh causes with b_{ij} moderate, $p_i = 1$ and $t_{ij1} = 1$. Saund (1994a;b) used a technique that separates out the contributions from causes that try to turn the pixel black from those that try to turn it white before recombining them. This can be seen as a different mean field approximation to equation 9. However it did not perform well in the examples we tried, suggesting that it might rely for its success on Saund's more powerful activation scheme, which has an inner optimisation loop.

The weak interaction that the competitive schemes use is rather particular – in general there may be causes that are separable on different dimensions but which interact strongly in producing an output (*eg* base pitch and timbre for a musical note, or illumination and object location for an image). The same competitive scheme as here could be used within a dimension (*eg* notes at different gross pitches might have roughly separable spectrograms like the horizontal bars in the figure) but learning how they combine is more complicated, introducing such issues as the binding problem. Yet it has applications to many interesting and difficult problems, such as image segmentation, where complex occlusion instances can be described based on the fact that each local image region can be accounted for by a single opaque object.

Acknowledgements

We are very grateful to Virginia de Sa, Geoff Hinton, Terry Sejnowski, Paul Viola and Chris Williams for helpful discussions, to Eric Saund for generously sharing unpublished results, and to two anonymous reviewers for their helpful comments. Support was from grants to Geoff Hinton, the Canadian NSERC, Terry Sejnowski, and the ONR.

References

[1] Barlow, H (1961). The coding of sensory messages. In *Current Problems in Animal Behaviour.* Cambridge, England: CUP, 331-360.

[2] Barlow, H, Kaushal, T & Mitchison, G (1989). Finding minimum entropy codes. *Neural Computation,* **1**, 412-423.

[3] Dayan, P, Hinton, GE, Neal, RM & Zemel, RS (1994). The Helmholtz machine. Submitted to *Neural Computation.*

[4] Földiák, P (1990). Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics,* **64**, 165-170.

[5] Jacobs, RA, Jordan, MI & Barto, AG (1991). Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive Science,* **15**, 219-250.

[6] Jacobs, RA, Jordan, MI, Nowlan, SJ & Hinton, GE (1991). Adaptive mixtures of local experts. *Neural Computation,* **3**, 79-87.

[7] Keeler, JD, Rumelhart, DE & Leow, WK (1991). Integrated segmentation and recognition of hand-printed numerals. In RP Lippmann, J Moody & DS Touretzky, editors, *Advances in Neural Information Processing Systems, 3.* San Mateo, CA: Morgan Kaufmann, 557-563.

[8] Nowlan, SJ (1990). *Competing Experts: An Experimental Investigation of Associative Mixture Models.* Technical report CRG-TR-90-5, Department of Computer Science, University of Toronto, Canada.

[9] Nowlan, SJ & Sejnowski, TJ (1993). Filter selection model for generating visual motion signals. In SJ Hanson, JD Cowan & CL Giles, editors, *Advance in Neural Information Processing Systems, 5.* San Mateo, CA: Morgan Kaufmann, 369-376.

[10] Pearl, J (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo, CA: Morgan Kaufmann.

[11] Saund, E (1994a). Unsupervised learning of mixtures of multiple causes in binary data. In JD Cowan, G Tesauro & J Alspector, editors, *Advance in Neural Information Processing Systems, 6.* San Mateo, CA: Morgan Kaufmann.

[12] Saund, E (1994b). A multiple cause mixture model for unsupervised learning. Submitted for publication.

[13] Schmidhuber, JH (1992). Learning factorial codes by predictability minimization. *Neural Computation,* **4**, 863-879.

[14] Zemel, RS (1993). *A Minimum Description Length Framework for Unsupervised Learning.* PhD Dissertation, Computer Science, University of Toronto, Canada.