connections. This is not surprising, since these weights did not affect the states of units in the non-settling case; so a pair of units which learned, for example, to respond to left-shifted patterns, could theoretically keep increasing the relative probability of network states for coherent left-shifted patterns by continually increasing the weight on the connection between the pair. For the experiments described in the remainder of this chapter, all networks were trained only with the non-settling procedure, using the method of conjugate gradients.

There are several ways in which we could evaluate the performance of the network on the shift problem. The absolute value of the objective function, or the relative value of state probabilities on positive versus negative cases, would indicate the overall performance of the network. But these measures do not tell us how the network has learned to solve the problem, in terms of the representation learned by the output units. A better measure for this purpose is the probability that pairs of output units in different modules are on together. This joint probability can be computed for different classes of patterns, such as the coherent left shift cases, coherent right shift cases, etc.

For the two-way shift problem, the network in figure 6.2a) was trained for 100 iterations without settling (as described in the previous section) using the conjugate gradient method, for ten runs starting from different initial conditions. All the lateral connections between output units had their weights initially set to 0.1; the rest of the weights were initialized to random values between -0.5 and 0.5. The network learned equivalent solutions on all ten runs. The joint probability statistics for pairs of output units in different modules on one run are summarized in Table 6.4. Units are numbered as shown in figure 6.2a); the first index refers to the module number, and the second refers to the output unit number within a module. On this particular run, units 1,1 and 2,2 are on together with high probability for coherent left shift cases (p = .79), and units 1,2 and 2,1 are on together with high probability for coherent right shift cases (p = .79). The same pairs of units are on together with low probabilities on negative cases. If we look at how often these pairs of units respond more strongly than other pairs over cases, as shown in Table 6.4, we see the effect even more clearly. The pair of units that prefers left shifts are always the winners on coherent left shift cases, and never on any other cases, and likewise for the pair that prefers right shifts. Thus, the network has perfectly classified the patterns into right and left shifts, with the two output units of a given module each responding to different shifts.

	Coherent left-shift cases	**Coherent right-shift cases**	**Negative cases**
$\langle p_{11}p_{21}\rangle$	0.100703	0.0968108	0.400207
$\langle p_{11}p_{22}\rangle$	0.785733	0.0120876	0.09746
$\langle p_{12}p_{21}\rangle$	0.0129014	0.792191	0.101096
$\langle p_{12}p_{22}\rangle$	0.100663	0.0989109	0.401237

Table 6.1: *The joint probabilities that pairs of units are on, averaged over three different subsets of cases. The units were trained on a simple two-way shift problem. On negative cases, the units in each pair were shown different shifts.*

For the three-way shift problem, the network in figure 6.2b) was trained for 1500 iterations without settling (as described in the previous section) using the conjugate gradient method, for five runs starting from different initial conditions. The output units usually became selective for one or two of the three

	Coherent left-shift cases	**Coherent right-shift cases**	**Negative cases**
$p(unit_{11}\ and\ unit_{21}\ win)$	0	0	0.5
$p(unit_{11}\ and\ unit_{22}\ win)$	1	0	0
$p(unit_{12}\ and\ unit_{21}\ win)$	0	1	0
$p(unit_{12}\ and\ unit_{22}\ win)$	0	0	0.5

Table 6.2: *The proportion of cases in which different pairs of units are the winners, i.e., the two most strongly active units. The proportions are averaged over three different subsets of cases. The units were trained on a simple two-way shift problem. On negative cases, the units in each pair were shown different shifts.*

possible shifts, as shown in Table 6.4.

	Left-shifts	**No-shifts**	**Right-shifts**
Run 1, module 1:			
$p(unit_{11}\ wins)$	0.5	0.25	0
$p(unit_{12}\ wins)$	0.5	0.75	0.25
$p(unit_{13}\ wins)$	0	0	0.75
Run 1, module 2:			
$p(unit_{21}\ wins)$	0	1	1
$p(unit_{22}\ wins)$	0.75	0	0
$p(unit_{23}\ wins)$	0.25	0	0
Run 2, module 1:			
$p(unit_{11}\ wins)$	0	0.75	0.5
$p(unit_{12}\ wins)$	0	0.25	0.5
$p(unit_{13}\ wins)$	1	0	0
Run 2, module 2:			
$p(unit_{21}\ wins)$	0	0.75	0.5
$p(unit_{22}\ wins)$	0	0.25	0.5
$p(unit_{23}\ wins)$	1	0	0

Table 6.3: *The proportion of cases in which each output unit is the winner within its module, for the three different classes of shift patterns. Data are shown for two runs, starting from different initial conditions.*

The joint probabilities of different pairs of units winning for different shift cases are shown in Table 6.4 for the first two of five runs. Units are numbered as shown in figure 6.2b). For this problem, since the network did not develop a unique shift selectivity for each output unit, it could not perfectly correlate the responses of pairs of output units in different modules. Consequently, it could not cleanly separate positive from negative cases. In the first run, there is some shift specialization for pairs of units; units 1,2 and 2,1 are the winners on 3/4 of the coherent no-shift cases, and units 1,3 and 2,1 win on 3/4 of the coherent right-shift cases. In the second run, units 1,3 and 2,3 win on all of the coherent left shift cases and none of the incoherent cases, but the other shifts are divided between different units. We should note that the learning was arbitrarily halted after 1500 iterations, although it had not fully converged. It is possible the learning would have eventually separated the positive and negative cases, although it would clearly require much more training time.

	Coherent left-shifts	**Coherent no-shifts**	**Coherent right-shifts**	**Negative cases**
Run 1:				
$p(unit_{11}\ and\ unit_{21}\ win)$	0	0.25	0	0.208333
$p(unit_{11}\ and\ unit_{22}\ win)$	0.375	0	0	0.03125
$p(unit_{11}\ and\ unit_{23}\ win)$	0.125	0	0	0.0104167
$p(unit_{12}\ and\ unit_{21}\ win)$	0	0.75	0.25	0.333333
$p(unit_{12}\ and\ unit_{22}\ win)$	0.375	0	0	0.125
$p(unit_{12}\ and\ unit_{23}\ win)$	0.125	0	0	0.0416667
$p(unit_{13}\ and\ unit_{21}\ win)$	0	0	0.75	0.125
$p(unit_{13}\ and\ unit_{22}\ win)$	0	0	0	0.09375
$p(unit_{13}\ and\ unit_{23}\ win)$	0	0	0	0.03125
Run 2:				
$p(unit_{11}\ and\ unit_{21}\ win)$	0	0.5625	0.25	0.125
$p(unit_{11}\ and\ unit_{22}\ win)$	0	0.1875	0.25	0.0833333
$p(unit_{11}\ and\ unit_{23}\ win)$	0	0	0	0.208333
$p(unit_{12}\ and\ unit_{21}\ win)$	0	0.1875	0.25	0.0833333
$p(unit_{12}\ and\ unit_{22}\ win)$	0	0.0625	0.25	0.0416667
$p(unit_{12}\ and\ unit_{23}\ win)$	0	0	0	0.125
$p(unit_{13}\ and\ unit_{21}\ win)$	0	0.0	0	0.208333
$p(unit_{13}\ and\ unit_{22}\ win)$	0	0.0	0	0.125
$p(unit_{13}\ and\ unit_{23}\ win)$	1	0	0	0

Table 6.4: *The proportion of cases in which different pairs of units are the winners, i.e., in which they are the two most strongly active units, averaged over four different subsets of cases. The units were trained on a three-way shift problem. On negative cases, the units in each pair were shown different shifts. The data for the two runs are for the same network, starting from different initial random weights.*

6.5 Adding a feed-forward hidden layer

The experiments described up to now have been on very small, simple patterns, using single layer networks. These simple patterns are in fact linearly separable, because we used shift without wrap-around. The shift of such patterns can be computed from the difference of the centres of gravity of the left and right halves of the pattern; this can be done with a single linear operator. A single linear threshold unit can therefore learn to separate left versus right shifts, and a single layer of n softmax'ed units can (at least theoretically) learn to separate patterns with n different shifts. If the pattern is instead shifted with wrap-around, or if the unspecified end-bit of the shifted half of the pattern is filled in with a random value instead of a zero, then nonlinear hidden units are required to detect the shift. The shift problem with wrap-around falls within the class of location-invariant feature detection problems studied by Minsky and Papert, and shown to be not linearly separable (Minsky and Papert, 1987). The shift problem with end-bits randomly specified is as hard as the shift problem with wrap-around, because a subset of these patterns are equivalent to shift patterns with wrap-around. In this section, we describe some preliminary experiments using the same learning procedure on a more difficult shift problem, in DBMs with hidden units.

Adding a hidden layer to a settling network increases the annealing time dramatically, because activity can now recirculate between the output layer and the hidden layer. Eventually the network will settle to a free energy minimum, but this may take a very long time. We can circumvent this problem by using a hybrid architecture, as shown in figure 6.3. The top two layers of the network constitute a DBM. For a given input pattern presentation, the hidden units' states are clamped to a transformed version of that pattern. The transformation is a result of the functions computed by the hidden unit's weights and sigmoidal nonlinearities. So there is no feedback of activity from the output layer to the hidden layer. The weights to the hidden layer are trained by back-propagating the derivatives of the DBM objective function in equation 6.22 (derived in Appendix D).

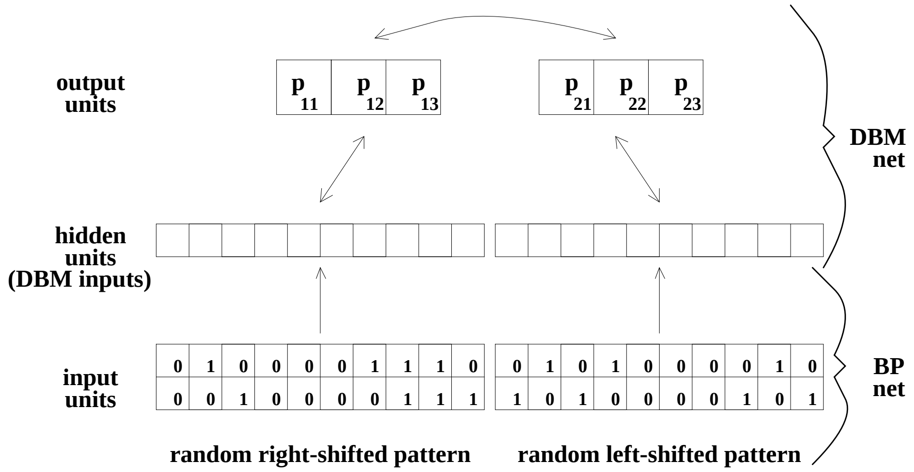

Figure 6.3: *The hybrid multi-layer network architecture used for discovering multiple shifts in random binary patterns, in unsupervised DBMs. The network consists of two modules, each receiving input from a separate part of the network, and each with ten hidden units. The input units are fully connected to the hidden units with feed-forward links in each module. The hidden units are fully connected to the output units with symmetric connections in each module. The output units of each module are also symmetrically connected to the other module's output units. There are no connections between the output units within a module, nor are there connections between the hidden units. The top two layers of the network are trained as a DBM, with the hidden units clamped to a pattern that is determined by the states of the input units and the input → hidden layer connection strengths. The lower layer of the network (labelled BP) is trained by back-propagating the derivatives of the DBM objective function to the input → hidden layer weights.*

We experimented with this learning procedure and the two-module network shown in figure 6.3. The three output units of each module used the "1-of-n" encoding described earlier in this chapter, defined by the state update equation given in 6.23, with the non-settling training procedure. The training patterns consisted of an ensemble of 2400 forty-bit random shift patterns, with three different shifts. As before, each pattern consisted of two sub-patterns, which served as the inputs to the two network modules. The two sub-patterns had either coherent or incoherent shifts. The left half of each twenty-bit sub-pattern was created by taking a random ten-bit vector, each bit flipped on with a probability of 1/3. The right half of the sub-pattern was created by shifting the left half by -1, 0, or 1 bits. The unspecified endpoints of the shifted half were also flipped on with probability 1/3. Complete patterns were created as follows: First, two random ten-bit vectors, P_1 and P_2, were generated, as well as shifted versions of these vectors: $Sh(P_1)$, $Sh(P_2)$, using one of three possible shifts. From these vectors, two positive and two negative patterns were generated, by taking the cross-product of two sets $\{P_1, Sh(P_1)\}$ and $\{P_2, Sh(P_2)\}$. The two positive examples from this cross-product are:

$$\left\{ \begin{matrix} P_1 & P_2 \\ Sh(P_1) & Sh(P_2) \end{matrix} \;,\; \begin{matrix} Sh(P_1) & Sh(P_2) \\ P_1 & P_2 \end{matrix} \right\}$$

and the two negative examples are:

$$\left\{ \begin{matrix} P_1 & Sh(P_2) \\ Sh(P_1) & P_2 \end{matrix} \;,\; \begin{matrix} Sh(P_1) & P_2 \\ P_1 & Sh(P_2) \end{matrix} \right\}$$

Since the same random subpatterns occur in both positive and negative examples for each module, the network should be unlikely to try to model any random structure in the training set, and should therefore be more likely to model the shift.

We trained the network in figure 6.3 on the ensemble of 2400 patterns for 500 conjugate gradient iterations, for two runs starting from different initial conditions. On the both runs, the output units became somewhat tuned to shift. Table 6.5 shows the shift preferences of individual units, as measured by the probability that each output unit is the winner within its module, for each class of shifts. Units nearly always show a preference for one particular shift, although their responses are very mixed. Table 6.5 shows the joint probabilities with which pairs of output units in different modules win on each class of patterns. As with the one-layer network trained on the simpler three-way shift problem, the network has learned a sub-optimal, distributed code for shift.

	Left-shifts	**No-shifts**	**Right-shifts**
Run 1, module 1:			
$p(unit_{11}\ wins)$	0.458333	0.166667	0.458333
$p(unit_{12}\ wins)$	0.5	0.666667	0.416667
$p(unit_{13}\ wins)$	0.0416667	0.166667	0.125
Run 1, module 2:			
$p(unit_{21}\ wins)$	0.333333	0.166667	0.25
$p(unit_{22}\ wins)$	0	0	0
$p(unit_{23}\ wins)$	0.666667	0.833333	0.75
Run 2, module 1:			
$p(unit_{11}\ wins)$	0	0	0
$p(unit_{12}\ wins)$	0.583333	0.25	0.416667
$p(unit_{13}\ wins)$	0.416667	0.75	0.583333
Run 2, module 2:			
$p(unit_{21}\ wins)$	0.5	0.0833333	0.416667
$p(unit_{22}\ wins)$	0.375	0.916667	0.583333
$p(unit_{23}\ wins)$	0.125	0	0

Table 6.5: *The proportion of cases in which each output unit is the winner within its module, for the three different classes of random shift patterns. Data are shown for two runs, starting from different initial conditions.*

	Coherent left-shifts	**Coherent no-shifts**	**Coherent right-shifts**	**Negative cases**
Run 1:				
$p(unit_{11}$ *and* $unit_{21}$ *win*$)$	0.19	0.055	0.2975	0.0208333
$p(unit_{11}$ *and* $unit_{22}$ *win*$)$	0	0	0	0.000833333
$p(unit_{11}$ *and* $unit_{23}$ *win*$)$	0.1225	0.21	0.2675	0.359167
$p(unit_{12}$ *and* $unit_{21}$ *win*$)$	0.02	0.03	0.015	0.181667
$p(unit_{12}$ *and* $unit_{22}$ *win*$)$	0.0025	0.005	0	0.00166667
$p(unit_{12}$ *and* $unit_{23}$ *win*$)$	0.6225	0.53	0.3475	0.340833
$p(unit_{13}$ *and* $unit_{21}$ *win*$)$	0.015	0.035	0.0125	0.0208333
$p(unit_{13}$ *and* $unit_{22}$ *win*$)$	0	0	0	0
$p(unit_{13}$ *and* $unit_{23}$ *win*$)$	0.0275	0.135	0.06	0.0741667
Run 2:				
$p(unit_{11}$ *and* $unit_{21}$ *win*$)$	0	0	0	0
$p(unit_{11}$ *and* $unit_{22}$ *win*$)$	0	0	0	0
$p(unit_{11}$ *and* $unit_{23}$ *win*$)$	0	0	0	0
$p(unit_{12}$ *and* $unit_{21}$ *win*$)$	0.3475	0.045	0.3625	0.0375
$p(unit_{12}$ *and* $unit_{22}$ *win*$)$	0.11	0.24	0.095	0.331667
$p(unit_{12}$ *and* $unit_{23}$ *win*$)$	0.0175	0	0.0025	0.0375
$p(unit_{13}$ *and* $unit_{21}$ *win*$)$	0.04	0.035	0.0775	0.265
$p(unit_{13}$ *and* $unit_{22}$ *win*$)$	0.39	0.68	0.435	0.318333
$p(unit_{13}$ *and* $unit_{23}$ *win*$)$	0.095	0	0.0275	0.01

Table 6.6: *The proportion of cases in which different pairs of units are the winners, i.e., in which they are the two most strongly active units, averaged over four different subsets of cases. The units were trained on a three-way shift problem, with random shift patterns. On negative cases, the units in each pair were shown different shifts. The data for the two runs are for the same network, starting from different initial random weights.*

6.6 Discussion

The learning procedure for unsupervised, deterministic Boltzmann machines described in this chapter attempts to model the coherence in a set of "positive" training patterns relative to a set of incoherent, or "negative" examples. We presented the results of some preliminary experiments with the algorithm, on binary shift patterns. For these patterns, the coherence consisted of a higher order property of the input patterns, namely, the coherence of shift across space. When a multi-layer network was trained on random shift patterns, the output units became only moderately shift-tuned, and the network always learned a sub-optimal solution. The optimal solution would be to make pairs of output units in different modules either perfectly correlated on positive cases and perfectly anti-correlated on negative cases, or vice versa. The only way such a pair of units could become perfectly correlated is if they both responded to the same shift, and no other shift.

One reason the network may have found such poor solutions is the small training set size. Although we used 2400 training patterns, there was much repetition within this training set, in order to create all the cross-over patterns for the negative cases. Thus, increasing the training set size should improve the solution learned by the network. We could also employ some of the same strategies we used to overcome this random sampling problem with the Imax algorithm (see Chapters 3 and 4), such as using multiple modules and enforcing equality constraints between modules. Recall from Chapter 3 that the Imax algorithm performed poorly on the shift problem without the use of equality constraints and large numbers of modules. When we reduced the number of model parameters by using equality constraints, the network was able to become well tuned to shift.

Unfortunately, as we increase the number of modules, the number of negative cross-over patterns grows exponentially. The same is true as we increase the number of possible shifts in the data set. In order to apply the procedure to more natural patterns, such as stereo images containing real-valued disparities, we need a better method of generating negative cases.

For a network that must settle to equilibrium upon each pattern presentation, the training time is rather long, even when using the mean field approximation. We have explored several ways of speeding the simulations in practice. The first is to eliminate the settling, and use the full non-equilibrium derivatives of the free energies in computing the weight updates. The second is to use a hybrid network with a feed-forward hidden layer, the weights of which can be trained by back-propagating derivatives of the DBM objective function. Finally, we can use a conjugate gradient optimization procedure to implement the learning algorithm. Although these speedups force us to throw away most of the desirable properties of the Boltzmann machine that we sought to begin with (i.e., a local Hebbian learning rule, recurrent activity flow, no back-propagation of derivatives), they are merely convenient tools for accelerating the learning. In principle, we could solve the same problems described in this chapter using a multi-layer, fully recurrent, DBM with settling.

An appealing aspect of using a network with settling is that it can use the lateral connections to perform interpolation across space. For example, a set of modules tuned to stereo disparity, each receiving input from different regions of an image, could represent a dense depth map even when presented with sparse data. For regions with missing data, the activities of the output units of a module would be largely driven by their lateral connections, so they would tend to adopt the same states as their similarly shift-tuned neighbors. After iterative relaxation, the network should be able to form a more globally consistent interpretation of the depths in the scene.

Chapter 7

General discussion

Our major goal in developing unsupervised learning procedures, as set out in Chapter 2, is to build self-organizing network modules which capture important regularities of the environment in a simple form suitable for further perceptual processing. We would like an unsupervised learning procedure to be applicable at successive layers, so that it can extract and explicitly represent progressively higher order features. If at one stage the algorithm could learn to explicitly represent continuous real-valued parameters such as relative depth, position and orientation of features in an image, subsequent learning could then discover higher order relations between these features, representing, for example, the location of object boundaries.

Previous approaches in unsupervised learning, such as clustering, principal-component-analyzing, and information-transmission-based methods, have several features in common. They make minimal assumptions about the kind of structure in the environment, and they are good at discovering low order structure such as correlations in the raw input. These methods can be useful for preprocessing raw signal data in order to reduce the dimensionality, improve the signal to noise ratio, and decorrelate the input components. Generally, these methods try to model *all* of the structure in the environment in a single processing stage.

The approach taken in this thesis is novel, in that our unsupervised learning algorithms do not try to preserve all of the information in the signal. Rather, we start by making strongly constraining assumptions about the kind of structure of interest in the environment. We then proceed to design learning algorithms which will discover precisely that structure. By constraining what kind of structure will be extracted by the network, we can force the network to discover higher level, more abstract features. Additionally, the constraining assumptions we make can provide a way of decomposing difficult learning problems into multiple simpler feature-extraction stages. This general approach appears to be a promising way of speeding up difficult problems such as object and speech recognition from raw sensory data. Once a good representation of the environment has been learned in an unsupervised manner, a supervised learning procedure can be applied to solve goal-directed problems such as the identification of particular objects, speech sounds, etc.

In the following sections, we evaluate the specific algorithms described in this thesis with respect to the above goals. In particular, we consider the sort of representations these algorithms can learn, their biological plausibility, their ability to model multiple stages of learning, and their applicability to real data, and real-world sensory processing problems such as the fusion of natural stereo images. We close with some suggestions for future work, and a summary of the main contributions of the thesis.

7.1 Representations

One desirable feature of an unsupervised learning procedure is that it transforms the input into a representation which is somehow simpler, so that subsequent processing can be done more easily. One notion of simplicity in a representation is that it teases apart, and explicitly encodes important underlying parameters of the signal. The algorithms presented in this thesis have tried to explicitly represent a single parameter which is spatially coherent in the visual world, such as depth in stereo images. We have explored a number of ways of representing a single, real-valued parameter, including interval codes (Chapter 3, n-valued Imax), value codes (Chapter 4, continuous Imax), and population codes (for Imax in Chapter 5, and deterministic Boltzmann machines in Chapter 6). The population code is an efficient way of transmitting a wide range of possible values of a signal using processing units with only a limited dynamic range of responses. It also has a much richer representational capacity, since some degree of uncertainty can be inferred from a population response.

Ideally, we would like to extract more than just one parameter. The Imax algorithm is based on pairwise mutual information maximization between parameters extracted from different parts of the sensory input. One way to generalize the continuous version of Imax to extract multiple parameters was described at the end of Chapter 4. This method maximizes the following objective function:

$$I_{y_a+y_b;signal} = 0.5 \log \frac{|Q_{y_a+y_b}|}{|Q_{y_a-y_b}|} \tag{7.1}$$

where y_a and y_b are parameter vectors extracted from neighboring patches, Q is a covariance matrix and $|Q|$ its determinant. Zemel and Hinton (1991) have applied this method to the problem of learning to represent the viewing parameters of two-dimensional objects. This method tries to extract multiple features from an image patch which are *uncorrelated* with each other, as well as being good predictors of the feature vector extracted from a neighboring patch. The method is potentially more powerful than linear methods such as principal components analysis, because the network can compute arbitrary nonlinear transformations in order to extract these features. One difficulty with the method is the practical limitations of computing determinants of ill-conditioned matrices, as mentioned in Chapter 4. Another drawback is that the representation learned by the algorithm, at least on the "viewing parameters" problem, is not all that simple. The features the network learns to represent are typically each nonlinear combinations of the viewing parameters (e.g., scale, location, and size), which cannot easily be interpreted by themselves.

An alternative goal is to try to extract features that are statistically independent. The general case of Imax between pairs of feature vectors would achieve such a code. However, it would be far too computationally expensive to implement in practice. One of the major unsolved problems in research on unsupervised learning is to find an efficient way to discover approximate solutions to this problem.

7.2 Biological plausibility

There are two major reasons why we might want to study learning procedures which are biologically realizable. First, by focusing our efforts on models which are consistent with our current state of knowledge of biological neural processing, we hope to develop algorithms which lead to a better understanding of human thought processes. Second, by incorporating brain-like computational processes and neuroanatomical constraints, we hope to develop efficient solutions to difficult perceptual processing problems.

We discussed some of the more biologically implausible details of the Imax algorithm at the beginning of Chapter 6. These included the requirement that units be able to communicate their outputs to each other without actually being connected to each other, the need to back-propagate derivatives to train hidden layers of the network, and the necessary requirement that different modules receive input from *non-overlapping* parts of the input. This led us to propose an alternative way of extracting spatially coherent signals, using deterministic Boltzmann machines (DBMs), with the "contrastive clamping" training procedure. This DBM learning procedure is based on a local, Hebb-like learning rule, and allows the network to have arbitrary connectivity, including recurrent links between the output units. Further, different modules could have overlapping or even identical receptive fields; for the examples presented in Chapter 6, modules would still have to extract disparity in order to model the difference between positive and negative cases. However, this learning procedure has the drawback that it requires one bit of supervision: the network must be told, for each pattern it sees, whether the example is a positive or negative case. Also, the procedure does not produce very finely depth-tuned output units for the architecture we used. It seems that the learning is very sensitive to random structure in the training set, as is Imax. Therefore, the addition of multiple network modules and equality constraints between modules should improve the DBM network's ability to model shift (as was the case for Imax).

Another implementation detail which is critical in the random dot stereogram problem is the use of equality constraints. We used equality constraints between corresponding weights of units in different modules to force each module to compute exactly the same function. This greatly reduces the number of free parameters in the network, and prevents it from overfitting a small number of cases in the training set. Given infinitely many training patterns, these constraints would presumably be unnecessary. For simulation purposes, they are a good way to speed the learning and improve the solution for small training set sizes.

7.3 Multiple learning stages

One of the "holy grails" of unsupervised learning research is a general self-organizing principle which could operate in multiple stages, to discover many different kinds of structure. We have shown how the Imax learning principle could be applied in several stages, by employing different architectural constraints at each stage. In Chapter 4, we first showed how simple modules could learn to extract depth locally from small image patches. Then a higher layer of units, receiving input from larger spatial regions, tried to predict the depth extracted by one local module by interpolating from the depths extracted by several neighboring modules. We could potentially keep applying the same principle at higher layers, using progressively larger spatial extents. However, the spatial coherence of simple image properties such as depth, colour, texture, etc. is more likely to break down as the spatial extent increases. So for larger image regions, it would be better to try to extract some more global spatially coherent feature, such as an object's orientation, or the location of its boundaries. We showed how to extract local discontinuities in depth using the mixture model of coherence described in Chapter 5. Given these locations of depth edges, we could apply another layer of spatial-coherence modeling units to try to predict the more global coherence of depth edges across space.

A promising general approach to multi-level unsupervised feature extraction, illustrated nicely by the stereo example described above, is to attempt to first model low-level coherence at early stages. This can be done by limiting the search to parts of the input which are narrowly localized in time and across space.

At higher levels, the system can try to make predictions over larger spatio-temporal extents, and to model discontinuities in the predictions of earlier processing stages. These general design principles could be applied to a variety of learning procedures, including Imax and the contrastive clamping procedure for DBMs.

7.4 Applying Imax to real images

In Chapter 4, we presented a special case of the Imax algorithm for continuous signals, which was derived by making Gaussian assumptions about the underlying spatially coherent features, as well as the noise. The method was shown to work well for the problem of extracting relative depth from random dot stereograms of curved surfaces. However, the model is by no means a complete model of human stereopsis, or a complete solution to the machine stereo vision problem, as it ignores many of the complexities found in natural stereo images. These include the following: 1) noisy and missing input features, 2) depth discontinuities, 3) multi-scale ambiguities, 4) lower order spatially coherent structure, and 5) monocular local and global depth cues. We discuss these problems below, and in some cases, discuss ways of extending the algorithm and/or network architecture to handle these problems.

Although this discussion focuses primarily on the stereo application, it addresses many of the general problems associated with applying the Imax algorithm to real images, regardless of the features of interest. If we can deal with the above complexities, we can apply the Imax algorithm to real images in order to extract many other features besides stereo disparity.

7.4.1 Noisy and missing input

Real world scenes often contain regions lacking any easily detectable features. Some regions may be devoid of any sharp intensity changes and therefore provide no stereo disparity information. Further, since no imaging system is perfect, features will be corrupted by noise. Our experiments with random dot patterns of varying density show that Imax is sensitive to the dot density. The network fails to learn disparity when presented with extremely sparse patterns. On these patterns, there is less information about disparity, so the network tends to learn highly suboptimal solutions.

There are a number of ways to cope with this problem. One is to use a much larger data set, to provide better statistics about disparity. Another possibility is to use a more robust objective function, which discounts cases that provide no information. One such objective function was presented in Chapter 5, based on a mixture model of coherence. This method is successful at identifying and throwing out cases of discontinuities, but only if we start from good initial conditions in which the network is already somewhat tuned to a spatially coherent feature.

The problem of noisy or missing data can be dealt with in another way, by using more global information to fill in missing features. A Boltzmann machine can accomplish this by using feedback connections, as described in Chapter 6. We showed one way to train a Boltzmann machine in an unsupervised manner, using the "contrastive clamping procedure" to force it to learn spatially coherent features. Although the algorithm was only moderately successful at solving the shift problem, as mentioned above, we expect that with the addition of further model constraints the learning procedure would give much better performance.

7.4.2 Depth discontinuities

In Chapter 5, we explored several ways of extending the continuous Imax algorithm to deal with depth discontinuities. One approach was to form a mixture model of the spatially coherent signal. This enabled a layer of depth-extracting units to form a population code for depth. A large population of disparity-tuned of units with different depth preferences and slightly different spatially localized receptive fields should be able to signal the presence of a discontinuity very clearly, because two subgroups of units with very different depth preferences would become active. Our results on the population code model were rather preliminary, and applied only to very small populations of four or five units. Further experiments must be done to test larger scale versions of this model, and evaluate its performance on data sets with discontinuities.

We also applied the idea of a mixture model of coherence to the depth interpolating architecture used with the Imax algorithm. By adding a set of extra "gating" units to compute the relative model likelihoods on each case, a set of competing interpolators was able to learn multiple interpolating models of depth, for different discontinuity locations. Additionally, the gating units learned to explicitly model the locations of discontinuities.

7.4.3 Multi-scale ambiguities

Another complexity of the stereo vision problem is the fact that natural scenes contain information at a variety of spatial scales. High spatial frequency information, such as in the texture of objects, may be useful for fine discriminations of (relative) depth. But high frequency features can give rise to ambiguities in disparity. At a given spatial frequency, it is only possible to detect the *relative phase* between two image patches, rather than the relative disparity. Fortunately, it is often possible to resolve ambiguities by combining information at a variety of spatial scales. If we presented our network with input features at a variety of spatial frequencies (e.g., by pre-filtering natural images using a set of band-pass filters), it should be able to learn how to integrate information at multiple scales in order to make optimal predictions about disparity.

7.4.4 Lower order spatially coherent structure

Our networks were able to learn to extract depth, a higher order feature of the image intensities, because it was the *only* spatially coherent feature of the images. In natural images, many other features, including shading, texture, colour, and orientation, are coherent across space. When applied to natural images, our algorithm would be expected to discover the simplest feature first; that would most certainly be the average intensity. To model multiple features in natural images, we could apply a multi-stage process which extracts different features at each stage, or we could apply a learning procedure that simultaneously extracts multiple features. The multi-stage approach is a natural way to separate low-level feature extraction from the problem of extracting disparity. Disparity extraction could be deferred to a later processing stage by first applying Imax to monocular images. Separate preprocessing networks could attempt to model the spatially coherent structure in the left and right views individually. The output of these low level networks could then be combined to extract stereo features.

7.4.5 Global constraints, and other depth cues

In addition to stereo disparity, animal visual systems appear to use a variety of monocular cues to infer depth, including relatively local cues such as shading and motion parallax, and more global cues such as occlusion and relative object sizes on the retina. In order to use global cues like occlusion, at least partial segmentation of the scene into parts must first be accomplished. Coherence of stereo disparity is likely a potent cue for segmenting parts of a scene, but it requires lateral communication between feature detectors at different spatial locations. Another extension of the Imax model, left for future work, is to add recurrent lateral connections between output units in different modules, so that the network can iteratively settle on a good global interpretation. The disparity predicted by each unit would depend not only on locally computed information, but also on the activity in the rest of the network. Neighbors which detect similar depths should encourage each other to respond, while neighbors with different depth preferences should inhibit each other. This would allow units with ambiguous inputs to sharpen their responses based on more global information. On the other hand, in the case of a discontinuity, conflicting global information should cause units' responses to be indecisive. The whole network should settle into a more globally consistent interpretation of a depth map, with high entropy regions indicating discontinuities.

7.5 More directions for future work

There is much room for further work on unsupervised learning. We have begun experimenting with a new learning procedure, based on the principle of discovering perceptual invariants, which we outline below. We then describe two other major topics for future research.

7.5.1 Learning perceptual invariants

The goal of our depth interpolating units in Chapter 4 was to discover how to predict one value from a linear combination of nearby values, by maximizing the log ratio of the variance of the signal plus predicted signal to the variance of the prediction error. A more general way to pose this problem is to find an invariant relation between the predicted value and the other values by learning a linear combination of *all* the values, such that the combined value always equals zero (Richard Durbin, *personal communication*). In this case, we would minimize the variance of the predicting unit's output about zero, while maximizing the variances of the individual inputs. This amounts to discovering invariant higher-order properties by learning invariance detectors that have low variance even though their input lines have high variances and large weights. One attractive aspect of this view is that the actual output of an invariance detector would represent the extent to which the current input violates the network's model of the regularities in the world. This is an efficient way of transmitting information about the current input. Furthermore, it has been suggested that the computation of invariant features about the world plays a fundamental role in human pattern recognition (Dodwell, 1983).

Several algorithms for learning invariant features of the input have previously been proposed. Uttley's Informon learning procedure (Uttley, 1970) tried to minimize the mutual information, for each connection, between the activity of its input and output units with respect to its weight. Kohonen and Oja (1976) proposed a learning algorithm for a single unit which acts as a "novelty detector", by responding best to patterns which are orthogonal to the principal subspace of the input distribution. Fallside (1989) proposed a learning procedure which implements a linear prediction filter. A unit receives input representing the values of a signal at several time frames, and tries to make its output zero by computing the sum of the signal at the current time slice and a linear combination of the signal values at previous time slices. Atick and Redlich (1989) proposed an equivalent learning procedure for a spatial predicting unit, which modelled the development of retinal ganglion cell kernels. Their cost function combined a redundancy term with an information loss term. In the linear case, this leads to a learning rule which minimizes the output variance of a unit by computing the error in predicting the central pixel of its receptive field by a linear combination of nearby pixel values. This latter model is equivalent to our Imax interpolating unit, for the special case where there are no hidden units.

An invariance detector that minimizes the ratio of its output variance divided by the variance that would be expected if the input lines were independent Gaussian variables is a continuous generalization of the G-Maximization learning procedure (Pearlmutter and Hinton, 1986). The goal of the Gmax algorithm is to discover *binary* features by maximizing the difference between the actual probability distribution of a unit's output and the distribution one would expect to see if its input lines were independent. If we generalize Gmax to the *continuous* case, by assuming the input signals are Gaussian, the Gmax objective function leads to two possible solutions, either maximizing or minimizing the variance of the output of a linear unit, normalized by the sum of the activation variances of the unit's input lines (see Appendix E for the derivation). The former objective is equivalent to Linsker's Infomax principle (Linsker, 1988), for the special case he analyzes in which the variances of the input lines are fixed. In this case, Linsker's Infomax principle (of maximizing the information a unit conveys about its input) leads to maximization of the the output variance of a linear unit divided by the sum of its squared weights.

If we instead choose the latter solution of continuous Gmax, to minimize the ratio of the output variance to the sum of input variances, we have a procedure that causes a unit to try to make its total input constant. It therefore tries to discover some invariant combination of its input lines. At the same time, the unit tries to have large variances (hence, large weights) on each individual input line. Since the objective forces the weights to remain large, a unit cannot find trivial invariants, e.g., by setting all the weights to zero.

We can generalize this algorithm to apply to a set of competing units, which learn to detect different invariant properties of the input patterns by forming a mixture model of the input distribution (see Appendix E). Preliminary simulations show this method can learn to detect disparity in random dot stereograms; the competing units each learn to become tuned to a different disparity. This latter model is much more biologically plausible than the spatial coherence model, since it does not require information to be exchanged between adjacent modules, and can produce moderate disparity-tuning in a single layer of competing units.

There are many temporally invariant features of the sensory input. The acoustic signal generated by a particular speaker is relatively constant in overall pitch, timbre etc. As one moves about in the world, the visual field flows by in characteristic patterns of expansion, dilation and translation. And independently moving rigid objects are invariant with respect to shape, texture and many other features, up to very high level properties such as the object's identity. A learning procedure which could discover this variety of perceptual constancies would be very exciting indeed.

7.5.2 Multi-sensory integration

There are many other forms of coherence in sensory patterns which our perceptual mechanisms rely on. Visual, acoustic and tactile input often provide us with multiple sources of information about the same underlying causes. For example, our tactile and visual senses give us much the same information about objects in the world, such as the texture, orientation, and boundaries of surfaces. We can combine these different sensory channels to disambiguate events in the world, or rely on one source when another is impoverished. One approach to the problem of integrating multi-sensory information is to use a supervised learning procedure (e.g., back-propagation) in a network which combines multiple sources of information to solve a classification task. Yuhas and colleagues (1990) have applied this idea to a network that learns to recognize vowel sounds from combined visual and acoustic input.

However, the mapping between two different information sources may be arbitrarily complex, so that the job of nonlinearly combining the sources in trying to solve an already difficult classification task (such as speech recognition) is highly non-trivial. Yuhas et al. circumvented this problem by pre-training the network to associate visual images of the speaker's mouth with acoustic features of the vocal tract, before combining the visual and acoustic inputs. Another possibility is to use unsupervised learning methods to transform *both* the acoustic and visual signals into a common set of features. If we could learn to transform these multiple information sources into a more compact representation, consisting of a small number of parameters that express the mutual information between the two sources, it should then make subsequent supervised learning much easier.

7.5.3 Temporal coherence

In addition to coherence across space and sensory modalities, a significant source of structure in the world is the coherence of sensory information across time. The Imax learning procedure allows pairs of network modules having adjacent receptive fields to discover spatially coherent features in visual images. The same idea could be applied to a single network module presented with temporally coherent input sequences, to learn features that are predictive across several time steps. For example, in image sequences of simple two-dimensional rigid shapes moving in random trajectories in the 2-D plane, one of the simplest features to learn, in order to predict the next time step, is the direction of motion. Over longer time scales, as the direction keeps changing, the optimal feature for the network to learn is the identity of the object. A multi-layer network should be able to discover features such as edges moving in particular directions in the first hidden layer, and object categories in the output layer.

7.6 Conclusions

In this thesis, we have proposed a novel design principle for unsupervised learning procedures. We start by making constraining assumptions about what kind of structure in the world is perceptually relevant. We then derive neural network learning algorithms and architectures that embody these constraints.

We have proposed an information-theoretic unsupervised learning algorithm called Imax, based on this design principle. This algorithm uses the assumption that there is spatial coherence in the visual world, to try to discover spatially coherent image features. This assumption is strongly constraining, but widely applicable to perceptual tasks. To apply the Imax algorithm to images, the assumption of spatial coherence is embodied in both the architecture and objective function. The architecture is constrained so that separate modules receive input from physically separate sensors. The objective function can be formulated in several ways, depending on whether one chooses a discrete or continuous model. In the former case, the objective is to maximize the mutual information between parameters extracted by different modules. In the latter, it is to maximize the information that these parameters convey about some presumed common underlying signal, which is assumed to be Gaussian.

We illustrated the discrete version of Imax (Chapter 3) on two problems: (i) classifying simple sinusoidal intensity patterns of different spatial frequencies and phases, and (ii) classifying binary random shift patterns. This discrete algorithm is best suited to modelling properties of the input which can be grouped into separate classes, as opposed to modelling real-valued features. In Chapter 4, the continuous version of Imax was illustrated on the problem of detecting stereo disparity in a restricted class of stereo images, using random dot stereograms. We also applied Imax in a second stage of learning, using the same principle to interpolate depth across larger spatial extents. Extensions to mixture models of coherence, applied to the continuous case (Chapter 5), allowed the network to learn population codes for depth, multiple surface interpolators for images with discontinuities, and locations of discontinuities.

We also proposed two alternative classes of algorithms for discovering spatially coherent features. One was based on a novel, unsupervised training procedure for deterministic Boltzmann machines (Chapter 6). This was applied to a restricted ensemble of binary shift patterns consisting of examples labelled as either spatially coherent or incoherent with respect to shift. The second class of algorithms, proposed in an earlier section of this chapter, was based on the idea of discovering spatially *invariant* features in the visual world. Both of these algorithms show promise, as more biologically plausible procedures for learning based on spatial coherence.

One conclusion we can draw from all of our experiments is that it is difficult for an unsupervised learning procedure to extract high-level features from noisy data which may also contain lower level regularities, even when the extraction of the high-level features ultimately leads to more globally optimal solutions. The use of equality constraints to reduce the number of free parameters in the network can greatly improve the chance of discovering more globally optimal solutions. Training set size is obviously another important factor in reducing sampling error. However, it is usually not possible to simulate unsupervised learning in large networks (i.e., for large input dimensionality) on adequately large training sets, due to computing resource limitations, so heuristics such as equality constraints are a more feasible solution.

The problem of multiple levels of coherent structure is a major barrier to applying the algorithms presented here to natural images. A promising way to deal with the problem is to use more stages of learning. Early stages can try to extract lower order structure from parts of the input which are narrowly restricted both spatially and temporally. Later stages can combine information across larger spatial and temporal extents. This information could be in the form of lower order spatially coherent features, as well as spatial or temporal discontinuities in those features.

The learning procedure we have described builds into the objective function (and the architecture) a type of prior knowledge that is strongly constraining, but widely applicable to perceptual tasks. We have applied the idea to the problem of learning spatially coherent features in visual images, but the same idea could be applied to learning coherent features across different sensory modalities, and across time.

Our simulations demonstrate the general utility of the Imax algorithm in discovering interesting, non-trivial structure (disparity and depth discontinuities) in artificial stereo images. This is the first attempt we know of to model perceptual learning beyond the earliest stages of low-level feature-extraction, and to model multiple stages of unsupervised learning.

Appendix A
The learning equations for discrete Imax

This appendix is divided into two sections. In this first section we derive the learning equations for Discrete Imax in the binary case. The second section shows similar derivations for the n-valued discrete case.

A.1 Discrete Imax for binary variables

We first consider the case where a spatially coherent parameter is represented as a binary variable. We want to maximize the mutual information between samples of this variable, y_i and y_j, extracted from neighboring image patches. We further assume that the network module assigned to each input patch is connected in a strictly feed-forward manner, it may have zero or more hidden layers, and it has a single output unit.

Each output unit computes a nonlinear probabilistic function of its real-valued total input $x_i = \sum_k w_{ik} y_k$, using the sigmoidal nonlinearity:

$$f(x) = \frac{1}{1+e^{-x}}$$

For a particular input case α, the output of the ith unit represents a binary variable $y_i \in \{0, 1\}$, which is 1 with probability

$$p_i^\alpha = f(x_i^\alpha)$$

However, rather than using stochastic output units and sampling each y_i to estimate the p_is, we use the exact values of the p_is as the outputs. The hidden units' outputs are deterministic real-valued variables, and use the same non-linearity as the output units in computing their outputs: $y_j = f(x_j)$.

We can estimate the overall probability of the events $y_i = 1$ and $y_i = 0$ for the ith output unit by averaging over the input sample distribution:

$$\begin{aligned} p_i &= \sum_{\alpha=1}^{N} P^\alpha p_i^\alpha \\ p_{\overline{i}} &= 1 - p_i \end{aligned}$$

where N is the number of input samples and P^α is the prior probability of an input case α. We treat every case in the training set as equi-probable, so $P^\alpha = \frac{1}{N}, \forall \alpha$.

Similarly, for a pair of binary units in a feed-forward network that are not connected to one another (but may share the same inputs), and hence whose outputs are independent given a particular case, we can compute the four possible values in the joint (discrete) probability distribution of the binary variables y_i and y_j represented by their outputs p_i and p_j, as follows:

$$\begin{aligned} p_{ij} &= \sum_\alpha P^\alpha p_i^\alpha p_j^\alpha \\ p_{\bar{i}j} &= p_j - p_{ij} \\ p_{i\bar{j}} &= p_i - p_{ij} \\ p_{\bar{i}\bar{j}} &= 1 - p_j - p_i + p_{ij} \end{aligned}$$

The partial derivatives of the (expected) individual and joint probabilities with respect to the expected output of the ith unit on case α are:

$$\begin{aligned} \frac{\partial p_i}{\partial p_i^\alpha} &= P^\alpha \\ \frac{\partial p_{\bar{i}}}{\partial p_{i}^\alpha} &= -P^\alpha \\ \frac{\partial p_{ij}}{\partial p_i^\alpha} &= P^\alpha p_j^\alpha \\ \frac{\partial p_{\bar{i}j}}{\partial p_i^\alpha} &= -P^\alpha p_j^\alpha \\ \frac{\partial p_{i\bar{j}}}{\partial p_i^\alpha} &= P^\alpha (1 - p_j^\alpha) \\ \frac{\partial p_{\bar{i}\bar{j}}}{\partial p_i^\alpha} &= -P^\alpha (1 - p_j^\alpha) \end{aligned}$$

The amount of information transmitted by the ith unit on case α when it is on is:

$$I(y_i^\alpha = 1) = -\log p_i^\alpha$$

If the log is base 2, the units of information are bits; for the natural log, the units are nats. From here on, we use log to denote the natural log. When the value of y_i^α is unknown, the average information (or equivalently the entropy or uncertainty) in y_i^α is:

$$H(y_i^\alpha) = -[p_i^\alpha \log p_i^\alpha + p_{\bar{i}}^\alpha \log p_{\bar{i}}^\alpha]$$

Averaged over all input cases, the entropy of y_i is:

$$H(y_i) = -\langle \log p(y_i) \rangle = -[p_i \log p_i + p_{\bar{i}} \log p_{\bar{i}}]$$

The mutual information between the outputs of the ith and jth units, y_i and y_j, is:

$$I(y_i; y_j) = H(y_i) + H(y_j) - H(y_i, y_j)$$

where $H(y_i, y_j)$ is entropy of the joint distribution $p(y_i, y_j)$:

$$\begin{aligned} H(y_i, y_j) &= -\langle \log p(y_i, y_j) \rangle \\ &= -[p_{ij} \log p_{ij} + p_{i\bar{j}} \log p_{i\bar{j}} + p_{\bar{i}j} \log p_{\bar{i}j} + p_{\bar{i}\bar{j}} \log p_{\bar{i}\bar{j}}] \end{aligned}$$

The partial derivative of $I(y_i;y_j)$ with respect to the expected output of unit i on case α, p_i^α, can now be computed; since $H(y_j)$ does not depend on p_i^α, we need only differentiate $H(y_i)$ and $H(y_i,y_j)$.

$$
\begin{aligned}
\frac{\partial H(y_i)}{\partial p_i^\alpha} &= \frac{\partial}{\partial p_i^\alpha}[-(p_i \log p_i + p_{\bar{i}} \log p_{\bar{i}})] \\
&= -[\frac{\partial p_i}{\partial p_i^\alpha} \log p_i + \frac{p_i}{p_i}\frac{\partial p_i}{\partial p_i^\alpha} + \frac{\partial p_{\bar{i}}}{\partial p_i^\alpha} \log p_{\bar{i}} + \frac{p_{\bar{i}}}{p_{\bar{i}}}\frac{\partial p_{\bar{i}}}{\partial p_i^\alpha}] \\
&= -[\frac{\partial p_i}{\partial p_i^\alpha}(\log p_i + 1) + \frac{\partial p_{\bar{i}}}{\partial p_i^\alpha}(\log p_{\bar{i}} + 1)] \\
&= -[P^\alpha(\log p_i + 1) - P^\alpha(\log p_{\bar{i}} + 1)] \\
&= -P^\alpha \log \frac{p_i}{p_{\bar{i}}}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial H(y_i,y_j)}{\partial p_i^\alpha} &= \frac{\partial}{\partial p_i^\alpha}[-(p_{ij} \log p_{ij} + p_{i\bar{j}} \log p_{i\bar{j}} + p_{\bar{i}j} \log p_{\bar{i}j} + p_{\bar{i}\bar{j}} \log p_{\bar{i}\bar{j}})] \\
&= -\left[\frac{\partial p_{ij}}{\partial p_i^\alpha} \log p_{ij} + \frac{p_{ij}}{p_{ij}}\frac{\partial p_{ij}}{\partial p_i^\alpha} + \frac{\partial p_{\bar{i}j}}{\partial p_i^\alpha} \log p_{\bar{i}j} + \frac{p_{\bar{i}j}}{p_{\bar{i}j}}\frac{\partial p_{\bar{i}j}}{\partial p_i^\alpha}\right. \\
&\quad \left. + \frac{\partial p_{i\bar{j}}}{\partial p_i^\alpha} \log p_{i\bar{j}} + \frac{p_{i\bar{j}}}{p_{i\bar{j}}}\frac{\partial p_{i\bar{j}}}{\partial p_i^\alpha} + \frac{\partial p_{\bar{i}\bar{j}}}{\partial p_i^\alpha} \log p_{\bar{i}\bar{j}} + \frac{p_{\bar{i}\bar{j}}}{p_{\bar{i}\bar{j}}}\frac{\partial p_{\bar{i}\bar{j}}}{\partial p_i^\alpha}\right] \\
&= -\left[\frac{\partial p_{ij}}{\partial p_i^\alpha}(\log p_{ij} + 1) + \frac{\partial p_{\bar{i}j}}{\partial p_i^\alpha}(\log p_{\bar{i}j} + 1)\right. \\
&\quad \left. + \frac{\partial p_{i\bar{j}}}{\partial p_i^\alpha}(\log p_{i\bar{j}} + 1) + \frac{\partial p_{\bar{i}\bar{j}}}{\partial p_i^\alpha}(\log p_{\bar{i}\bar{j}} + 1)\right] \\
&= -\left[P^\alpha p_j^\alpha(\log p_{ij} + 1) - P^\alpha p_j^\alpha(\log p_{\bar{i}j} + 1)\right. \\
&\quad \left. + P^\alpha(1 - p_j^\alpha)(\log p_{i\bar{j}} + 1) - P^\alpha(1 - p_j^\alpha)(\log p_{\bar{i}\bar{j}} + 1)\right] \\
&= -P^\alpha[p_j^\alpha \log \frac{p_{ij}}{p_{\bar{i}j}} + p_{\bar{j}}^\alpha \log \frac{p_{i\bar{j}}}{p_{\bar{i}\bar{j}}}]
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial I(y_i;y_j)}{\partial p_i^\alpha} &= \frac{\partial H(y_i)}{\partial p_i^\alpha} - \frac{\partial H(y_i,y_j)}{\partial p_i^\alpha} \\
&= -P^\alpha\left[\log \frac{p_i}{p_{\bar{i}}} - p_j^\alpha \log \frac{p_{ij}}{p_{\bar{i}j}} - p_{\bar{j}}^\alpha \log \frac{p_{i\bar{j}}}{p_{\bar{i}\bar{j}}}\right]
\end{aligned}
$$

A.1.1 Differentiating I with respect to the weights by back-propagation

The partial derivative of $I(y_i;y_j)$ with respect to an incoming weight to the ith output unit is computed as follows:

$$
\begin{aligned}
\frac{\partial I(y_i;y_j)}{\partial w_{ik}} &= \sum_\alpha \frac{\partial I(y_i;y_j)}{\partial p_i^\alpha}\frac{\partial p_i^\alpha}{\partial x_i^\alpha}\frac{\partial x_i^\alpha}{\partial w_{ik}} \\
&= \sum_\alpha \frac{\partial I(y_i;y_j)}{\partial p_i^\alpha} f'(x_i^\alpha) y_k^\alpha
\end{aligned}
$$

The derivative of the sigmoid function is:

$$f'(x_j^\alpha) = f(x_j^\alpha)(1 - f(x_j^\alpha))$$

The partial derivatives of I with respect the hidden units' weights are computed in the same manner as in the back-propagation learning algorithm (Rumelhart, Hinton and Williams, 1986). Using the chain rule, the gradients for a given layer are computed from those of the layer above. We start at the top layer, where each unit computes the derivative of the objective function, $I(y_i; y_j)$, with respect to its total input:

$$\frac{\partial I(y_i;y_j)}{\partial x_j^\alpha} = \frac{\partial I(y_i;y_j)}{\partial p_j^\alpha}\frac{\partial p_j^\alpha}{\partial x_j^\alpha} = \frac{\partial I(y_i;y_j)}{\partial p_j^\alpha} f'(x_j^\alpha)$$

Then, for each remaining layer L, traversing layers in reverse order (from the 2nd last to the input layer), each unit k in layer L computes the partial derivative of I with respect to its total input, $x_{k,L}$, using a weighted sum of the partials computed by the layer above, $\frac{\partial I}{\partial x_{j,L+1}}$:

$$\begin{aligned}
\frac{\partial I(y_i;y_j)}{\partial x_{k,L}^\alpha} &= \sum_i \frac{\partial I(y_i;y_j)}{\partial x_{i,L+1}^\alpha}\frac{\partial x_{i,L+1}^\alpha}{\partial x_{k,L}^\alpha} \\
&= \sum_i \frac{\partial I(y_i;y_j)}{\partial x_{i,L+1}^\alpha}\frac{\partial x_{i,L+1}^\alpha}{\partial y_{k,L}^\alpha}\frac{dy_{k,L}^\alpha}{dx_{k,L}^\alpha} \\
&= \sum_i \frac{\partial I}{\partial x_{i,L+1}^\alpha} w_{ik}\, f'(x_{k,L}^\alpha)
\end{aligned}$$

The final weight update is computed by multiplying by the above by the presynaptic input, and accumulating these terms over cases:

$$\begin{aligned}
\frac{\partial I(y_i;y_j)}{\partial w_{km}} &= \sum_\alpha \frac{\partial I(y_i;y_j)}{\partial x_{k,L}^\alpha}\frac{\partial x_{k,L}^\alpha}{\partial w_{km}} \\
&= \sum_\alpha \frac{\partial I(y_i;y_j)}{\partial x_{k,L}^\alpha} y_{m,L-1}^\alpha
\end{aligned}$$

A.2 Discrete Imax for n-valued variables

We now consider the case where a spatially coherent parameter is approximated as an n-valued discrete variable. We give the derivations in slightly less detail than for the binary case, as they are very similar. As in the binary case, we want to maximize the mutual information between two parameters, A and B, extracted from nearby image patches, $\mathcal{A}$ and $\mathcal{B}$. In this case the parameters are discrete n-valued random variables, $A \in \{a_1 \cdots a_n\}$, and $B \in \{b_1 \cdots b_n\}$.

A set of n units is assigned to each patch. The output of the ith unit of patch $\mathcal{A}$ on case α, $p_{i\mathcal{A}}^\alpha$, represents the probability that parameter A takes on value a_i on that case:

$$P(A = a_i \mid \alpha) = p_{i\mathcal{A}}^\alpha = \frac{e^{x_{i\mathcal{A}}^\alpha}}{\sum_{j=1}^n e^{x_{j\mathcal{A}}^\alpha}}$$

where $x_{i\mathcal{A}}^\alpha$ is the total weighted summed input to the ith unit for patch $\mathcal{A}$. Similarly, the probability that B takes on value b_j on the same case is:

$$P(B = b_j \mid \alpha) = p_{j\mathcal{B}}^\alpha = \frac{e^{x_{j\mathcal{B}}^\alpha}}{\sum_{i=1}^n e^{x_{i\mathcal{B}}^\alpha}}$$

and the joint probability of these events given case α is:

$$P(A = a_i, B = b_j \mid \alpha) = p_{i\mathcal{A}}^\alpha p_{j\mathcal{B}}^\alpha$$

Each point in the individual and joint probability distributions of A and B can be estimated from its expectation in the sample distribution:

$$
\begin{aligned}
P(A = a_i) &= \sum_{\alpha=1}^{N} P^\alpha p_{i\mathcal{A}}^\alpha \\
P(B = b_j) &= \sum_{\alpha=1}^{N} P^\alpha p_{j\mathcal{B}}^\alpha \\
P(A = a_i, B = b_j) &= \sum_{\alpha=1}^{N} P^\alpha p_{i\mathcal{A}}^\alpha p_{j\mathcal{B}}^\alpha
\end{aligned}
$$

As in the binary case, $P^\alpha = \frac{1}{N}, \forall \alpha$.

The mutual information between two n-valued variables, A and B is:

$$
\begin{aligned}
I(A; B) &= H(A) + H(B) - H(A, B) \\
&= -\sum_i P(A = a_i) \log P(A = a_i) - \sum_j P(B = b_j) \log P(B = b_j) \\
&\quad + \sum_{ij} P(A = a_i, B = b_j) \log P(A = a_i, B = b_j)
\end{aligned}
$$

The derivatives of the above probabilities with respect to the total input $x_{k\mathcal{A}}^\alpha$ to the kth output unit for patch $\mathcal{A}$ on a particular case α can be computed as follows:

$$
\begin{aligned}
\frac{\partial P(A = a_i)}{\partial x_{k\mathcal{A}}^\alpha} &= P^\alpha \frac{\partial p_{i\mathcal{A}}^\alpha}{\partial x_{k\mathcal{A}}^\alpha} \\
\frac{\partial P(A = a_i, B = b_j)}{\partial x_{k\mathcal{A}}^\alpha} &= P^\alpha p_{j\mathcal{B}}^\alpha \frac{\partial p_{i\mathcal{A}}^\alpha}{\partial x_{k\mathcal{A}}^\alpha} \\
\frac{\partial p_{i\mathcal{A}}^\alpha}{\partial x_{k\mathcal{A}}^\alpha} &= \begin{cases} p_{k\mathcal{A}}^\alpha (1 - p_{k\mathcal{A}}^\alpha) & \text{if } i = k \\ -p_{i\mathcal{A}}^\alpha p_{k\mathcal{A}}^\alpha & \text{otherwise} \end{cases}
\end{aligned}
$$

Note that the output of the ith unit, $p_{i\mathcal{A}}$, depends on the total inputs of *all* of the competing output units for module $\mathcal{A}$, $x_{k\mathcal{A}}$.

The derivative of I with respect to the output of the ith output unit is:

$$
\begin{aligned}
\frac{\partial I(A; B)}{\partial p_{i\mathcal{A}}^\alpha} &= \frac{\partial H(A)}{\partial p_{i\mathcal{A}}^\alpha} - \frac{\partial H(A, B)}{\partial p_{i\mathcal{A}}^\alpha} \\
&= -\left[\log(P(A = a_i) + 1\right] \frac{\partial P(A = a_i)}{\partial p_{i\mathcal{A}}^\alpha} \\
&\quad + \sum_j \left[\log P(A = a_i, B = b_j) + 1\right] \frac{\partial P(A = a_i, B = b_j)}{\partial p_{i\mathcal{A}}^\alpha} \\
&= -\left[\log(P(A = a_i) + 1\right] P^\alpha + \sum_j \left[\log P(A = a_i, B = b_j) + 1\right] P^\alpha p_{j\mathcal{B}}^\alpha \\
&= -P^\alpha \left[\log P(A = a_i) - \sum_j \log P(A = a_i, B = b_j) p_{j\mathcal{B}}^\alpha\right]
\end{aligned}
$$

The derivative of I with respect to a weight in the network is computed, using the above expression for $\frac{\partial I(A;B)}{\partial p_{i\mathcal{A}}^\alpha}$, in a manner identical to that shown in Subsection A.1.1, substituting $I(A; B)$ for $I(y_i; y_j)$.

Appendix B

The learning equations for continuous Imax

In the real-valued case, we assume that the underlying spatially coherent signal is continuous and approximately Gaussian, as are the parameters extracted from nearby image patches, y_i and y_j. Each unit deterministically computes its output y_i according to some (linear or nonlinear) function $y_i = f(x_i)$ of its total input, $x_i = \sum_k w_{ik} y_k$.

In Chapter 4 we presented experimental results using two alternate objective functions for this case. These objective functions were derived under different assumptions about the signal. In the first model, the ith unit assumed its output to be a noisy version of the signal, $y_i = signal + noise$, and its neighbor's output to be the pure signal, $y_j = signal$; at the same time, the jth unit assumed $y_j = signal + noise$, and $y_i = signal$. (We discussed in Chapter 4 the problems associated with these assumptions, and presented two alternative models.) In the second model, both units assumed $y_i = signal + noise_i$ and $y_j = signal + noise_j$. In the following two sections, we derive the learning equations for each of these.

B.1 Model I.

Our preliminary experiments used the following objective function:[1]

$$I(y_i; y_j) \doteq 0.5 \left(\log \frac{V(y_i)}{V(y_i - y_j)} + \log \frac{V(y_j)}{V(y_i - y_j)} \right)$$

For unit i to maximize the above information measure, it must store four statistics: its output mean $\langle y_i \rangle$ and variance $V(y_i)$, and the mean and variance of the difference of the units' outputs, $\langle y_i - y_j \rangle$ and $V(y_i - y_j)$. These statistics are computed in an initial pass through the training set:

$$\langle y_i \rangle = \frac{1}{N} \sum_{\alpha=1}^{N} y_i^\alpha$$

$$V(y_i) = \frac{1}{N} \sum_{\alpha=1}^{N} (y_i^\alpha)^2 - \langle y_i \rangle^2$$

[1] We actually added a small constant κ to the denominator terms $V(y_i - y_j)$; this prevents the information measure from growing infinitely large and stabilizes the convergence. Thus a minor change to the above derivation is required when we add in this stabilizing term. The stabilizing term κ is necessary for noise-free problems which the network can solve perfectly, by making $V(y_i - y_j)$ zero while keeping $V(y_i)$ and $V(y_j)$ large. For the random dot stereogram problem the stabilizer is unnecessary, as there is sufficient random variation in the training set to prevent the network from perfectly solving it.

$$\langle y_i - y_j \rangle = \frac{1}{N}\sum_{\alpha=1}^{N}(y_i^\alpha - y_j^\alpha)$$

$$V(y_i - y_j) = \frac{1}{N}\sum_{\alpha=1}^{N}(y_i^\alpha - y_j^\alpha)^2 - \langle y_i - y_j \rangle^2$$

Now we can compute the partial derivative of the information the ith unit conveys about the jth unit with respect to the output of the ith unit on case α:

$$\begin{aligned}
\frac{\partial I(y_i;y_j)}{\partial y_i^\alpha} &= \frac{\partial}{\partial y_i^\alpha}\left[\log\frac{V(y_i)}{V(y_i-y_j)} + \log\frac{V(y_j)}{V(y_i-y_j)}\right] \\
&= \frac{\partial \log V(y_i)}{\partial y_i^\alpha} - 2\frac{\partial \log V(y_i-y_j)}{\partial y_i^\alpha} \\
&= \frac{1}{V(y_i)}\frac{\partial V(y_i)}{\partial y_i^\alpha} - \frac{2}{V(y_i-y_j)}\frac{\partial V(y_i-y_j)}{\partial y_i^\alpha} \\
&= \frac{1}{V(y_i)}\left(\frac{2}{N}y_i^\alpha - \frac{2}{N}\langle y_i\rangle\right) - 2\left[\frac{1}{V(y_i-y_j)}\left(\frac{2}{N}(y_i^\alpha - y_j^\alpha) - \frac{2}{N}\langle y_i - y_j\rangle\right)\right] \\
&= \frac{2}{N}\left[\frac{y_i^\alpha - \langle y_i\rangle}{V(y_i)} - 2\frac{(y_i^\alpha - y_j^\alpha) - \langle y_i - y_j\rangle}{V(y_i-y_j)}\right]
\end{aligned}$$

The derivatives of I with respect to the weights are computed exactly as in Appendix A.1.1, substituting $\frac{\partial I}{\partial y_i^\alpha}$ for $\frac{\partial I}{\partial p_i^\alpha}$.

B.2 Model II

In the second model, we maximize the following objective function:

$$\begin{aligned}
I^* = I\left(\frac{y_i+y_j}{2}; sig\right) &= 0.5\log\frac{V(sig + \frac{noise_a+noise_b}{2})}{V(\frac{noise_a+noise_b}{2})} \\
&= 0.5\log\frac{V(sig + \frac{noise_a+noise_b}{2})}{V(\frac{noise_a-noise_b}{2})} \\
&= 0.5\log\frac{V(\frac{y_i+y_j}{2})}{V(\frac{y_i-y_j}{2})} \\
&= 0.5\log\frac{V(y_i+y_j)}{V(y_i-y_j)}
\end{aligned}$$

Each output unit must store four statistics, for each neighboring output unit j: the mean and variance of the sum and difference of the units' outputs, $\langle y_i + y_j\rangle$, $V(y_i+y_j)$, $\langle y_i - y_j\rangle$ and $V(y_i - y_j)$. These statistics are computed in an initial pass through the training set:

$$\langle y_i + y_j \rangle = \frac{1}{N}\sum_{\alpha=1}^{N}(y_i^\alpha + y_j^\alpha)$$

$$V(y_i + y_j) = \frac{1}{N}\sum_{\alpha=1}^{N}(y_i^\alpha + y_j^\alpha)^2 - \langle y_i + y_j \rangle^2$$

$$\langle y_i - y_j \rangle = \frac{1}{N}\sum_{\alpha=1}^{N}(y_i^\alpha - y_j^\alpha)$$

$$V(y_i - y_j) = \frac{1}{N}\sum_{\alpha=1}^{N}(y_i^\alpha - y_j^\alpha)^2 - \langle y_i - y_j \rangle^2$$

Now we can compute the partial derivative of I^* with respect to the output of the ith output unit on case α:

$$\begin{aligned}
\frac{\partial {I^*}_{y_i;y_j}}{\partial y_i^\alpha} &= 0.5 \frac{\partial}{\partial y_i^\alpha} \log \frac{V(y_i+y_j)}{V(y_i-y_j)} \\
&= 0.5 \left[\frac{\partial \log V(y_i+y_j)}{\partial y_i^\alpha} - \frac{\partial \log V(y_i-y_j)}{\partial y_i^\alpha} \right] \\
&= 0.5 \left[\frac{1}{V(y_i+y_j)} \frac{\partial V(y_i+y_j)}{\partial y_i^\alpha} \;-\; \frac{1}{V(y_i-y_j)} \frac{\partial V(y_i-y_j)}{\partial y_i^\alpha} \right] \\
&= 0.5 \left[\frac{1}{V(y_i+y_j)} \left(\frac{2}{N}(y_i^\alpha + y_j^\alpha) \;-\; \frac{2}{N}\langle y_i + y_j \rangle \right) \right. \\
&\qquad \left. - \frac{1}{V(y_i-y_j)} \left(\frac{2}{N}(y_i^\alpha - y_j^\alpha) \;-\; \frac{2}{N}\langle y_i - y_j \rangle \right) \right] \\
&= \frac{1}{N} \left[\frac{y_i^\alpha + y_j^\alpha - \langle y_i + y_j \rangle}{V(y_i)} - \frac{(y_i^\alpha - y_j^\alpha) - \langle y_i - y_j \rangle}{V(y_i - y_j)} \right]
\end{aligned}$$

The derivatives of I^* with respect to the weights are computed exactly as in Appendix A.1.1, substituting I^* for I, and $\frac{\partial I^*}{\partial y_i^\alpha}$ for $\frac{\partial I}{\partial p_i^\alpha}$.

Appendix C
The learning equations for Imax with mixture models

In Chapter 5, we extended the continuous Imax learning procedure in several ways, using mixture models. Three different models were considered. First, we assumed a model in which a spatially coherent signal is roughly equal in nearby patches, and can be approximated by a mixture of Gaussians; under this model, we showed how a set of competing units, each of which is assigned to model one of the mixture components, can learn to form a population code for depth. Second, we formed a mixture model of coherent and incoherent cases, and trained a network to maximize I only on coherent cases, thereby throwing out cases with discontinuities. Third, we formed a mixture model of the signal across curved surfaces, and trained a network to learn multiple interpolating models for depth, as well as discontinuity detectors to gate the interpolators. We also presented an alternative objective function for the third model, based on the mixture of competing experts algorithm (Jacobs, Jordan, Nowlan and Hinton, 1991). In the following four sections, we show derivations of the Imax learning equations for the three models, as well as the learning equations for the competing experts implementation of the third model.

C.1 Imax based on a mixture model of the spatially coherent signal: Population codes

Under a finite mixture model of a signal, s, the signal is assumed to have come from one of a set of n alternative distributions, $f_i(s;\theta_i)$, each characterized by a parameter vector θ_i. Each probability distribution function, f_i, represents the probability of s given the ith model.

The total probability of a particular observation of s can be computed by decomposing this observation into the probabilities of s given each model:

$$p(s) = \sum_{i=1}^{n} \pi_i f_i(s, \theta_i)$$

where the π_is are the "mixing proportions", or marginal probabilities of the models.

For our spatial coherence model, we assume that there is some spatially coherent signal s, whose probability distribution can be modelled by a mixture of n equi-probable Gaussians having equal variances:

$$p(s) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-(s-\mu_i)^2/2\sigma^2}$$

Under this mixture model, the probability that s was generated by the ith Gaussian is:

$$p_i(s) = \frac{e^{-(s-\mu_i)^2/2\sigma^2}}{\sum_j e^{-(s-\mu_j)^2/2\sigma^2}}$$

For each input patch a, we have n competing output units. We interpret the linear output of each of these units, y_{ia}, as the signal shifted toward the mean of the ith Gaussian, plus noise:

$$y_{ia} = s - \mu_i + noise_{ia}$$

The variance of the Gaussians, σ^2, is estimated by each output unit to be some proportion T of its own output variance ($0 \leq T \leq 1$); this variance is approximately equal to a constant T' (which absorbs T) times the sum of its squared weights (assuming equi-variance independent signals on its input lines): $\sigma^2 \doteq T'V(y_i) \doteq T \sum_j w_{ij}^2$.

Under the above assumptions, we can compute the probability that each of the n Gaussians generated the signal on a particular case, purely in terms of the outputs of the group of units:

$$p_{ia}(s) \doteq \frac{e^{-y_{ia}{}^2/(2T\ \hat{V}(y_{ia}))}}{\sum_j e^{-y_{ja}{}^2/(2T\ \hat{V}(y_{ja}))}}$$

The information that the ith neighboring modules' outputs convey about the common underlying signal, s, given that model i holds, is;

$$I(y_{ia} + y_{ib}; s \mid model\ \ i) = 0.5 \log \frac{V^{iab}(y_{ia} + y_{ib})}{V^{iab}(y_{ia} - y_{ib})}$$

where the V^{iab}s are the variances given that the underlying signal in patches a *and* b was generated by the ith model.

These variances are computed by weighting each term by the probability that model i holds in *both* image patches, $p_{iab} = p_{ia}p_{ib}$, for that case:

$$V^{iab}(y_{ia} + y_{ib}) = \left\langle (y_{ia} + y_{ib})^2 p_{iab} \right\rangle - \langle (y_{ia} + y_{ib})\, p_{iab} \rangle^2$$
$$V^{iab}(y_{ia} - y_{ib}) = \left\langle (y_{ia} - y_{ib})^2 p_{iab} \right\rangle - \langle (y_{ia} - y_{ib})\, p_{iab} \rangle^2$$

The total objective function to maximize is:

$$I^{**} = \sum_{a<b} \sum_i 0.5 \langle p_{iab} \rangle \ \log \frac{V^{iab}(y_{ia} + y_{ib})}{V^{iab}(y_{ia} - y_{ib})}$$

where $\langle p_{iab} \rangle = \langle p_{ia} p_{ib} \rangle$ is the expectation that the ith model holds in patches a and b, averaged over all cases.

The derivative of the total probability of the ith model holding for patches a and b, $\langle p_{iab} \rangle$, with respect to the total input, x_{ka}^α, to the kth output unit for patch a on case α is:

$$\frac{\partial \langle p_{iab} \rangle}{\partial x_{ka}^\alpha} = P^\alpha p_{ib}^\alpha \frac{\partial p_{ia}^\alpha}{\partial x_{ka}^\alpha}$$

The probability of the signal in patch a, under each model i, on a particular case α, can be differentiated with respect to x_{ka}^α as follows:

$$\begin{aligned}
\frac{\partial p_{ia}^\alpha}{\partial x_{ka}^\alpha} &= \frac{\partial}{\partial x_{ka}^\alpha} \left(\frac{-y_{ka}^2}{2T\hat{V}(y_{ka})} \right) \begin{cases} p_{ka}^\alpha(1 - p_{ka}^\alpha) & \text{if } i = k \\ -p_{ia}^\alpha p_{ka}^\alpha & \text{otherwise} \end{cases} \\
&= -\frac{y_{ka}^\alpha}{T\hat{V}(y_{ka})} \frac{dy_{ka}^\alpha}{dx_{ka}^\alpha} \begin{cases} p_{ka}^\alpha(1 - p_{ka}^\alpha) & \text{if } i = k \\ -p_{ia}^\alpha p_{ka}^\alpha & \text{otherwise} \end{cases}
\end{aligned}$$

To differentiate the same probability with respect to a weight to the kth output unit, w_{kj}, we also need to consider how that weight affects p^α_{ia} via the estimated variance, $\hat{V}(y_{ka})$:

$$\begin{aligned}
\frac{\partial p^\alpha_{ia}}{\partial w_{kj}} &= \frac{\partial p^\alpha_{ia}}{\partial x^\alpha_{ka}}\frac{\partial x^\alpha_{ka}}{\partial w_{kj}} + \frac{\partial p^\alpha_{ia}}{\partial \hat{V}(y_{ka})}\frac{\partial \hat{V}(y_{ka})}{\partial w_{kj}} \\
&= \frac{\partial p^\alpha_{ia}}{\partial x^\alpha_{ka}}\frac{\partial x^\alpha_{ka}}{\partial w_{kj}} + \frac{{y^\alpha_{ka}}^2}{T\hat{V}(y_{ka})^2}\frac{\partial \hat{V}(y_{ka})}{\partial w_{kj}} \begin{cases} p^\alpha_{ka}(1-p^\alpha_{ka}) & \text{if } i=k \\ -p^\alpha_{ia}p^\alpha_{ka} & \text{otherwise} \end{cases} \\
&= \frac{\partial p^\alpha_{ia}}{\partial x^\alpha_{ka}}\frac{\partial x^\alpha_{ka}}{\partial w_{kj}} + \frac{{y^\alpha_{ka}}^2}{T\hat{V}(y_{ka})^2} 2w_{kj} \begin{cases} p^\alpha_{ka}(1-p^\alpha_{ka}) & \text{if } i=k \\ -p^\alpha_{ia}p^\alpha_{ka} & \text{otherwise} \end{cases}
\end{aligned}$$

The partial derivative of the objective function I^{**} with respect to the output of the kth output unit on case α is:

$$\begin{aligned}
\frac{\partial I^{**}}{\partial y^\alpha_{ka}} &= 0.5\sum_b\sum_i \frac{\partial}{\partial y^\alpha_{ka}}\left[\langle p_{iab}\rangle \log\frac{V^{iab}(y_{ia}+y_{ib})}{V^{iab}(y_{ia}-y_{ib})}\right] \\
&= 0.5\sum_b\sum_i \left[\langle p_{iab}\rangle\left(\frac{\partial \log V^{iab}(y_{ia}+y_{ib})}{\partial y^\alpha_{ka}} - \frac{\partial \log V^{iab}(y_{ia}-y_{ib})}{\partial y^\alpha_{ka}}\right)\right. \\
&\qquad \left. + \frac{\partial\langle p_{iab}\rangle}{\partial y^\alpha_{ka}}\log\frac{V^{iab}(y_{ia}+y_{ib})}{V^{iab}(y_{ia}-y_{ib})}\right] \\
&= 0.5\sum_b\sum_i \left[\frac{\langle p_{iab}\rangle}{V^{iab}(y_{ia}+y_{ib})}\frac{\partial V^{iab}(y_{ia}+y_{ib})}{\partial y^\alpha_{ka}} - \frac{\langle p_{iab}\rangle}{V^{iab}(y_{ia}-y_{ib})}\frac{\partial V^{iab}(y_{ia}-y_{ib})}{\partial y^\alpha_{ka}}\right. \\
&\qquad \left. +\log\frac{V^{iab}(y_{ia}+y_{ib})}{V^{iab}(y_{ia}-y_{ib})}P^\alpha p^\alpha_{ib}\frac{\partial p^\alpha_{ia}}{\partial y^\alpha_{ka}}\right] \\
&= 0.5\sum_b\left[\frac{\langle p_{kab}\rangle\, p^\alpha_{kab}}{V^{kab}(y_{ka}+y_{kb})}\left(\frac{2}{N}(y^\alpha_{ka}+y^\alpha_{kb}) - \frac{2}{N}\langle y_{ka}+y_{kb}\rangle\right)\right. \\
&\qquad \left. - \frac{\langle p_{kab}\rangle\, p^\alpha_{kab}}{V^{kab}(y_{ka}-y_{kb})}\left(\frac{2}{N}(y^\alpha_{ka}-y^\alpha_{kb}) - \frac{2}{N}\langle y_{ka}-y_{kb}\rangle\right)\right] \\
&\quad +;0.5\sum_b\sum_i\left[\frac{\langle p_{iab}\rangle}{V^{iab}(y_{ia}+y_{ib})}\frac{1}{N}\left(y^\alpha_{ia}+y^\alpha_{ib} - \langle y_{ia}+y_{ib}\rangle\right)^2\frac{\partial p^\alpha_{iab}}{\partial y^\alpha_{ka}}\right. \\
&\qquad - \frac{\langle p_{iab}\rangle}{V^{iab}(y_{ia}-y_{ib})}\frac{1}{N}\left(y^\alpha_{ia}-y^\alpha_{ib} - \langle y_{ia}-y_{ib}\rangle\right)^2\frac{\partial p^\alpha_{iab}}{\partial y^\alpha_{ka}} \\
&\qquad \left. +\log\frac{V^{iab}(y_{ia}+y_{ib})}{V^{iab}(y_{ia}-y_{ib})}\frac{1}{N}p^\alpha_{ib}\frac{\partial p^\alpha_{ia}}{\partial y^\alpha_{ka}}\right] \\
&= \sum_b\frac{1}{N}\left[\langle p_{kab}\rangle\, p^\alpha_{kab}\left(\frac{y^\alpha_{ka}+y^\alpha_{kb}-\langle y_{ka}+y_{kb}\rangle}{V^{kab}(y_{ka}+y_{kb})} - \frac{(y^\alpha_{ka}-y^\alpha_{kb})-\langle y_{ka}-y_{kb}\rangle}{V^{kab}(y_{ka}-y_{kb})}\right)\right. \\
&\qquad + 0.5\sum_i\left(\langle p_{iab}\rangle\frac{(y^\alpha_{ia}+y^\alpha_{ib}-\langle y_{ia}+y_{ib}\rangle)^2}{V^{iab}(y_{ia}+y_{ib})} - \langle p_{iab}\rangle\frac{(y^\alpha_{ia}-y^\alpha_{ib}-\langle y_{ia}-y_{ib}\rangle)^2}{V^{iab}(y_{ia}-y_{ib})}\right. \\
&\qquad \left.\left. +\log\frac{V^{iab}(y_{ia}+y_{ib})}{V^{iab}(y_{ia}-y_{ib})}\right)p^\alpha_{ib}\frac{\partial p^\alpha_{ia}}{\partial y^\alpha_{ka}}\right]
\end{aligned}$$

To compute the derivative of I^{**} with respect an incoming weight w_{kj} to the kth output unit for patch a, we must include the effect of the variance term $V(y_{ka})$:

$$\frac{\partial I^{**}}{\partial w_{kj}} = \sum_\alpha\left[\frac{\partial I^{**}}{\partial y^\alpha_{ka}}\frac{dy^\alpha_{ka}}{dx^\alpha_{ka}}\frac{\partial x^\alpha_{ka}}{\partial w_{kj}} + \sum_i\frac{\partial I^{**}}{\partial p^\alpha_{ia}}\frac{\partial p^\alpha_{ia}}{\partial V(y_{ka})}\frac{\partial V(y_{ka})}{\partial w_{kj}}\right]$$

$$= \sum_\alpha \left[\frac{\partial I^{**}}{\partial y^\alpha_{ka}} \frac{dy^\alpha_{ka}}{dx^\alpha_{ka}} \frac{\partial x^\alpha_{ka}}{\partial w_{kj}} + 0.5 \sum_i \sum_b \frac{1}{N} \left(\langle p_{iab} \rangle \frac{(y^\alpha_{ia} + y^\alpha_{ib} - \langle y_{ia} + y_{ib} \rangle)^2}{V^{iab}(y_{ia} + y_{ib})} \right. \right.$$

$$\left. \left. - \langle p_{iab} \rangle \frac{(y^\alpha_{ia} - y^\alpha_{ib} - \langle y_{ia} - y_{ib} \rangle)^2}{V^{iab}(y_{ia} - y_{ib})} + \log \frac{V^{iab}(y_{ia} + y_{ib})}{V^{iab}(y_{ia} - y_{ib})} \right) p^\alpha_{ib} \frac{\partial p^\alpha_{ia}}{\partial V(y_{ka})} \frac{\partial V(y_{ka})}{\partial w_{kj}} \right]$$

For other weights in the network, the derivative is just:

$$\frac{\partial I^{**}}{\partial w_{lm}} = \sum_\alpha \frac{\partial I^{**}}{\partial x^\alpha_{la}} \frac{\partial x^\alpha_{la}}{\partial w_{lm}}$$

This can be computed by back-propagation, as shown in Appendix A.1.1.

C.2 Imax using a mixture of coherence, discontinuity models

We can deal with discontinuities by forming a mixture model of coherence and incoherence, and only maximizing information about the signal on coherent cases. We applied this model to the depth interpolating units described in Chapter 4. The goal of the learning is to predict a locally extracted parameter y_c, by computing a weighted sum of parameters extracted from four nearby image regions, $\hat{y}_c = w_a y_a + w_b y_b + w_d y_d + w_e y_e$. We estimate the probability of the continuity model holding, on a given case α, as follows:

$$p_{cont}(\hat{y}^\alpha_c) = \frac{N(\hat{y}^\alpha_c, y^\alpha_c, V_{cont}(\hat{y}^\alpha_c))}{N(\hat{y}^\alpha_c, y^\alpha_c, V_{cont}(\hat{y}^\alpha_c)) + k_{discont}}$$

where $k_{discont}$ is a constant representing a uniform density, $V_{cont}(\hat{y}_c)$ is a parameter which is gradually shrunk over the course of learning, and N is the normal distribution:

$$N(\hat{y}^\alpha_c, y^\alpha_c, V_{cont}(\hat{y}_c)) = \frac{1}{\sqrt{2\pi V_{cont}(\hat{y}_c)}} \exp(-(\hat{y}^\alpha_c - y^\alpha_c)^2 / 2V_{cont}(\hat{y}_c))$$

The objective function to maximize is:

$$I_{cont} = 0.5 \;\; P_{cont} \;\; \log \frac{V_{cont}(y_c + \hat{y}_c)}{V_{cont}(y_c - \hat{y}_c)}$$

where $P_{cont} = \langle p_{cont}(\hat{y}_c) \rangle$ is the total probability of the continuity model across cases, and the V_{cont}s are the variances given the continuity model. The variances are are computed as follows:

$$V_{cont}(\hat{y}_c + y_c) = \left\langle (\hat{y}_c + y_c)^2 p_{cont}(\hat{y}_c) \right\rangle - \langle (\hat{y}_c + y_c) \, p_{cont}(\hat{y}_c) \rangle^2$$
$$V_{cont}(\hat{y}_c - y_c) = \left\langle (\hat{y}_c - y_c)^2 p_{cont}(\hat{y}_c) \right\rangle - \langle (\hat{y}_c - y_c) \, p_{cont}(\hat{y}_c) \rangle^2$$

The probability of the continuity model for a particular case can be differentiated with respect to the outputs of the interpolating unit, $\hat{y}^\alpha_c$, and the local depth-extracting unit, y^α_c, as follows:

$$\frac{\partial p_{cont}(\hat{y}^\alpha_c)}{\partial \hat{y}^\alpha_c} = -p_{cont}(\hat{y}^\alpha_c)(1 - p_{cont}(\hat{y}^\alpha_c)) \frac{1}{V_{cont}(\hat{y}_c)} (\hat{y}^\alpha_c - y^\alpha_c)$$
$$\frac{\partial p_{cont}(\hat{y}^\alpha_c)}{\partial y^\alpha_c} = -\frac{\partial p_{cont}(\hat{y}^\alpha_c)}{\partial \hat{y}^\alpha_c}$$

The objective function I_{cont} can now be differentiated with respect to the output of the interpolating unit on case

α, $\hat{y}_c^\alpha$:

$$\begin{aligned}
\frac{\partial I_{cont}}{\partial \hat{y}_c^\alpha} &= 0.5 \frac{\partial}{\partial \hat{y}_c^\alpha}\left[\langle p_{cont}\rangle \log \frac{V_{cont}(\hat{y}_c + y_c)}{V_{cont}(\hat{y}_c - y_c)}\right] \\
&= 0.5 \left[\langle p_{cont}\rangle \left(\frac{\partial \log V_{cont}(\hat{y}_c + y_c)}{\partial \hat{y}_c^\alpha} - \frac{\partial \log V_{cont}(\hat{y}_c - y_c)}{\partial \hat{y}_c^\alpha}\right)\right. \\
&\qquad \left. + \frac{\partial \langle p_{cont}\rangle}{\partial \hat{y}_c^\alpha} \log \frac{V_{cont}(\hat{y}_c + y_c)}{V_{cont}(\hat{y}_c - y_c)}\right] \\
&= 0.5 \left[\frac{\langle p_{cont}\rangle}{V_{cont}(\hat{y}_c + y_c)} \frac{\partial V_{cont}(\hat{y}_c + y_c)}{\partial \hat{y}_c^\alpha} - \frac{\langle p_{cont}\rangle}{V_{cont}(\hat{y}_c - y_c)} \frac{\partial V_{cont}(\hat{y}_c - y_c)}{\partial \hat{y}_c^\alpha}\right. \\
&\qquad \left. + \log \frac{V_{cont}(\hat{y}_c + y_c)}{V_{cont}(\hat{y}_c - y_c)} P^\alpha \frac{\partial p_{cont}^\alpha}{\partial \hat{y}_c^\alpha}\right] \\
&= 0.5 \left[\frac{\langle p_{cont}\rangle p_{cont}^\alpha}{V_{cont}(\hat{y}_c + y_c)} \left(\frac{2}{N}(\hat{y}_c^\alpha + y_c^\alpha) - \frac{2}{N}\langle \hat{y}_c + y_c\rangle\right)\right. \\
&\qquad - \frac{\langle p_{cont}\rangle p_{cont}^\alpha}{V_{cont}(\hat{y}_c - y_c)} \left(\frac{2}{N}(\hat{y}_c^\alpha - y_c^\alpha) - \frac{2}{N}\langle \hat{y}_c - y_c\rangle\right) \\
&\qquad + \frac{\langle p_{cont}\rangle}{V_{cont}(\hat{y}_c + y_c)} \frac{1}{N} (\hat{y}_c^\alpha + y_c^\alpha - \langle \hat{y}_c + y_c\rangle)^2 \frac{\partial p_{cont}^\alpha}{\partial \hat{y}_c^\alpha} \\
&\qquad - \frac{\langle p_{cont}\rangle}{V_{cont}(\hat{y}_c - y_c)} \frac{1}{N} (\hat{y}_c^\alpha - y_c^\alpha - \langle \hat{y}_c - y_c\rangle)^2 \frac{\partial p_{cont}^\alpha}{\partial \hat{y}_c^\alpha} \\
&\qquad \left. + \log \frac{V_{cont}(\hat{y}_c + y_c)}{V_{cont}(\hat{y}_c - y_c)} \frac{1}{N} \frac{\partial p_{cont}^\alpha}{\partial \hat{y}_c^\alpha}\right] \\
&= \frac{1}{N} \langle p_{cont}\rangle p_{cont}^\alpha \left(\frac{\hat{y}_c^\alpha + y_c^\alpha - \langle \hat{y}_c + y_c\rangle}{V_{cont}(\hat{y}_c + y_c)} - \frac{(\hat{y}_c^\alpha - y_c^\alpha) - \langle \hat{y}_c - y_c\rangle}{V_{cont}(\hat{y}_c - y_c)}\right) \\
&\quad + \frac{1}{2N}\left[\langle p_{cont}\rangle \frac{(\hat{y}_c^\alpha + y_c^\alpha - \langle \hat{y}_c + y_c\rangle)^2}{V_{cont}(\hat{y}_c + y_c)} - \langle p_{cont}\rangle \frac{(\hat{y}_c^\alpha - y_c^\alpha - \langle \hat{y}_c - y_c\rangle)^2}{V_{cont}(\hat{y}_c - y_c)}\right. \\
&\qquad \left. + \log \frac{V_{cont}(\hat{y}_c + y_c)}{V_{cont}(\hat{y}_c - y_c)}\right] \frac{\partial p_{cont}^\alpha}{\partial \hat{y}_c^\alpha}
\end{aligned}$$

The derivative of I with respect to y_c^α can be computed in a similar manner. Given $\frac{\partial I_{cont}}{\partial \hat{y}_c^\alpha}$ and $\frac{\partial I_{cont}}{\partial y_c^\alpha}$, the derivatives for all the weights in the network can easily be computed by applying the chain rule, using the back-propagation procedure described in Appendix A, Section A.1.1.

C.3 Imax using a mixture of coherence models

The final mixture model we considered in Chapter 5 caused a network to learn a mixture of interpolating models, and a set of location-specific discontinuity detectors. We assume, as in the previous section, that there is a spatially coherent parameter which can be predicted at any given image region by linearly combining several values at spatially adjacent regions.

Like the model in Section C.1, we have a set of competing output units, each trying to form a different model of the current case. But this time, each is trying to learn a good interpolator for a locally extracted parameter y_c, $\hat{y}_{ic} = w_{ia}y_a + w_{ib}y_b + w_{id}y_d + w_{ie}y_e$, for the set of cases on which the ith model holds. Instead of making the probability that each model holds a function of the interpolators' outputs, we allocate a set controlling units, in one-to-one correspondence with the interpolators. The output of the ith controller, p_i, represents the probability that the ith interpolator's model holds.

Each controller's output is a normalized exponential function of its *squared* total input:

$$p_i = \frac{e^{{x_i}^2/\hat{\sigma}(x_i)^2}}{\sum_j e^{{x_j}^2/\hat{\sigma}(x_j)^2}}$$

As in the population code model, we divide the squared total input in the exponential by an estimate of its variance, $\hat{\sigma}(x_j)^2 = k\sum_{ji} w_j i^2$.

The objective function to maximize is identical to the one for the population code model presented in Section C.1, except that the p_is are computed differently:

$$I^{**} = \sum_i \langle p_i \rangle \log \frac{V^i(\hat{y}_{ic} + y_c)}{V^i(\hat{y}_{ic} - y_c)}$$

where the V^is represent variances given that the ith model holds.

The derivation of the learning equations for this model is very similar to those of the previous two sections. It is made simpler by the fact that separate units compute the p_is. So we can consider separately the effect of the variables $\hat{y}_c$ and y_c on I (which depend on weights to the interpolating units and their hidden units) and the effect of the p_is on I (which depend on the weights to the controller units).

The derivative of I^{**} with respect to the output of the ith interpolating unit, $\hat{y}_{ic}^{\alpha}$, on case α is:

$$\begin{aligned}
\frac{\partial I^{**}}{\partial \hat{y}_{ic}^{\alpha}} &= 0.5 \frac{\partial}{\partial \hat{y}_{ic}^{\alpha}} \left[\langle p_i \rangle \log \frac{V^i(\hat{y}_{ic} + y_c)}{V^i(\hat{y}_{ic} - y_c)} \right] \\
&= 0.5 \langle p_i \rangle \left[\frac{\partial \log V^i(\hat{y}_{ic} + y_c)}{\partial \hat{y}_{ic}^{\alpha}} - \frac{\partial \log V^i(\hat{y}_{ic} - y_c)}{\partial \hat{y}_{ic}^{\alpha}} \right] \\
&= 0.5 \langle p_i \rangle \left[\frac{1}{V^i(\hat{y}_{ic} + y_c)} \frac{\partial V^i(\hat{y}_{ic} + y_c)}{\partial \hat{y}_{ic}^{\alpha}} - \frac{1}{V^i(\hat{y}_{ic} - y_c)} \frac{\partial V^i(\hat{y}_{ic} - y_c)}{\partial \hat{y}_{ic}^{\alpha}} \right] \\
&= 0.5 \langle p_i \rangle \left[\frac{p_i^{\alpha}}{V^i(\hat{y}_{ic} + y_c)} \left(\frac{2}{N}(\hat{y}_{ic}^{\alpha} + y_c^{\alpha}) - \frac{2}{N}\langle \hat{y}_{ic} + y_c \rangle \right) \right. \\
&\qquad \left. - \frac{p_i^{\alpha}}{V^i(\hat{y}_{ic} - y_c)} \left(\frac{2}{N}(\hat{y}_{ic}^{\alpha} - y_c^{\alpha}) - \frac{2}{N}\langle \hat{y}_{ic} - y_c \rangle \right) \right] \\
&= \frac{1}{N} \langle p_i \rangle p_i^{\alpha} \left[\frac{\hat{y}_{ic}^{\alpha} + y_c^{\alpha} - \langle \hat{y}_{ic} + y_c \rangle}{V^i(\hat{y}_{ic} + y_c)} - \frac{(\hat{y}_{ic}^{\alpha} - y_c^{\alpha}) - \langle \hat{y}_{ic} - y_c \rangle}{V^i(\hat{y}_{ic} - y_c)} \right]
\end{aligned}$$

From this term, we can easily compute the derivatives of I with respect to weights affecting the interpolating units (including those of the hidden units), using back-propagation.

The output of one controller, p_i, depends on the inputs to all of the other controllers, x_i, as well as its own, because of the normalization. The derivative of p_i^{α} with respect to the total input to the jth controller is:

$$\begin{aligned}
\frac{\partial p_i^{\alpha}}{\partial x_j^{\alpha}} &= \frac{\partial}{\partial x_j^{\alpha}} \left(\frac{x_j^2}{\sigma^2(x_j)} \right) \begin{cases} p_j^{\alpha}(1 - p_j^{\alpha}) & \text{if } i = j \\ -p_i^{\alpha} p_j^{\alpha} & \text{otherwise} \end{cases} \\
&= \frac{2x_j^{\alpha}}{\sigma^2(x_j)} \begin{cases} p_j^{\alpha}(1 - p_j^{\alpha}) & \text{if } i = j \\ -p_i^{\alpha} p_j^{\alpha} & \text{otherwise} \end{cases}
\end{aligned}$$

To compute the derivatives of the p_is with respect to the weights, we also need the derivative of the p_is with respect to the approximated variance of each controller, $\sigma^2(x_j)$:

$$\frac{\partial p_i^{\alpha}}{\partial \sigma^2(x_j)} = \frac{\partial}{\partial \sigma^2(x_j)} \left(\frac{x_j^2}{\sigma^2(x_j)} \right) \begin{cases} p_j^{\alpha}(1 - p_j^{\alpha}) & \text{if } i = j \\ -p_i^{\alpha} p_j^{\alpha} & \text{otherwise} \end{cases}$$

$$= \quad -\frac{(x_j^\alpha)^2}{\sigma^4(x_j)} \begin{cases} p_j^\alpha(1-p_j^\alpha) & \text{if } i=j \\ -p_i^\alpha p_j^\alpha & \text{otherwise} \end{cases}$$

The derivative of I^{**} with respect to the output of the ith controller, p_i^α, on case α is:

$$\frac{\partial I^{**}}{\partial p_i^\alpha} = 0.5 \frac{\partial}{\partial p_i^\alpha}\left[\langle p_i\rangle \log \frac{V^i(\hat{y}_{ic}+y_c)}{V^i(\hat{y}_{ic}-y_c)}\right]$$

$$= 0.5\left[\langle p_i\rangle \left(\frac{\partial \log V^i(\hat{y}_{ic}+y_c)}{\partial p_i^\alpha} - \frac{\partial \log V^i(\hat{y}_{ic}-y_c)}{\partial p_i^\alpha}\right) + \frac{\partial \langle p_i\rangle}{\partial p_i^\alpha}\log \frac{V^i(\hat{y}_{ic}+y_c)}{V^i(\hat{y}_{ic}-y_c)}\right]$$

$$= 0.5\left[\frac{\langle p_i\rangle}{V^i(\hat{y}_{ic}+y_c)}\frac{\partial V^i(\hat{y}_{ic}+y_c)}{\partial p_i^\alpha} - \frac{\langle p_i\rangle}{V^i(\hat{y}_{ic}-y_c)}\frac{\partial V^i(\hat{y}_{ic}-y_c)}{\partial p_i^\alpha} + \log\frac{V^i(\hat{y}_{ic}+y_c)}{V^i(\hat{y}_{ic}-y_c)}\frac{1}{N}\right]$$

$$= 0.5\left[\frac{\langle p_i\rangle}{V^i(\hat{y}_{ic}+y_c)}\frac{1}{N}\left(\hat{y}_{ic}^\alpha + y_c^\alpha \;-\; \langle \hat{y}_{ic}+y_c\rangle\right)^2 \;-\; \frac{\langle p_i\rangle}{V^i(\hat{y}_{ic}-y_c)}\frac{1}{N}\left(\hat{y}_{ic}^\alpha - y_c^\alpha \;-\; \langle \hat{y}_{ic}-y_c\rangle\right)^2 \right.$$
$$\left. + \log\frac{V^i(\hat{y}_{ic}+y_c)}{V^i(\hat{y}_{ic}-y_c)}\frac{1}{N}\right]$$

$$= 0.5\frac{1}{N}\left[\langle p_i\rangle \frac{(\hat{y}_{ic}^\alpha + y_c^\alpha - \langle \hat{y}_{ic}+y_c\rangle)^2}{V^i(\hat{y}_{ic}+y_c)} - \langle p_i\rangle \frac{(\hat{y}_{ic}^\alpha - y_c^\alpha - \langle \hat{y}_{ic}-y_c\rangle)^2}{V^i(\hat{y}_{ic}-y_c)} + \log\frac{V^i(\hat{y}_{ic}+y_c)}{V^i(\hat{y}_{ic}-y_c)}\right]$$

C.4 Mixture of competing experts

We can reformulate the problem solved by the learning procedure in the previous section, using an unsupervised version of the mixture of competing experts algorithm (Jacobs, Jordan, Nowlan and Hinton, 1991), assuming that the inputs to the interpolating and gating units are fixed (that is, we do not back-propagate derivatives through the hidden layers to adjust the weights to the units in layers 1 and 2). The output of the i^{th} expert, $\hat{y}_{i,c}$, is treated as the mean of a Gaussian distribution with variance σ^2 and the normalized output of each controller, p_i, is treated as the mixing proportion of that Gaussian. So, for each training case, the outputs of the experts and their controllers define a probability distribution that is a mixture of Gaussians. The aim of the learning is to maximize the log probability density of the desired output, y_c, under this mixture of Gaussians distribution. For a particular training case, α, this log probability is given by:

$$\log P(y_c^\alpha) = \log \sum_i p_i^\alpha \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y_c^\alpha - \hat{y}_{i,c}^\alpha)^2}{2\sigma^2}\right)$$

where the variance parameter σ^2 is fixed, and each mixing proportion, p_i, is computed by one of the controller units as follows:

$$p_i = \frac{e^{{x_i}^2/2\sigma^2}}{\sum_j e^{{x_j}^2/2\sigma^2}}$$

and each mean, $\hat{y}_{i,c}$, is computed by one of the interpolating experts:

$$\hat{y}_{i,c} = w_{ia}y_a + w_{ib}y_b + w_{id}y_d + w_{ie}y_e$$

The derivative of the objective function on each training case, $\log P(y_c^\alpha)$, with respect to the output of one of the interpolating units, y_i^α, is:

$$\frac{\partial}{\partial \hat{y}_{i,c}^\alpha}\log P(y_c^\alpha) \quad = \quad \frac{1}{P(y_c^\alpha)}\left(\sum_j P_j^\alpha \frac{\partial}{\partial \hat{y}_{j,c}^\alpha} N(y_c^\alpha, \hat{y}_{j,c}^\alpha, \sigma^2)\right)$$

$$\begin{aligned} &= \frac{1}{P(y_c^\alpha)} P_i^\alpha \frac{\partial}{\partial \hat{y}_{i,c}^\alpha} N(y_c^\alpha, \hat{y}_{i,c}^\alpha, \sigma^2) \\ &= \frac{P_i^\alpha}{P(y_c^\alpha)} N(y_c^\alpha, \hat{y}_{i,c}^\alpha, \sigma^2) \frac{y_c^\alpha - \hat{y}_{i,c}^\alpha}{\sigma^2} \end{aligned}$$

The derivative of the output of one controller, p_i^α, with respect to the total input to the jth controller, x_j^α, is:

$$\frac{\partial p_i^\alpha}{\partial x_j^\alpha} = \frac{x_j^\alpha}{\sigma^2} \begin{cases} p_j^\alpha(1 - p_j^\alpha) & \text{if } i = j \\ -p_i^\alpha p_j^\alpha & \text{otherwise} \end{cases}$$

The derivative of $\log P(y_c^\alpha)$ with respect to the output of a controller, p_i^α, is:

$$\frac{\partial \log P(y_c^\alpha)}{\partial p_i^\alpha} = \frac{1}{P(y_c^\alpha)} N(y_c^\alpha, \hat{y}_{i,c}^\alpha, \sigma^2)$$

Appendix D

Equations for learning by contrastive clamping in DBMs

In Chapter 6, we introduced a way to train Boltzmann machines using an unsupervised (or, more accurately, semi-supervised) learning procedure we refer to as contrastive clamping. We used deterministic Boltzmann machines (DBMs) with a restricted architecture. The architecture consists of two (or more) modules. Each module has input units whose states are determined by patterns drawn from some environment, and n output units, whose states encode an n-valued parameter using a "1-of-n" encoding. The state of the ith output unit for a module a is defined as follows:

$$p_{ia} = \frac{e^{x_{ia}/T}}{\sum_j e^{x_{ja}/T}}$$

where x_{ia} is the total input to the ith output unit in module a. The output units of neighboring modules are fully interconnected (but there are no within-module connections between output units).

D.1 Networks that settle to equilibrium

Since the network has recurrent connections (the connections between the two module's sets of output units), the units states are updated so as to settle to fixed points defined by the above equation. To do this, we repeatedly, synchronously apply the following state update rule to all of the output units, until the system converges:

$$p_{ia}(t) = \eta p_{ia}(t-1) + (1-\eta)\frac{e^{x_{ia}(t)/T}}{\sum_j e^{x_{ja}(t)/T}}$$

It can be shown (given the mean field assumptions for the DBM, stated in Chapter 6) that a system which follows the above dynamics converges to a minimum in the Helmholtz free energy, defined as follows for systems with n-state units:

$$\begin{aligned} F &= E - T\,H \\ &= -\sum_{i<j} w_{ij} p_i p_j + T \sum_j p_j \log p_j \end{aligned}$$

We define the probability of a distribution of network states when the inputs are clamped to a positive pattern,

α, relative to the probabilities for all patterns in the training set, β, as follows:

$$\frac{P(I_\alpha)}{\sum_\beta P(I_\beta)} = \frac{e^{-F_\alpha{}^*/T}}{\sum_\beta e^{-F_\beta{}^*/T}}$$

where β indexes over both positive and negative cases, and F_α^* is the free energy of the equilibrium distribution with visible units clamped to pattern α.

The objective function to maximize is the log probability of the network generating the set of positive training cases, $\alpha \in \mathcal{S}^+$, relative to the combined set of positive and negative training cases $\beta \in \mathcal{S} = \mathcal{S}^+ \cup \mathcal{S}^-$:

$$\sum_{\alpha \in \mathcal{S}+} \log \frac{P(I_\alpha)}{\sum_{\beta \in \mathcal{S}} P(I_\beta)} = \sum_{\alpha \in \mathcal{S}+} \log \left(\frac{e^{-F_\alpha/T}}{\sum_{\beta \in \mathcal{S}} e^{-F_\beta/T}} \right)$$

The derivative of the objective function, the log likelihood of the positive training cases, with respect to one of the weights in the network, w_{ji}, for (positive) training case α can be computed at thermal equilibrium as follows:

$$\begin{aligned}
\frac{\partial}{\partial w_{ji}} \log \left(\frac{P(I_\alpha)}{\sum_{\beta \in \mathcal{S}} P(I_\beta)} \right) &= -\frac{1}{T} \left[\frac{\partial F_\alpha{}^*}{\partial w_{ji}} - \frac{1}{\sum_{\gamma \in \mathcal{S}} e^{-F_\gamma{}^*/T}} \sum_{\beta \in \mathcal{S}} e^{-F_\beta{}^*/T} \frac{\partial F_\beta{}^*}{\partial w_{ji}} \right] \\
&= \frac{1}{T} \left[p_i^\alpha p_j^\alpha - \frac{1}{\sum_{\gamma \in \mathcal{S}} e^{-F_\gamma{}^*/T}} \sum_{\beta \in \mathcal{S}} e^{-F_\beta{}^*/T} p_i^\beta p_j^\beta \right] \\
&= \frac{1}{T} \left[p_i^\alpha p_j^\alpha - \sum_{\beta \in \mathcal{S}} \frac{P(I_\alpha)}{\sum_{\gamma \in \mathcal{S}} P(I_\gamma)} p_i^\beta p_j^\beta \right]
\end{aligned}$$

D.1 Non-settling networks

For a network with recurrent connections which we pre-train without settling, we set the states of units as follows, omitting the lateral/recurrent inputs from the x_{ja}s (we can do this by initializing the states of output units to zero, and then applying the following state update rule once):

$$p_{ia} = \frac{e^{x_{ia}/T}}{\sum_j e^{x_{ja}/T}}$$

When a network is permitted to settle to the equilibrium distribution, the following holds:

$$\frac{\partial F_\alpha{}^*}{\partial w_{ji}} = -p_i^\alpha p_j^\alpha$$

If we have not settled to equilibrium, it is no longer true in general that $\frac{\partial F}{\partial p_i} = 0, \forall i$, so the derivatives are more complicated. In this case, the partial derivative of F with respect to a weight from a hidden unit to an output unit in module a is computed as follows:

$$\begin{aligned}
\frac{\partial F_\alpha}{\partial w_{jk}} &= -p_{ka}^\alpha p_{ja}^\alpha + \sum_i \frac{\partial F_\alpha}{\partial p_i^\alpha} \frac{\partial p_i^\alpha}{\partial w_{jk}} \\
&= -p_{ka}^\alpha p_{ja}^\alpha + \sum_i \frac{\partial E_\alpha}{\partial p_i^\alpha} \frac{\partial p_i^\alpha}{\partial w_{jk}} - T \sum_i \frac{\partial H_\alpha}{\partial p_i^\alpha} \frac{\partial p_i^\alpha}{\partial w_{jk}} \\
&= -p_{ka}^\alpha p_{ja}^\alpha - \sum_{i<l} w_{il} \left[p_l^\alpha \frac{\partial p_i^\alpha}{\partial w_{jk}} + p_i^\alpha \frac{\partial p_l^\alpha}{\partial w_{jk}} \right] + T \sum_i (\log p_{ia} + 1) \frac{\partial p_i^\alpha}{\partial w_{jk}} \\
\frac{\partial p_i^\alpha}{\partial w_{jk}} &= \frac{\partial p_i^\alpha}{\partial x_j^\alpha} \frac{\partial x_j^\alpha}{\partial w_{jk}}
\end{aligned}$$

$$= \frac{1}{T}\frac{\partial x_j^\alpha}{\partial w_{jk}}\begin{cases} p_{ja}^\alpha(1-p_{ja}^\alpha) & \text{if } i=j \\ -p_{ia}^\alpha p_{ja}^\alpha & \text{otherwise} \end{cases}$$

$$= \frac{1}{T}y_k^\alpha\begin{cases} p_{ja}^\alpha(1-p_{ja}^\alpha) & \text{if } i=j \\ -p_{ia}^\alpha p_{ja}^\alpha & \text{otherwise} \end{cases}$$

For a weight, w_{ji}, between two output units in different modules,

$$\frac{\partial p_{ia}^\alpha}{\partial w_{ji}} = \frac{\partial p_{jb}^\alpha}{\partial w_{ji}} = 0$$

Therefore,

$$\frac{\partial F_\alpha}{\partial w_{ji}} = -p_{ia}^\alpha p_{jb}^\alpha$$

Appendix E
Equations for learning perceptual invariants

In Chapter 7 (in the section on future work), we described an objective function for a procedure which learns invariant properties of the input. The algorithm we propose is based on a continuous generalization of the GMAX algorithm (Pearlmutter and Hinton, 1986). GMAX is an unsupervised learning procedure for a single, binary stochastic unit. The goal of the learning is to maximize the Kullback divergence between the actual output distribution of the unit, $P(y_i)$, and the distribution $Q(y_i)$ which would be expected if the unit's input lines were independent:

$$\begin{aligned} G(P,Q) &= \int_{y_i} P(y_i) \log \frac{P(y_i)}{Q(y_i)} \\ &= P(y_i = 1)\ \log\ \frac{P(y_i = 1)}{Q(y_i = 1)} + P(y_i = 0)\ \log\ \frac{P(y_i = 0)}{Q(y_i = 0)} \end{aligned}$$

where y_i is the binary output of a unit.

We can generalize the GMAX algorithm to the continuous case, assuming that y_i is computed as a linear function of the inputs, and that the inputs all have Gaussian distributions. In this case, P and Q are Gaussians, so the actual probability distribution of y_i is the normal distribution $P(y_i) = N(y_i, \overline{y_i}, V_p(y_i))$, where $V_p(y_i) = V(y_i)$. The expected distribution given independent input lines, Q, is a Gaussian with the same mean but a different variance: $Q(y_i) = N(y_i, \overline{y_i}, V_q(y_i))$, where $V_q(y_i) = V_q(\sum_j w_{ij} y_j) = \sum_j w_{ij}^2 V(y_j)$. In this case, the G-error is

$$\begin{aligned} G(P,Q) &= \int_{y_i} P(y_i) \log \frac{P(y_i)}{Q(y_i)} \\ &= \left\langle \frac{1}{\sqrt{2\pi V_p(y_i)}} e^{-(y_i - \overline{y_i})^2/2V_p(y_i)} \log \frac{\frac{1}{\sqrt{2\pi V_p(y_i)}} e^{-(y_i - \overline{y_i})^2/2V_p(y_i)}}{\frac{1}{\sqrt{2\pi V_q(y_i)}} e^{-(y_i - \overline{y_i})^2/2V_q(y_i)}} \right\rangle \\ &= \left\langle \frac{1}{\sqrt{2\pi V_p(y_i)}} e^{-(y_i - \overline{y_i})^2/2V_p(y_i)} \left[\log \frac{\sqrt{(V_q)}}{\sqrt{(V_p)}} - \frac{(y_i - \overline{y_i})^2}{2V_p(y_i)} + \frac{(y_i - \overline{y_i})^2}{2V_q(y_i)} \right] \right\rangle \\ &= 0.5 \log \frac{V_q(y_i)}{V_p(y_i)} - \frac{V_p(y_i)}{2V_p(y_i)} + \frac{V_p(y_i)}{2V_q(y_i)} \\ &= 0.5 \left[\frac{V_p(y_i)}{V_q(y_i)} - 1 + \log \frac{V_q(y_i)}{V_p(y_i)} \right] \end{aligned}$$

Differentiating G with respect to y_i yields the following:

$$\begin{aligned}\frac{\partial G}{\partial y_i} &= 0.5\frac{V_q(y_i)-V_p(y_i)}{V_q(y_i)}\left[\frac{1}{V_q(y_i)}\ \frac{\partial V_q(y_i)}{\partial y_i} - \frac{1}{V_p(y_i)}\ \frac{\partial V_p(y_i)}{\partial y_i}\right] \\ &= 0.5\frac{V_q(y_i)-V_p(y_i)}{V_q(y_i)}\ \frac{\partial}{\partial y_i}\log\frac{V_q(y_i)}{V_p(y_i)}\end{aligned}$$

Depending upon the initial conditions, the system either converges to a minimum or a maximum of the term:

$$\log\frac{V_q(y_i)}{V_p(y_i)}$$

To see this, note that if initially $V_q(y_i) > V_p(y_i)$, then the multiplier $\frac{V_q(y_i)-V_p(y_i)}{V_q(y_i)}$ is positive, so $\frac{\partial G}{\partial y_i}$ is in the same direction as $\frac{\partial \log \frac{V_q(y_i)}{V_p(y_i)}}{\partial y_i}$. Moving a small amount in this direction will increase the ratio $\frac{V_q(y_i)}{V_p(y_i)}$, so the multiplier will remain positive, and keep getting larger. If, on the other hand, initially $V_q(y_i) < V_p(y_i)$, then the multiplier is negative, so the variance ratio will be minimized.

We can choose the former solution, which maximizes the ratio $\frac{V_q(y_i)}{V_p(y_i)}$, by just maximizing the objective function $\log\frac{V_q(y_i)}{V_p(y_i)}$. This results in a unit which tries to maximize the sum of the variances on its input lines, while minimizing the variance of its output. This objective function fulfills the desired criteria of an invariant-discovering unit, by finding a set of weights which usually makes the inputs sum to zero, while keeping the weights large.

To generalize the idea to multiple units, we can adopt a mixture model of the input, and assume that the input on each case is modelled by one of a group of competing units. Each unit i responds in proportion to the goodness-of-fit of its linear model:

$$e^{-{x_i}^2/T}$$

A collection of units compete to respond to each pattern. The probability that each unit's model holds is given by:

$$p_i = \frac{e^{-{x_i}^2/T}}{\sum_j e^{-{x_j}^2/T}}$$

We can maximize the same objective function, but using the variances of each unit's total input, x_i:

$$\log\frac{V_q^i(x_i)}{V_p^i(x_i)}$$

which are computed as before for y_i, but this time we compute the variance for the ith unit given that the ith model holds. Thus, we weight the variance contribution for the ith unit on each case by p_i.

Now the total objective function to maximize is:

$$\sum_i \overline{p_i}\ \log\ \frac{V_q^i(x_i)}{V_p^i(x_i)} \quad = \quad \sum_i \overline{p_i}\ \log\ \frac{\sum_j {w_{ij}}^2\ \ V(x_j)}{V(x_i)}$$

Bibliography

Ackley, D., Hinton, G., and Sejnowski, T. (1985). A learning algorithm for boltzmann machines. *Cognitive Science*, 9.

Aczél, J. and Daróczy, Z. (1975). *On measures of information and their characterization*. Academic Press.

Almeida, L. B. (1987). A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *Proceedings, 1st First International Conference on Neural Networks*, pages 609–618, San Diego, CA. IEEE.

Ambros-Ingerson, J., Granger, R., and Lynch, G. (1990). Simulation of paleocortex performs hierarchical clustering. *Science*, 247:1344–1348.

Anderson, S. (1988). Dynamic system categorization with recurrent networks. *Proceedings of the 1988 Connectionist Models Summer School*.

Atick, J. J. and Redlich, A. N. (1989). Predicting ganglion and simple cell receptive field organizations from information theory. Technical Report IASSNS-HEP-89/55, Institute for Advanced Study, Princeton.

Atick, J. J. and Redlich, A. N. (1990). Towards a theory of early visual processing. Technical Report IASSNS-HEP-90/10, Institute for Advanced Study, Princeton.

Bahl, L. R., Brown, P. F., de Souza, P. V., and Mercer, R. L. (1989). A tree-based statistical language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(7):1001–1008.

Baldi, P. and Hornik, K. (1989). Neural networks and principal components analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58.

Ballard, D. H. (1986). Cortical connections and parallel processing: Structure and function. *The Behavioral and Brain Sciences*, 9:67–120.

Barlow, H., Kaushal, T., and Mitchison, G. (1989). Finding minimum entropy codes. *Neural Computation*, 1:412–423.

Barlow, H. B. (1985). Cognitronics: Methods for acquiring and holding cognitive knowledge. Unpublished manuscript.

Barlow, H. B. (1989). Unsupervised learning. *Neural Computation*, 1:295–311.

Barlow, H. B. and Földiák, P. (1989). Adaptation and decorrelation in the cortex. In *The Computing Neuron*, chapter 4, pages 54–72. Addison-Wesley Publishing Corp.

Barrow, H. G. (1987). Learning receptive fields. In *Proceedings of the IEEE first annual conference on Neural Networks*, pages 115–121.

Barto, A. G. and Sutton, R. S. (1983). Neural problem solving. COINS Technical Report 83-03.

Becker, S. and Hinton, G. E. (1989). Spatial coherence as an internal teacher for a neural network. Technical Report CRG-TR-89-7, University of Toronto.

Becker, S. and Hinton, G. E. (1992). A self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355:161–163.

Bialek, W., Ruderman, D. L., and Zee, A. (1991). Optimal sampling of natural images: A design principle for the visual system? In *Advances In Neural Information Processing Systems 3*, pages 363–369. Morgan Kaufmann Publishers.

Bienenstock, E. L., Cooper, L. N., and Munro, P. W. (1982). Theory for the development of neuron selectivity; orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, 2:32–48.

Birch, E. E., Gwiazda, J., and Held, R. (1982). Stereoacuity development for crossed and uncrossed disparities in human infants. *Vison research*, 22:507–513.

Birch, E. E., Stager, D. R., Berry, P., and Everett, M. E. (1990). Prospective assessment of acuity and stereopsis in amblyopic infantile esotropes following early surgery. *Investigative Ophthalmology and Visual Science*, 31(4):758–765.

Blakemore, C. and Cooper, G. (1970). Development of the brain depends on the visual environment. *Nature*, 228:477–478.

Blakemore, C. and van Sluyters, R. (1975). Innate and environmental factors in the development of the kitten's visual cortex. *Journal of Physiology*, 248:663–716.

Bridle, J. S. (1990). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In Touretzky, D. S., editor, *Neural Information Processing Systems, Vol. 2*, pages 111–217, San Mateo, CA. Morgan Kaufmann.

Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Lafferty, R. L., Mercer, R. L., and Roossin, P. S. (1990a). A statistical approach to machine translation. *Computational Linguistics*, 16(2).

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., and Mercer, R. L. (1991). Word sense disambiguation using statistical methods. In *Proceedings of the 29th Annual Meeting of the Association for Compuational Linguistics*, pages 265–270.

Brown, P. F., Della Pietra, V. J., de Souza, P. V., Lai, J. C., and Mercer, R. L. (1990b). Class-based n-gram models of natural language. In *Proceedings of the IBM Natural Language ITL*, pages 283–298.

Carpenter, G. and Grossberg, S. (1983). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision*, 37:54–115.

Carpenter, G. and Grossberg, S. (1987). ART 2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*. Special issue on neural networks.

Cottrell, G., Munro, P., and Zipser, D. (1987). Image compression by back-propagation: an example of extensional programming. ICS Report 8702.

Crawford, M., Smith, E., Harwerth, R., and von Noorden, G. (1984). Stereoblind monkeys have few binocular neurons. *Investigative Opthalmology and Visual Science*, 25(7):779–781.

Crawford, M., von Noorden, G., Meharg, L., Rhodes, J., Harwerth, R., Smith, E., and Miller, D. (1983). Binocular neurons and binocular function in monkeys and children. *Investigative Opthalmology and Visual Science*, 24(4):491–495.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Proceedings of the Royal Statistical Society*, B-39:1–38.

Derrington, A. (1984). Development of spatial frequency selectivity in striate cortex of vision-deprived cats. *Experimental Brain Research*, 55:431–437.

Dodwell, P. (1983). The lie transform group model of visual perception. *Perception and Psychophysics*, 34(1):1–16.

Fallside, F. (1989). On the analysis of multi-dimensional linear predicitve/autoregressive data by a class of single layer connectionist models. In *IEE Conference on Artificial Neural Networks*, pages 176–180.

Földiák, P. (1990). Forming sparse representations by local anti-hebbian learning. *Biological Cybernetics*, 64:165–170.

Frégnac, Y., Shulz, D., Thorpe, S., and Bienenstock, E. (1988). A cellular analogue of visual cortical plasticity. *Nature*, 333(6170):367–370.

Freund, Y. and Haussler, D. (1992). Learning hidden causes. In to appear in *Advances In Neural Information Processing Systems 4*. Morgan Kaufmann Publishers.

Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, 20:121–136.

Fukushima, K. and Miyake, S. (1982). Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, 15:455–469.

Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11:23–63.

Hartman, E. J. (1990). A high storage capacity neural network content addressable memory. Technical Report ACT-NN-173-90, MCC.

Hastie, T. J. and Tibshirani, R. J. (1990). *Generalized Additive Models*. Chapman and Hall, London.

Hebb, D. O. (1949). *The Organization of Behavior.* Wiley, New York.

Held, R., Birch, E. E., and Gwiazda, J. (1980). Stereoacuity of human infants. *Proceedings of the national academy of sciences USA*, 77(9):5572–5574.

Hinton, G. E. (1987). Connectionist learning procedures. Technical Report CMU-CS-87-115, Carnegie-Mellon University, Pittsburgh, PA 15213.

Hinton, G. E. (1989). Deterministic Boltzmann learning performs steepest descent in weight-space. *Neural Computation*, *1*:143–150.

Hinton, G. E. and Sejnowski, T. J. (1986). Learning and relearning in Boltzmann machines. In *Parallel distributed processing: Explorations in the microstructure of cognition*, pages 282–317. Cambridge, MA: MIT Press.

Hinton, G. E., Sejnowski, T. J., and Ackley, D. H. (1984). Boltzmann machines: Constraint satisfaction networks that learn. Technical Report CMU-CS-84-119, Carnegie-Mellon University.

Hirsch, H. V. B. and Spinelli, D. (1970). Visual experience modifies distribution of horizontally and vertically oriented receptive fields in cats. *Science*, 168:869–871.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences U.S.A.*, 79:2554–2558.

Hopfield, J. J. and Tank, D. W. (1985). 'Neural' computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152.

Hubel, D. and Wiesel, T. (1963). Receptive fields of cells in striate cortex of very young, visually inexperienced kittens. *Journal of Neurophysiology*, 26:994–1002.

Huber, P. (1985). Projection pursuit. *The Annals of Statistics*, 13(2):435–475.

Intrator, N. (1990). A neural network for feature extraction. In *Advances in Neural Information Processing Systems 2*, pages 719–726. Morgan Kaufmann Publishers.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3(1).

Jepson, A. D. and Jenkin, M. R. M. (1989). The fast computation of disparity from phase differences. In *Proceedings of the IEEE CVPR*, pages 398–403.

Judd, J. S. (1987). Complexity of connectionist learning with various node functions. COINS Technical Report 87-60.

Klopf, A. H. (1987). Drive-reinforcement learning: A real-time learning mechanism for unsupervised learning. In *IEEE First Annual Conference on Neural Networks, San Diego, California.*

Kohonen, T. (1982). Clustering, taxonomy, and topological maps of patterns. In Lang, M., editor, *Proceedings of the Sixth International Conference on Pattern Recognition*, Silver Spring, MD. IEEE Computer Society Press.

Kohonen, T. (1988). The 'neural' phonetic typewriter. *IEEE Computer*, 21:11–22.

Kohonen, T. and Oja, E. (1976). Fast adaptive formation of orthogonalizing filters and associative memory in recurrent networks of neuron-like elements. *Biological Cybernetics*, 21:85–95.

Kosko, B. (1986). Differential hebbian learning. In Denker, J. S., editor, *Neural Networks for Computing, AIP Conference Proceedings, Snowbird, Utah*, pages 277–282.

Kullback, S. (1959). *Information Theory and Statistics.* Wiley, New York.

Lang, K. J. and Hinton, G. E. (1988). A time-delay neural network architure for speech recognition. Technical Report CMU-CS-88-152, Carnegie-Mellon University.

le Cun, Y. (1987). *Modèles Connexionnistes de l'Apprentissage.* PhD thesis, Université Pierre et Marie Curie, Paris, France.

Lehky, S. R. and Sejnowski, T. J. (1990). Neural model of stereoacuity and depth interpolation based on a distributed representation of stereo disparity. *The Journal of Neuroscience*, 10:2281–2299.

Linsker, R. (1986a). From basic network principles to neural architecture: Emergence of orientation-selective cells. *Proceedings of the National Academy of Sciences USA*, 83:8390–8394.

Linsker, R. (1986b). From basic network principles to neural architecture: Emergence of orientation columns. *Proceedings of the National Academy of Sciences USA*, 83:8779–8783.

Linsker, R. (1986c). From basic network principles to neural architecture: Emergence of spatial opponent cells. *Proceedings of the National Academy of Sciences USA*, 83:7508–7512.

Linsker, R. (1988). Self-organization in a perceptual network. *IEEE Computer,* March, 21:105–117.

Mardia, K. V., Kent, J. T., and Bibby, J. M. (1979). *Multivariate Analysis.* Academic Press.

Maurer, D. and Lewis, T. L. (1992). Visual outcomes in infant cataract. In *Symposium on Infant Vision Research.*

McLachlan, G. J. and Basford, K. E. (1988). Chapters 1 and 2. In McLachlan and Basford, editors, *Mixture models: inference and applications to clustering*, pages 1–69. Marcel Dekker, Inc.

Merzenich, M. (1987). Dynamic neocortical processes and the origins of higher brain functions. In *The Neural and Molecular Bases of Learning, Dahlem Konferenzen*, pages 337–358. John Wiley and Sons Limited.

Merzenich, M. M., Allard, T., Jenkins, W. M., and Recanzone, G. (1988). Self-organizing processes in adult neo-cortex. In *Organization of neural networks: Structures and models.* VCH publishers.

Miller, K., Keller, J., and Stryker, M. (1989). Ocular dominance column development: Analysis and simulation. *Science*, 245:605–615.

Miller, K. D. (1990). Correlation-based models of neural development. In *Neuroscience and Connectionist Theory.* Lawrence Erlbaum Associates.

Minsky, M. L. and Papert, S. (1969). *Perceptrons.* MIT Press, Cambridge, Mass.

Minsky, M. L. and Papert, S. (1987). *Perceptrons: A View from 1987*, chapter Prologue and Epilogue. MIT Press, Cambridge, Mass.

Moody, J. and Darken, C. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, *1*(2):281–294.

Neal, R. M. (1992). Connectionist learning of belief networks. to appear in *Artificial Intelligence.*

Newell, A. and Simon, H. A. (1981). Computer science as empirical inquiry: Symbols and search. In Haugeland, J., editor, *Mind Design.* Cambridge, Mass.: The MIT Press.

Nowlan, S. (1990). The hard versus soft distinction in competitive adaptation. Ph.D. Thesis Proposal, Department of Computer Science, Carnegie-Mellon University.

Oja, E. (1982). A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273.

Oja, E. (1989). Neural networks, principal components, and subspaces. *International Journal Of Neural Systems*, 1(1):61–68.

Olson, C. and Freeman, R. (1980). Profile of the sensitive period for monocular deprivation in kittens. *Experimental Brain Research*, 39:17–21.

Parker, D. B. (1985). Learning-logic. Technical Report TR-47, Center for Computational Research in Economics and Management Science, Massachusetts Institute of Technology, Cambridge, MA.

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

Pearlmutter, B. A. and Hinton, G. E. (1986). G-maximization: An unsupervised learning procedure for discovering regularities. In Denker, J. S., editor, *Neural Networks for Computing: American Institute of Physics Conference Proceedings 151*, pages 333–338.

Peterson, C. and Anderson, J. R. (1987). A mean field theory learning algorithm for neural networks. *Complex Systems*, 1:995–1019.

Peterson, C. and Hartman, E. (1989). Explorations of the mean field theory learning algorithm. *Neural Networks.*

Peterson, C. and Söderberg, B. (1989). A new method for mapping optimization problems onto neural networks. *International Journal of Neural Systems*, 1(3).

Pineda, F. J. (1987). Generalization of back propagation to recurrent and higher order neural networks. In *Proceedings of IEEE Conference on Neural Information Processing Systems*, Denver, Colorado. IEEE.

Plumbley, M. and Fallside, F. (1988). An information-theoretic approach to unsupervised connectionist models. *Proceedings of the 1988 Connectionist Models Summer School.*

Poggio, T. (1989). A theory of networks for approximation and learning. A.I. Memo No. 1140, C.B.I.P. Paper No. 31, MIT Artifiacal Intelligence Laboratory, and Center for Biological Information Processing, Whitaker College.

Press, W., Flannery, B., Teukolsky, S., and Vetterling, W. (1988). *Numerical Recipes in C*. Cambridge University Press.

Pylyshyn, Z. (1981). Complexity and the study of artificial and human intelligence. In Haugeland, J., editor, *Mind Design*. Cambridge, Mass.: The MIT Press.

Reiter, H. O., Waitzman, D. M., and Stryker, M. P. (1986). Cortical activity blockade prevents ocular dominance plasticity in the kitten visual cortex. *Experimental Brain Research*, 65:182–188.

Renals, S. and Rohwer, R. (1989). Phoneme classification experiments using radial basis functions. In *the IEEE International Conference on Neural Networks*.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by back-propagating errors. *Nature*, *323*:533–536.

Rumelhart, D. E. and Zipser, D. (1985). Competitive learning. *Cognitive Science*, 9:75–112.

Rumelhart, D. E. and Zipser, D. (1986). Feature discovery by competitive learning. In *Parallel distributed processing: Explorations in the microstructure of cognition*. Bradford Books, Cambridge, MA.

Sanger, T. (1989a). Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2:459–473.

Sanger, T. (1989b). An optimality principle for unsupervised learning. In Touretzky, D., editor, *Advances in Neural Information Processing Systems*, pages 11–19, Denver 1988. Morgan Kaufmann, San Mateo.

Sanger, T. D. (1989c). Optimal unsupervised learning in feedforward neural networks. M.Sc. Thesis, Department of Electrical Engineering and Computer Science, MIT.

Sejnowski, T. and Tesauro, G. (1989). The Hebb rule for synaptic plasticity: implementations and applications. In *Neural Models of Plasticity*, pages 94–103. Academic Press, San Diego.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423,623–656.

Simard, P. Y., Ottaway, M. B., and Ballard, D. H. (1989). Fixed point analysis for recurrent networks. In Touretsky, D. S., editor, *Advances in Neural Information Processing Systems 1*, pages 149–159, San Matoe, CA. Morgan Kaufmann.

Spinelli, D. and Jensen, F. (1979). Plasticity: The mirror of experience. *Science*, 203:75–78.

Stryker, M. P. and Harris, W. A. (1986). Binocular impulse blockade prevents the formation of ocular dominance columns in cat visual cortex. *The Journal of Neuroscience*, 6(8):2117–2133.

Sur, M., Garraghty, P., and Roe, A. (1988). Experimentally induced visual projections into auditory thalamus and cortex. *Science*, 242:1437–1441.

Sutton, R. S. and Barto, A. (1981). Toward a modern theory of adaptive networks: expectation and prediction. *Psychology Review*, 88:135–170.

Tesauro, G. (1986). Simple neural models of classical conditioning. *Biological Cybernetics*, 55:187–200.

Uttley, A. (1970). The informon: A network for adaptive pattern recognition. *Journal of Theoretical Biology*, 27:31–67.

von der Malsburg, C. (1973). Self-organization of orientation sensitive cells in striate cortex. *Kybernetik*, 14:85–100.

Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University.

Wiesel, T. N. and Hubel, D. H. (1965). Comparison of the effects of unilateral and bilateral eye closure on cortical unit responses in kittens. *Journal of Neurophysiology*, 28:1029–1040.

Yuhas, B. P., Goldstein Jr., M. H., Sejnowski, T. J., and Jenkins, R. E. (1990). Neural network models of sensory integration for improved vowel recognition. In *Proceedings of the IEEE*, volume 78, pages 1658–1668.

Yuille, A., Kammen, D., and Cohen, D. (1989). Quadrature and the development of orientation selective cortical cells by Hebb rules. *Biological Cybernetics*, 61:183–194.

Zemel, R. S. and Hinton, G. E. (1991). Discovering viewpoint-invariant relationships that characterize objects. In *Advances In Neural Information Processing Systems 3*, pages 299–305. Morgan Kaufmann Publishers.

Zipser, D. (1986). Programming networks to compute spatial functions. Technical Report ICS Report 8606, UCSD.