Proc. International Joint Conf. on Neural Networks,
Washington, DC, Jan 1990.

An Unsupervised Learning Procedure that Discovers Surfaces in Random-dot Stereograms

Geoffrey E. Hinton and Suzanna Becker
Department of Computer Science, University of Toronto
10 King's College Road, Toronto M5S 1A4, Canada

Abstract

A major goal of research on unsupervised learning procedures is to discover an objective function that defines the quality of an internal representation without any externally supplied information about the desired outputs of the system. If such a function could be found, it should allow a hierarchy of representations to be organized bottom-up in a time roughly linear in the depth of the network. This would allow much faster learning than supervised procedures which are generally very slow in networks with many layers of hidden units. Following (Gibson, 1950), we propose that a good objective for perceptual learning is to extract higher-order features that are coherent across time or space. This can be done by maximizing the explicit mutual information between parameters extracted from spatially or temporally adjacent parts of the input.

Introduction

The intensity values in one patch of an image contain information about the intensity values in nearby patches, but this information is in a complicated form because the imaging process combines several different physical parameters to produce the intensity of each pixel. If we could first extract important, underlying, intrinsic parameters such as depth, reflectance, or surface orientation, we could then express the mutual information between neighboring patches in a simpler form. This suggests that we could insist on the mutual information being in a simple form and search for the parameters that must be extracted to allow this. One obvious simple form of spatial coherence is for the underlying parameters for one patch to be equal to underlying parameters for the neighboring patch plus some gaussian noise. Another more interesting form of coherence, which would be appropriate for the depth of slanted planes, is for a parameter value extracted from one patch to be the average of the values extracted from neighboring patches. We describe a family of learning procedures that start by making an assumption about the form the coherence will take and then try to discover parameters that are coherent in this way.

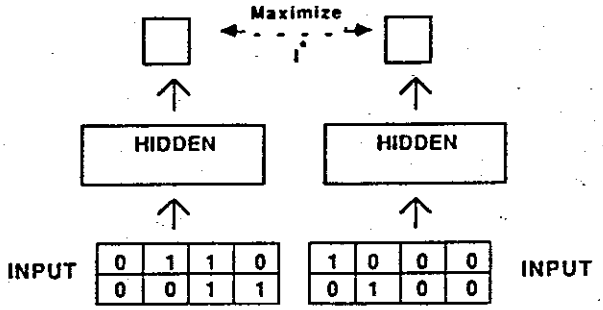

Figure 1: *Two modules that receive input from adjacent, non-overlapping parts of the image. Each module has one layer of hidden units. The learning algorithm adjusts the weights in each module to maximize the mutual information, over the ensemble of training cases, between the states of the two output units.*

Some unpublished results of Peter Brown suggest that a good way to implement this general idea is to try to maximize the explicit mutual information between pairs of parameters extracted from adjacent but non-overlapping parts of the input. The mutual information between two binary variables, a and b, is given by

$$I(a;b) = H(a) + H(b) - H(a,b)$$

where $H(a)$ is the entropy of a, and $H(a,b)$ is the entropy of the joint distribution of a and b. The equation shows that the mutual information between two variables can only be high if each variable has high individual entropy. This is one advantage of mutual information over measures like the correlation between two variables. Mutual information forces each variable to convey a lot of information about the image. Figure 1 shows how this objective function can be used in a multilayer network. The derivative of the mutual information between the outputs of two local modules provides error signals that are back-propagated in order to train the modules.

A very simple example

Our method works well for the task of discovering depth in an ensemble of very simple, binary random-dot stereograms. Each input vector consists of a one dimensional strip from the right image and the corresponding strip from the left image. The right image is purely random and the left image is generated from it by choosing, at random, a single global shift. So the input can be interpreted as an approximation to a one-dimensional stereogram of a fronto-parallel surface at an integer depth. The only local property that is invariant across space is the depth (i.e. the shift). Hence, if one module looks at one area of the two images, and another module looks at another area, the only way they can provide mutual information about each other's outputs is by representing the depth.

We used two global shifts (one pixel rightwards or one pixel leftwards) operating on binary image strips. Each pair of strips was divided into 4 by 2 patches with a gap of one pixel between patches. Each 4 by 2 patch was used as input to a separate module that contained a layer of hidden units and one stochastic binary "output" unit, that used the logistic non-linearity to determine the probability of outputting a 1. Each of the output units tried to maximize the sum of its mutual information (over the ensemble of training cases) with each of the other output units. The derivatives of this objective function are simple to compute using two passes through the training set (Becker and Hinton, 1989). In the first pass we accumulate the probability that each output unit is active, and also all the pairwise joint probabilities. In the second pass we use these accumulated probabilities to compute, for each training case, the derivatives of the objective function w.r.t. the output of a module, and these derivatives are then backpropagated and accumulated to determine the direction of the vector of weight changes within the module at the end of the second pass. The magnitude of the weight change vector is determined by a crude line search along the direction of steepest descent.

With random patterns and a small training set size there is a high probability that units will learn some of the random structure in the data in addition to the shift; as the number of training cases increases, and as we increase the number of modules (and hence the size of the input), sampling error decreases and units become more tuned to shift. The most shift-tuned network we tested had 5 modules and took about 500 passes for the learning to converge on a training set of 1000 random patterns. When we present the complete set of unambiguous binary patterns to a smaller network consisting of just two modules, we usually get output units that are pure shift detectors within about 300 passes through the training set.

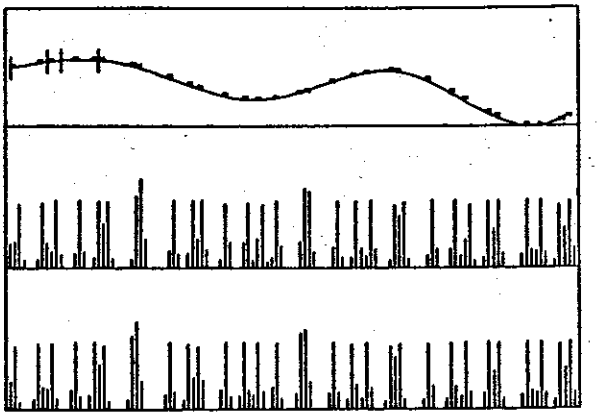

Figure 2: *Part of a cubic spline fitted through seven randomly chosen control points, with randomly located features scattered on it, and the "intensity" values in the two images of the surface strip. The images are made by taking two slightly different parallel projections of the feature points, filtering the projections through a gaussian, and sampling the filtered projections at evenly spaced sample points. The sample values in corresponding patches of the two images are used as the inputs to a module. The boundaries of two neighboring patches are shown on the spline.*

The learning is rather slow for two reasons. First, we are not specifying desired values for the "output" units—we are only specifying that their pairwise mutual information should be high. The derivative of this objective function w.r.t a weight depends on *three* other layers of adaptive weights—one other layer in the same module and two layers in the adjacent module. So in this respect the difficulty of learning resembles back-propagation through four layers of weights. Second, with random starting weights, the initial gradient of the objective function is very small. The convergence speed is greatly increased by using a "bootstrapping" method that starts by applying the objective function between pairs of units in the first hidden layers of pairs of modules until these units are somewhat tuned to the shift. Then the gradients of the mutual information between the output units are much bigger and the objective function can be applied at that layer and the derivatives back-propagated. More globally coherent information can now be provided to the hidden units that failed to find any useful features in the bootstrapping phase.

We compared the performance of the algorithm on a network with 2 modules, and 8 hidden units per module, on the "complete pattern set" with and without 50 bootstrapping iterations. After boot-

strapping, units were highly shift-tuned within 50 learning iterations, and by 250 iterations the algorithm nearly always found a globally optimal solution where the mutual information reached 1 bit (in 43 out of 50 repetitions). Without bootstrapping, while the top-level units always became highly shift-tuned, only in 7 out of 50 repetitions did they converge to the global maximum.

Modules with real-valued outputs

The binary output units we used in our initial experiments are suitable for extracting binary features, but they make it difficult to represent depth in more realistic images that contain smoothly curved surfaces which have real-valued depths. In the following simulations, we use images like those shown in figure 2 and modules with deterministic, real-valued outputs that learn to represent real-valued depths (disparities) with sub-pixel accuracy.

We start by making the following very simple coherence assumption (which will be relaxed later): There is some locally detectable parameter which is approximately constant for nearby patches. So, given two modules A and B that receive input from neighboring patches, we want the output of A, a, to be approximately equal to the output of B, b. We can think of b as a signal that we are trying to predict and a as a noisy version of that signal that is corrupted by additive, independent, gaussian noise. If we assume that both a and b have gaussian distributions, the information (ignoring a factor of 2) that a provides about b is determined by the ratio of two variances:

$$I_{a;b} = \log \frac{V(signal + noise)}{V(noise)} = \log \frac{V(a)}{V(a-b)}$$

So, for a to provide a lot of information about b we need a to have high variance but $a - b$ to have low variance. For symmetry, we actually optimize the following function:

$$I^*_{a;b} = I_{a;b} + I_{b;a} = \log \frac{V(a)}{V(a-b)} + \log \frac{V(b)}{V(a-b)}$$

Some possible variations of this objective function are discussed in (Becker and Hinton, 1989).

Speeding the learning using radial basis functions

Instead of using the bootstrapping method described above to speed the learning, we used an alternative method in which the adaptive hidden units of each module are replaced by a large number of non-adaptive radial basis functions (Moody and Darken, 1989). Each radial basis unit has a "center" that is equal to a typical input vector selected at random, and gives an output which is a gaussian function of the distance between the current input vector and the unit's "center". All the gaussians have the same variance which is chosen by hand. So the only adaptive weights in a module are those from the radial basis units to the output unit. In the experiments with curved surface strips each module had 8 by 2 input units connected to a layer of 100 radial basis units. Every module used exactly the same set of radial basis functions so that we could constrain all the modules to compute the same function.

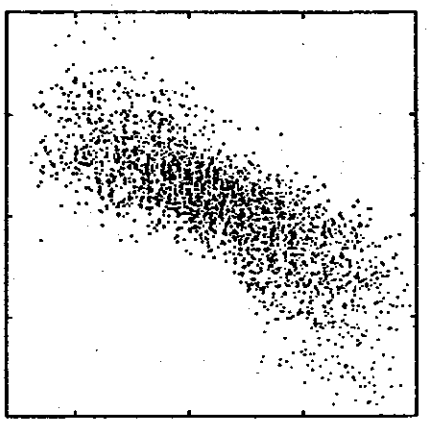

Figure 3: *The activity of the output of a module (vertical axis) as a function of the disparity (horizontal axis) for all 3000 training cases using planar surface strips.*

Discovering real-valued depth for planar surfaces

Figure 2 shows how we generate stereo images of curved surface strips. The same technique can be applied to generate images of planar surfaces with randomly chosen slants. Using 3000 training cases of this simpler type of input, we trained a network that contained 10 modules each of which tried to maximize I^* with the immediately adjacent modules. Each update of the weights involves two complete passes through the training set. In the first pass, we compute the mean and variance of each output value and the mean and variance of each pairwise difference between output values given the current weights. In the second pass we compute the derivatives of I^* for each pair of modules, and use these derivatives to accumulate dI^*/dw for all weights, w, from the radial basis units to the output units. Then we update all the weights in parallel using steepest descent with a simple line search. After each weight update, we average corresponding weights in all the modules in order to enforce the constraint that every module computes exactly the same function of its input vector. After 30 weight updates, the output of a typical module gave a good representation of the disparity as shown in figure 3.

More complex types of coherence

So far, we have used a very simple model of coherence in which an underlying parameter at one location is assumed to be approximately equal to the parameter at a neighboring location. This model is fine for fronto-parallel surfaces but it is far from the best model of slanted or curved surfaces. Fortunately, we can use a far more general model of coherence in which the parameter at one location is assumed to be an unknown linear function of the parameters at nearby locations. The particular linear function that is appropriate can be learned by the network.

We used a network of the type shown in figure 4 (but with 10 modules and with a contextual predictor unit for all modules except the two at the ends). We tried to maximize I^* between the output of each module and the contextual prediction of this output that was produced by computing a linear function of the outputs of one adjacent module on each side. We used weight averaging to constrain this interpolating function to be identical for all modules. We also back-propagated the error derivatives through the interpolating weights. Before applying this new objective function, we first used a bootstrapping stage in which we maximized I^* between adjacent pairs of modules as before, for 30 learning iterations.

After having been trained for 100 iterations on 3000 patterns, the two weights learned for the interpolating function were .55, .54. The output of each of these units is similar to the response profile shown in figure 3, but even more finely depth-tuned. Thus, the interpolating units have learned that the depth at one patch on a planar surface can be approximated by the average of the depths of surrounding patches.

Discovering coherence in curved surfaces

As we introduce curvature in the surfaces, the prediction of depth from neighboring patches becomes more difficult; at regions of high curvature, a simple average of the depths of 2 adjacent patches will under- or over-estimate the true depth. In this case, a better interpolator would base its predictions on more than two local measurements of depth, thereby taking curvature into account.

We trained a network of 10 modules on 1000 of the stereograms of curved surface strips, using the same architecture and objective function as for the planar surface task, for 30 iterations. We then added an interpolating layer; this time, however, the contextual prediction of a given module was a linear function of the outputs of *two* adjacent modules on either side. After 100 iterations, the four weights learned for the interpolating function were $-.04, .64, .65, -.04$. Positive weights are given to inputs coming from the immediately adjacent modules, and smaller negative weights are given to inputs coming from the more distant neighbors. The activity of these units is well tuned to disparity, as shown in figure 5. Given noise-free depth values, the optimal linear interpolator for the surface strips we used is approximately -.2, .7, .7, -.2. But with noisy depth estimates it is better to use an interpolator more like the one the network learned because the noise amplification is determined by the sum of the squares of the weights.

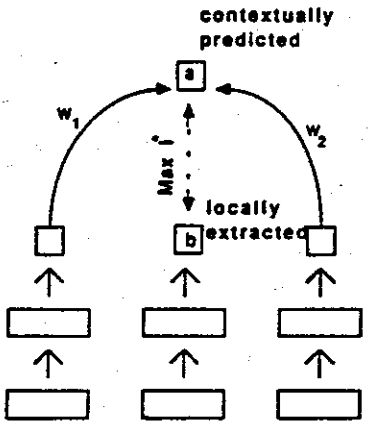

Figure 4: *A network in which the goal of the learning is to maximize the information between the output of a local module and the contextually predicted output that is computed by using a linear function of the outputs of nearby modules.*

Discussion

Discovering how to predict one value from a linear combination of nearby values is equivalent to finding a linear combination of all the values that always equals zero (Richard Durbin, *personal communication*). This amounts to discovering invariant higher-order properties by learning invariance detectors that have low variance even though their input lines have high variances (when weighted by the squares of the 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.

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 real-valued, deterministic version of the G-Maximization learning procedure (Pearlmutter and Hinton, 1986) which finds regularities by maximizing the extent to which the independence assumption is incorrect in predicting the output of a unit. It also has an interesting rela-

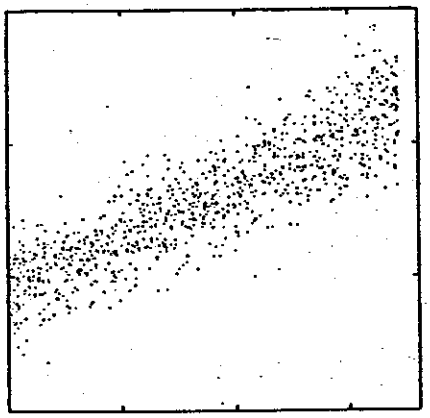

Figure 5: *The output of a unit (vertical axis) as a function of the local disparity (horizontal axis) when trained on 1000 curved surface strips. The unit learned to predict the depth locally extracted from one module as a linear function of the outputs of the 2 adjacent modules on either side.*

tion to Linsker's learning procedure (Linsker, 1988). Linsker assumes the variances and covariances of the activities on the input lines to a unit are fixed (because he does not backpropagate derivatives) and he shows that, with the appropriate gaussian assumptions, the information conveyed by the unit about its input vector is maximized by using weights which *maximize* the ratio of the output variance divided by the sum of the squares of the weights.

We have described the learning procedure for modules which each have a single real-valued output. For modules with several real-valued outputs, the natural way to generalize the objective function is to replace the variance by the determinant of the covariance matrix. It remains to be seen whether this causes unacceptable problems with the learning speed and whether it can be modified to avoid the difficulties that arise as the covariance matrix becomes singular.

We have also ignored the ubiquitous problem of discontinuities. Images of real scenes have strong local coherence punctuated by discontinuities. We do not want our learning procedure to smear out the strong local coherence by trying to convey information across the discontinuities. We would prefer a module to make accurate predictions in continuity cases and no predictions in other cases rather than making rather inaccurate predictions in all cases. We can achieve this by letting each module use a mixture of two gaussian models to predict the output of a neighboring module. One part of the mixture model is for continuity cases, and the other part is for discontinuity cases. During learning, the module computes the probability that the current case is an example of continuity by comparing the probability densities, under both its continuity and its discontinuity models, of the observed output of its neighbor. The contribution to the accumulated gradient is then made proportional to the probability that the current case is a continuity case. This means that clear cases of discontinuity do not affect the weights learned by the continuity model.

In this paper, we have used coherence across space, but the same techniques could be applied to coherence across time. The procedure we have described has several appealing properties. First, it builds into the objective function (and the architecture) a type of prior knowledge that is strongly constraining but widely applicable to perceptual tasks. Second, using the bootstrapping approach it may be possible to train deep networks fairly rapidly, provided the domain is such that the very high-order features that exhibit very long-range coherence can be built out of lower-order features that exhibit shorter range coherence.

Acknowledgements

We thank Peter Brown, Francis Crick, Allan Jepson, and Barak Pearlmutter for helpful discussions. This research was supported by grants from DuPont, the Ontario Information Technology Research Center, and the Canadian National Science and Engineering Research Council. Geoffrey Hinton is a fellow of the Canadian Institute for Advanced Research.

References

Becker, S. and Hinton, G. E. (1989). Using spatial coherence as an internal teacher for a neural network. Technical Report in preparation, University of Toronto.

Gibson, J. J. (1950). *The perception of the visual world.* Houghton Mifflin, Boston, Mass.

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

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

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.