From: AAAI Technical Report FS-93-04. Compilation copyright © 1993, AAAI (www.aaai.org). All rights reserved.

Developing Population Codes For Object Instantiation Parameters

Richard S. Zemel
University of Toronto &
The Salk Institute, CNL
10010 North Torrey Pines Rd.
La Jolla, CA 92037 USA
zemel@helmholtz.sdsc.edu

Geoffrey E. Hinton
6 King's College Rd.
Computer Science Department
University of Toronto
Toronto, ONT M5S 1A4 Canada
hinton@cs.toronto.edu

Abstract

An efficient and useful representation for an object viewed from different positions is in terms of its instantiation parameters. We show how the Minimum Description Length principle (MDL) can be used to train the hidden units of a neural network to develop a population code for the instantiation parameters of an object in an image. Each hidden unit has a location in a low-dimensional *implicit* space. If the hidden unit activities form a standard shape (a bump) in this space, they can be cheaply encoded by the center of this bump. So the weights from the input units to the hidden units in a self-supervised network are trained to make the activities form a bump. The coordinates of the hidden units in the implicit space are also learned, thus allowing flexibility, as the network develops separate population codes when presented with different objects.

1 Introduction

An important area in which machine learning can contribute to computer vision is in the development of appropriate representations for vision problems. Unsupervised learning is an especially relevant paradigm, since large numbers of labeled samples are often not available. These algorithms can be directly useful if they can discover low-dimensional representations of a set of images that capture important underlying variables in the images.

We have developed a framework based on the Minimum Description Length (MDL) principle (Rissanen, 1989) that leads to several new unsupervised learning algorithms. In this paper we introduce the general framework, and motivate a particular form of representing images of objects. We then show how an algorithm defined within the framework can be used to develop these kind of representations of the underlying parameters in a set of simple images.

2 An MDL framework for unsupervised learning

Most existing unsupervised learning algorithms can be understood using MDL. Given an ensemble of input vectors, the aim of the learning algorithm is to find a method of coding each input vector that minimizes the total cost, in bits, of communicating the input vectors to a receiver. There are three terms in the total description length:

- The **code-cost** is the number of bits required to communicate the code that the algorithm assigns to each input vector.
- The **model-cost** is the number of bits required to specify how to reconstruct input vectors from codes (e.g., the hidden-to-output weights in Figure 2).
- The **reconstruction-error** is the number of bits required to fix up any errors that occur when the input vector is reconstructed from its code.

For example, in competitive learning (vector quantization), the code is the identity of the winning hidden unit, so by limiting the system to N units we limit the average code-cost to at most $\log_2 N$ bits. The reconstruction-error is proportional to the squared difference between the input vector and the weight-vector of the winner, and this is what competitive learning algorithms minimize. The model-cost is usually ignored.

The representations produced by vector quantization contain very little information about the input (at most $\log_2 N$ bits). To get richer representations we must allow many hidden units to be active at once and to have varying activity levels. Principal components analysis (PCA) achieves this for linear mappings from inputs to codes. It can be viewed as a version of MDL in which we limit the code-cost by only having a few hidden units, and ignoring the model-cost and the accuracy with which the hidden activities must be coded. A self-supervised network (see Figure 2) that tries to reconstruct the input vector on its output units will perform a version of PCA if the output units are linear. Novel and interesting unsupervised learning algorithms can be obtained by considering various alternative methods of communicating the hidden activities. The algorithms can all be implemented by backpropagating the derivative of the code-cost for the hidden units in addition to the derivative of the reconstruction-error backpropagated from the output units.

3 Representing instantiation parameters with population codes

In order to communicate the codes, or hidden activities of the network, we need to assume some sort of model

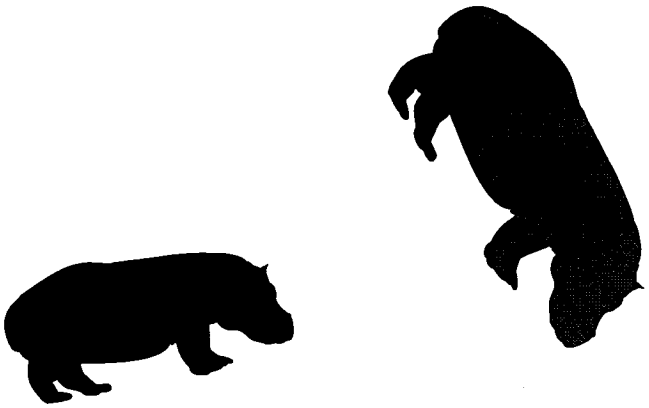

Figure 1: Two example images where only a few dimensions of variability underlie seemingly unrelated vectors of pixel intensity values. Here the underlying dimensions are the 4 instantiation parameters (position, orientation, and size) of the 2D hippo.

for them. The MDL framework forces us to make explicit our prior expectations for these codes; different models lead to different code-cost terms in the description length. When the model is appropriate for a given dataset, the algorithm can discover succinct descriptions, as the hidden units come to correspond to characteristics or constraints under which the source that generates the data operates.

3.1 Instantiation parameters

In many cases, these constraints are continuous-valued dimensions, so that each datum is a point on an underlying *constraint surface*. For example, a set of points constrained to lie on a complicated curve can be described by their (x, y)-coordinates. A more compact description of each point is its position s along the curve. This simple example shows that uncovering the underlying constraint can lead to *dimensionality-reduction*, or a recoding of the data in a lower-dimensional space.

Another example is shown in Figure 1. If we imagine a high-dimensional input representation produced by digitizing these images, many bits would be required to describe each instance of the hippopotamus. But after seeing several such instances, we can describe any particular one by describing a canonical 2D hippo, and then just specifying the four 2D *instantiation parameters*: position, orientation, and scale.

It can be quite useful to represent an image in terms of the instantiation parameters of the objects contained in it. The instantiation parameters describe the variability due to seeing the same object from different viewpoints. For rigid 2D objects, the transformation from the instantiation parameters of the object to its component features is linear, which allows for efficient recognition schemes (e.g., TRAFFIC (Zemel, Mozer, and Hinton, 1990)). If we know the spatial relations between the features of a given object, specifying the object instantiation parameters allows us to predict the feature instantiation parameters, and vice versa. Also, this representation scheme allows one to describe the image at several levels of description, as hierarchical representations of features and objects can be recursively constructed based on their instantiation parameters.

3.2 Population codes

The general goal is to find and represent the m-dimensional constraint surface, such as the instantiation parameters, underlying the n-dimensional data set, given a set of samples drawn from the constraint surface. A representation can be said to capture the surface if samples drawn nearby on the surface are assigned similar representations.

A point on this surface may be represented in many different forms. In the algorithm described here, we employ a coarse-coding of the underlying space, or a *population code*: each unit represents a local range of points on a surface, and the activity of several units together convey a particular point.

This representational form has several merits. It is a form of sparse distributed coding; it thus possesses the efficiency of a localist encoding while also possessing the robustness and capacity of a more distributed encoding. The resulting representation is fault-tolerant, in that each unit need not function perfectly in order to represent a set of inputs. More than one value can be represented at a time, and each unit has a small dynamic range. Also, as first suggested by Barlow (1972), this form of representation seems to be a property of most cortical cells.

3.3 Introducing redundancy into the representation

Population codes, however, would seemingly be difficult to develop in the MDL framework. Any method that communicates each hidden activity separately and independently will tend to lead to *factorial* codes because any mutual information between hidden units will cause redundancy in the communicated message, so the pressure to keep the message short will squeeze out the redundancy.

In a population code, however, the activities of the representation units would be highly correlated. The representation divides the surface into overlapping patches, so units representing nearby patches will have similar activities. Our aim here is to show how the MDL approach can be used to develop representations of underlying constraint surfaces, such as object instantiation parameters, in which the activities of the representation units are highly correlated.

4 Learning population codes

Population codes involve *three* quite different spaces: the input-vector space; the hidden-vector space; and another, low-dimensional space which we call the *implicit space*. Most topographic-map formation algorithms, such as the Kohonen algorithm (Kohonen, 1982), define the topology of this implicit space through neighborhood relations; this topology remains fixed while the network learns to associate points in implicit space with points in input space. In Saund's (1989) model, the patterns of activity of the hidden units in a self-supervised network

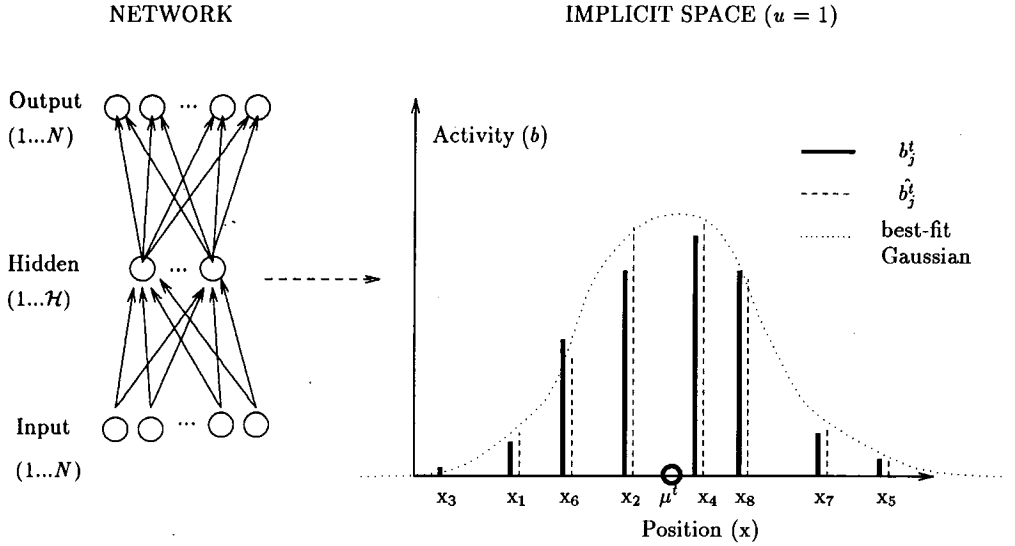

Figure 2: Each of the $\mathcal{H}$ hidden units in the self-supervised network has an associated position in implicit space. Here we show a 1D implicit space. The activity b_j^t of each hidden unit j on case t is shown by a solid line. The network fits the best Gaussian to this pattern of activity in implicit space. The predicted activity, $\hat{b}_j^t$, of unit j under this Gaussian is based on the distance from $\mathbf{x_j}$ to the mean $\mu^{\mathbf{t}}$; it serves as a target for b_j^t.

are trained to form a one-dimensional Gaussian bump, such that the bump mean corresponds to the position in an underlying dimension of the inputs. Here the implicit space is statically determined by the ordering of the hidden units. Recent variations of Kohonen's algorithm (e.g., Martinetz and Schulten, 1991) learn the implicit coordinates, but this occurs after the mapping from input to hidden space has been learned.

In our model, each hidden unit has weights coming from the input units that determine its activity level. But in addition to these weights, it has another set of adjustable parameters that represent its coordinates in the implicit space. To determine what is represented by a vector of hidden activities, we average together the implicit coordinates of the hidden units, weighting each coordinate vector by the activity level of the unit.

Suppose, for example, that each hidden unit is connected to an 8x8 retina and has 2 implicit coordinates that represent the position of a particular kind of shape on the retina. Given a population of such units all broadly-tuned to different positions we can represent any particular instance of the shape by the relative activity levels in the population. If we plot the hidden activity levels in the implicit space (not the input space), we would like to see a bump of activity of a standard shape (e.g., a Gaussian) whose center represents the instantiation parameters of the shape (Figure 2 depicts this for a 1D implicit space).[1] If the activities form a perfect Gaussian bump of fixed variance we can communicate them by simply communicating the coordinates of the mean of the Gaussian; this is very economical if there are many less implicit coordinates than hidden units.

[1]Each hidden unit is intended to act as a Radial Basis Function (RBF). Unlike standard RBFs, however, the RBF activity serves as a target for the activity levels, and is determined by distance in a space other than the input space.

It is important to realize that the activity of a hidden unit is actually caused by the input-to-hidden weights, but by setting these weights appropriately we can make the activity match the height under the Gaussian in implicit space. If the activity bump is not quite perfect, we must also encode the *bump-error*—the misfit between the actual activity levels and the levels predicted by the Gaussian bump. The cost of encoding this misfit is what forces the activity bump in implicit space to approximate a Gaussian.

Currently, we ignore the model-cost, so the description length to be minimized is:

$$\begin{aligned} E^t &= B^t + R^t \\ &= \frac{1}{2V_B}\sum_{j=1}^{\mathcal{H}}(b_j^t - \hat{b}_j^t)^2 + \frac{1}{2V_R}\sum_{k=1}^{N}(a_k^t - c_k^t)^2 \quad (1) \end{aligned}$$

where a, b, c are the activities of units in the input, hidden, and output layers, respectively, V_B and V_R are the fixed variances used for coding the bump-errors and the reconstruction-errors, and the other symbols are explained in the caption of Figure 2.

We compute the actual activity of a hidden unit, b_j^t, as a normalized exponential of its total input.[2] Its expected activity is its normalized value under the predicted Gaussian bump:

$$\hat{b}_j^t = \exp(-(\mathbf{x_j} - \mu^{\mathbf{t}})^2/2\sigma^2)/\sum_{i=1}^{\mathcal{H}}\exp(-(\mathbf{x_i} - \mu^{\mathbf{t}})^2/2\sigma^2) \quad (2)$$

On each case, a separate minimization determines $\mu^{\mathbf{t}}$; it is the position in implicit space that minimizes B^t given $\{\mathbf{x_j}, b_j^t\}$. We assume for now that σ is fixed

[2]$b_j^t = \exp(net_j^t)/\sum_{i=1}^{H}\exp(net_i^t)$, where net_j^t is the net input into unit j on case t.

throughout training. The network has full inter-layer connectivity, and linear output units. Both the network weights and the implicit coordinates of the hidden units are adapted to minimize E.

5 Experimental Results

In the first experiment, each 8x8 real-valued input image contained an instance of a simple shape in a random (x, y)-position. The network began with random weights, and each of 100 hidden units had random 2D implicit coordinates. We trained the weights and positions using conjugate gradient on 400 examples, and tested generalization with a test set of 100 examples. The network converged after 25 epochs. Each hidden unit developed a receptive field so that it responded to inputs in a limited neighborhood that corresponded to its learned position in implicit space (see Figure 3). The set of hidden units covered the range of possible positions. Each test image gave rise to a Gaussian bump of activity in implicit space, where the bump mean corresponded to the instantiation parameters of the shape in that image.

In a second experiment, we also varied the orientation of the shape, and we gave each hidden unit three implicit coordinates. The network converged after 60 epochs of training on 1000 images. The hidden unit activities again formed a population code that allowed the input to be accurately reconstructed. The three dimensions of the implicit space corresponded to a recoding of the object instantiation parameters, such that smooth changes in the object's parameters produced similar changes in the implicit space codes.

A third experiment employed a training set where each image contained either a horizontal or vertical bar, in some random position. The hidden units formed an interesting 2D implicit space in this case: one set of hidden units moved to one corner of the space, and represented instantiation parameters of instances of one shape, while the other group moved to an opposite corner and represented parameters of the other shape (Figure 4). This type of representation would be difficult to learn in a Kohonen network (Kohonen, 1982); the fact that the hidden units can *learn* their implicit coordinates allows much more flexibility than any system in which these coordinates are fixed in advance.

6 Conclusions and Current Directions

We have shown how MDL can be used to develop non-factorial, redundant representations. Instead of communicating each hidden unit activity independently we communicate the location of a Gaussian bump in a low-dimensional implicit space. If hidden units are appropriately tuned in this space their activities can then be inferred from the bump location. When the underlying dimensions of variability in a set of input images are the instantiation parameters of an object, this implicit space comes to correspond to these parameters.

Our method can easily be applied to networks with multiple hidden layers, where the implicit space is constructed at the last hidden layer before the output and derivatives are then backpropagated; this allows the implicit space to correspond to arbitrarily high-order input properties. This makes the algorithm potentially useful in many vision problems, where the relevant underlying variables correspond to higher-order non-linear properties of the input.

Several limitations exist in the algorithm as described here. First, fitting a single Gaussian to the pattern of activity in implicit space presumes that only one instance of an object exists in the image. A straightforward extension would be to fit a bump in each local image patch, and include the ability to fit a "no-object" bump.

A second limitation is the need to predetermine the number of dimensions in implicit space; we are currently working on an extension that will allow the learning algorithm to determine for itself the appropriate dimensionality. We start with many dimensions but include the cost of specifying $\mu^{\mathbf{t}}$ in the description length. This depends on how many implicit coordinates are used. If all of the hidden units have the same value for one of the implicit coordinates, it costs nothing to communicate that value for each bump. In general, the cost of an implicit coordinate depends on the ratio between its variance (over all the different bumps) and the accuracy with which it must be communicated. So the network can save bits by reducing the variance for unneeded coordinates. This creates a smooth search space for determining how many implicit coordinates are really needed.

Acknowledgements

We thank Peter Dayan, Sue Becker, Chris Williams, and members of the University of Toronto Connectionist Research Group for helpful discussions and comments. This research was supported by a grant from the Information Technology Research Center of Ontario to Geoffrey Hinton. Some of the work described here was done at the Computational Neurobiology Laboratory at the Salk Institute. We thank Terry Sejnowski for this support.

References

Barlow, H. B. (1972). Single units and sensation: A neuron doctrine for perceptual psychology? *Perception*, 1:371–394.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69.

Martinetz, T. and Schulten, K. (1991). A 'neural gas' network learns topologies. In *Proceedings of ICANN-91*, pages 397–402.

Rissanen, J. (1989). *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing Co., Singapore.

Saund, E. (1989). Dimensionality-reduction using connectionist networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(3):304–314.

Zemel, R. S., Mozer, M. C., and Hinton, G. E. (1990). TRAFFIC: Object recognition using hierarchical reference frame transformations. In *Advances in Neural Information Processing Systems 2*, pages 266–273, San Mateo, CA. Morgan Kaufmann.

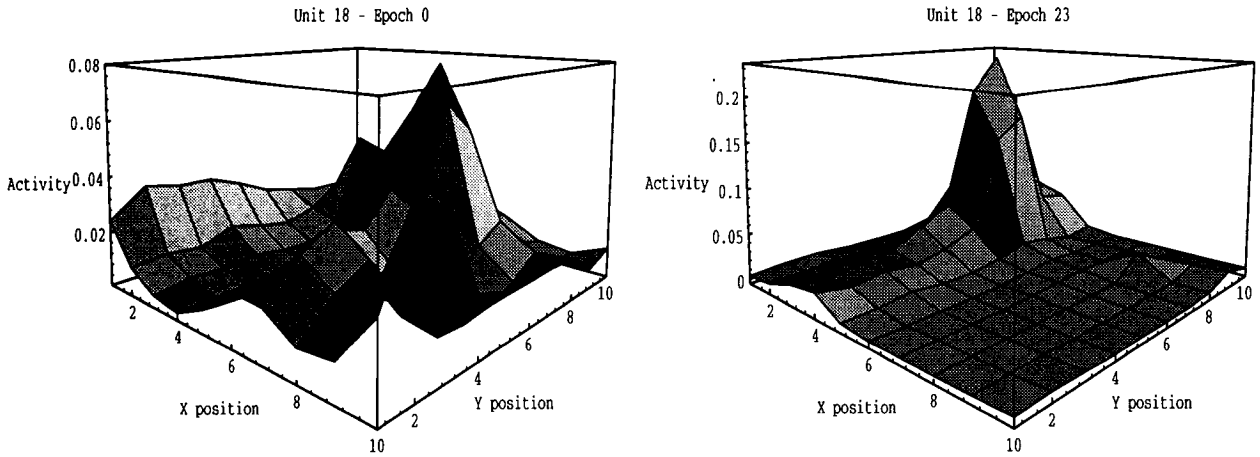

Figure 3: This figure shows the receptive field in implicit space for a hidden unit. The left panel shows that before learning, the unit responds randomly to 100 different test patterns, generated by positioning a shape in the image at each point in a 10x10 grid. Here the 2 dimensions in implicit space correspond to x and y positions. The right panel shows that after learning, the hidden unit responds to objects in a particular position, and its activity level falls off smoothly as the object position moves away from the center of the learned receptive field.

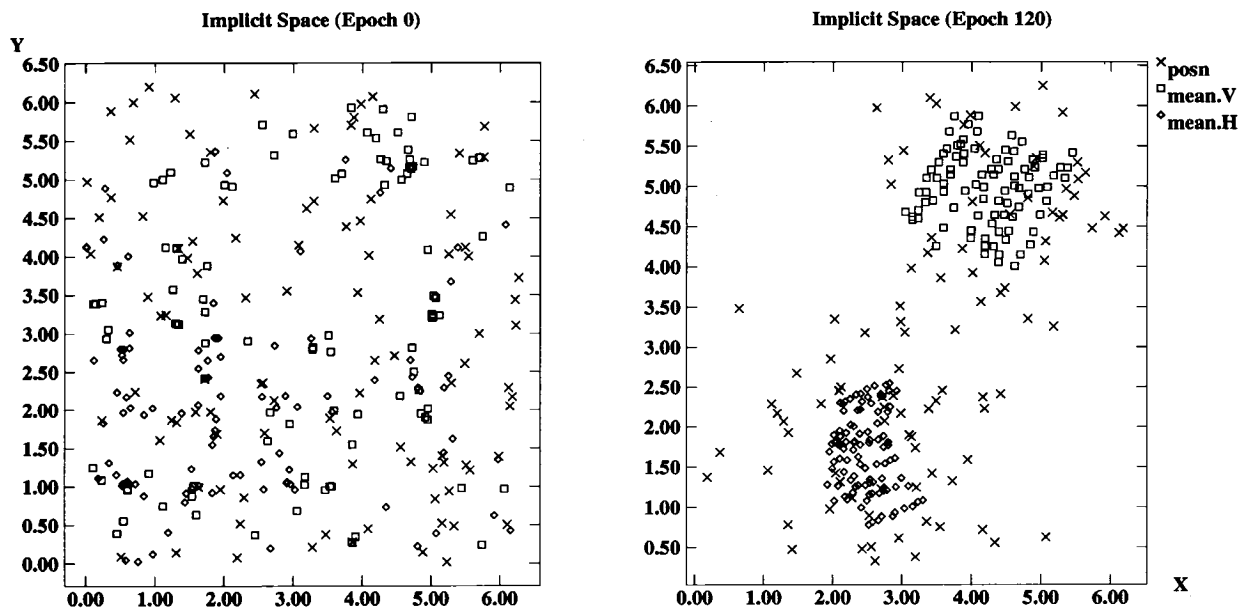

Figure 4: This figure shows the positions of the hidden units and the means in the 2D implicit space before and after training on the horizontal/vertical task. The means in the top right of the second plot all correspond to images containing vertical bars, while the other set correspond to horizontal bar images. Note that some hidden units are far from all the means; these units do not play a role in the coding of the input, and are free to be recruited for other types of input cases.