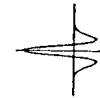


Excerpt re. Coarse Coding from:



Computational Models of Cognition and Perception

Editors

Jerome A. Feldman
Patrick J. Hayes
David E. Rumelhart

PARALLEL DISTRIBUTED PROCESSING

Explorations in the Microstructure of Cognition

Volume 1: Foundations

Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations, by David E. Rumelhart, James L. McClelland, and the PDP Research Group

Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological Models, by James L. McClelland, David E. Rumelhart, and the PDP Research Group

Neurophilosophy: Toward a Unified Science of the Mind-Brain, by Patricia S. Churchland

Qualitative Reasoning About Physical Systems, edited by Daniel G. Bobrow

Visual Cognition, edited by Steven Pinker

David E. Rumelhart James L. McClelland
and the PDP Research Group

Chisato Asanuma Alan H. Kawamoto Paul Smolensky
Francis H. C. Crick Paul W. Munro Gregory O. Stone
Jeffrey L. Elman Donald A. Norman Ronald J. Williams
Geoffrey E. Hinton Daniel E. Rabin David Zipser
Michael I. Jordan Terrence J. Sejnowski

Institute for Cognitive Science
University of California, San Diego

A Bradford Book
The MIT Press
Cambridge, Massachusetts
London, England

Creating New Concepts

Any plausible scheme for representing knowledge must be capable of learning novel concepts that could not be anticipated at the time the network was initially wired up. A scheme that uses local representations must first make a discrete decision about *when* to form a new concept, and then it must find a spare hardware unit that has suitable connections for implementing the concept involved. Finding such a unit may be difficult if we assume that, after a period of early development, new knowledge is incorporated by changing the strengths of the existing connections rather than by growing new ones. If each unit only has connections to a small fraction of the others, there will probably not be any units that are connected to just the right other ones to implement a new concept. For example, in a collection of a million units each connected at random to ten thousand others, the chance of there being *any* unit that is connected to a particular set of 6 others is only one in a million.

In an attempt to rescue local representations from this problem, several clever schemes have been proposed that use two classes of units. The units that correspond to concepts are not directly connected to one another. Instead, the connections are implemented by indirect pathways through several layers of intermediate units (Fahlman, 1980; Feldman, 1982). This scheme works because the number of *potential* pathways through the intermediate layers far exceeds the total number of physical connections. If there are k layers of units, each of which has a fan-out of n connections to randomly selected units in the following layer, there are n^k potential pathways. There is almost certain to be a pathway connecting any two concept-units, and so the intermediate units along this pathway can be dedicated to connecting those two concept-units. However, these schemes end up having to dedicate several intermediate units to each effective connection, and once the dedication has occurred, all but one of the actual connections emanating from each intermediate unit are wasted. The use of several intermediate units to create a single effective connection may be appropriate in switching networks containing elements that have units with relatively small fan-out, but it seems to be an inefficient way of using the hardware of the brain.

The problems of finding a unit to stand for a new concept and wiring it up appropriately do not arise if we use distributed representations. All we need to do is modify the interactions between units so as to create a new stable pattern of activity. If this is done by modifying a large number of connections very slightly, the creation of a new pattern need not disrupt the existing representations. The difficult problem is

to choose an appropriate pattern for the new concept. The effects of the new representation on representations in other parts of the system will be determined by the units that are active, and so it is important to use a collection of active units that have roughly the correct effects. Fine-tuning of the effects of the new pattern can be achieved by slightly altering the effects of the active units it contains, but it would be unwise to choose a *random* pattern for a new concept because major changes would then be needed in the weights, and this would disrupt other knowledge. Ideally, the distributed representation that is chosen for a new concept should be the one that requires the least modification of weights to make the new pattern stable and to make it have the required effects on other representations.

Naturally, it is not necessary to create a new stable pattern all in one step. It is possible for the pattern to emerge as a result of modifications on many separate occasions. This alleviates an awkward problem that arises with local representations: The system must make a discrete all-or-none decision about when to create a new concept. If we view concepts as stable patterns, they are much less discrete in character. It is possible, for example, to differentiate one stable pattern into two closely related but different variants by modifying some of the weights slightly. Unless we are allowed to clone the hardware units (and all their connections), this kind of gradual, conceptual differentiation is much harder to achieve with local representations.

One of the central problems in the development of the theory of distributed representation is the problem of specifying the exact procedures by which distributed representations are to be learned. All such procedures involve connection strength modulation, following "learning rules" of the type outlined in Chapter 2. Not all the problems have been solved, but significant progress is being made on these problems. (See the chapters in Part II.)

DISTRIBUTED REPRESENTATIONS THAT WORK EFFICIENTLY

In this section, we consider some of the technical details about the implementation of distributed representations. First, we point out that certain distributed representation schemes can fail to provide a sufficient basis for differentiating different concepts, and we point out what is required to avoid this limitation. Then, we describe a way of using distributed representations to get the most information possible out of a simple network of connected units. **The central result is a surprising one: If you want to encode features accurately using as few units as**

possible, it pays to use units that are very coarsely tuned, so that each feature activates many different units and each unit is activated by many different features. A specific feature is then encoded by a pattern of activity in many units rather than by a single active unit, so coarse coding is a form of distributed representation.

To keep the analysis simple, we shall assume that the units have only two values, on and off.³ We shall also ignore the dynamics of the system because the question of interest, for the time being, is how many units it takes to encode features with a given accuracy. We start by considering the kind of feature that can be completely specified by giving a type (e.g., line-segment, corner, dot) and the values of some continuous parameters that distinguish it from other features of the same type (e.g., position, orientation, size.) For each type of feature there is a space of possible instances. Each continuous parameter defines a dimension of the feature space, and each particular feature corresponds to a point in the space. For features like dots in a plane, the space of possible features is two-dimensional. For features like stopped, oriented edge-segments in three-dimensional space, the feature space is six-dimensional. We shall start by considering two-dimensional feature spaces and then generalize to higher dimensionalities.

Suppose that we wish to represent the position of a single dot in a plane, and we wish to achieve high accuracy without using too many units. We define the accuracy of an encoding scheme to be the number of different encodings that are generated as the dot is moved a standard distance through the space. One encoding scheme would be to divide the units into an X group and a Y group, and dedicate each unit to encoding a particular X or Y interval as shown in Figure 2. A given dot would then be encoded by activity in two units, one from each group, and the accuracy would be proportional to the number of units used. Unfortunately, there are two problems with this. First, if two dots have to be encoded at the same time, the method breaks down. The two dots will activate two units in each group, and there will be no way of telling, from the active units, whether the dots were at (x_1, y_1) and (x_2, y_2) or at (x_1, y_2) and (x_2, y_1) . This is called the *binding problem*. It arises because the representation does not specify what goes with what.

³ Similar arguments apply with multivalued activity levels, but it is important not to allow activity levels to have arbitrary precision because this makes it possible to represent an infinite amount of information in a single activity level. Units that transmit a discrete impulse with a probability that varies as a function of their activation seem to approximate the kind of precision that is possible in neural circuitry (see Chapters 20 and 21).

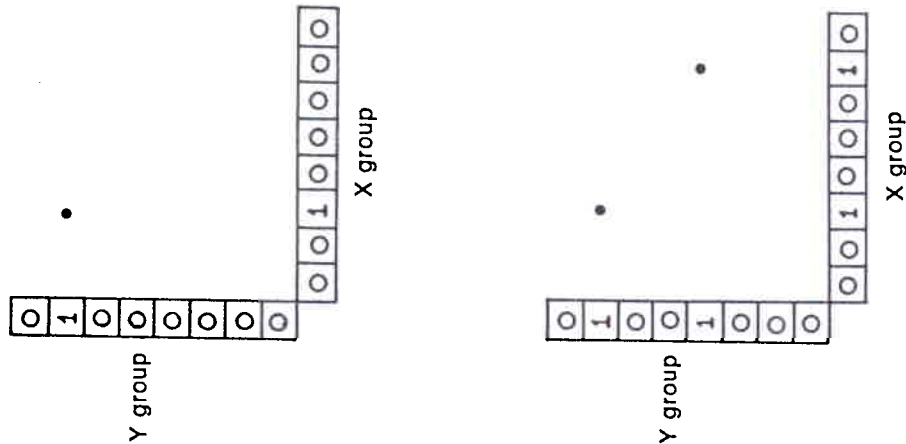


FIGURE 2. A: A simple way of using two groups of binary units to encode the position of a point in a two-dimensional space. The active units in the X and Y groups represent the x- and y-coordinates. B: When two points must be encoded at the same time, it is impossible to tell which x-coordinate goes with which y-coordinate.

The second problem arises even if we allow only one point to be represented at a time. Suppose we want certain representations to be associated with an overt response, but not others: We want (x_1, y_1) and (x_2, y_2) to be associated with a response, but not (x_1, y_2) or (x_2, y_1) . We cannot implement this association using standard weighted connections to response units from units standing for the values on the two dimensions separately. For the unit for x_1 and the unit for x_2 would both have to activate the response, and the unit for

$\gamma 1$ and the unit for $\gamma 2$ would both have to activate the response. There would be no way of preventing the response from being activated when the unit for $x 1$ and the unit for $\gamma 2$ were both activated. This is another aspect of the binding problem since, again, the representation fails to specify what must go with what.

In a conventional computer it is easy to solve the binding problem. We simply create two records in the computer memory. Each record contains a pair of coordinates that go together as coordinates of one dot, and the binding information is encoded by the fact that the two coordinate values are sitting in the same record (which usually means they are sitting in neighboring memory locations). In parallel networks it is much harder to solve the binding problem.

Conjunctive Encoding

One approach is to set aside, in advance, one unit for each possible combination of X and Y values. This amounts to covering the plane with a large number of small, nonoverlapping zones and dedicating a unit to each zone. A dot is then represented by activity in a single unit so this is a *local representation*. The use of one unit for each discriminable feature solves the binding problem by having units which stand for the conjunction of values on each of two dimensions. In general, to permit an arbitrary association between particular combinations of features and some output or other pattern of activation, some conjunctive representation may be required.

However, this kind of local encoding is very expensive. It is much less efficient than the previous scheme because the accuracy of pinpointing a point in the plane is only proportional to the square root of the number of units. In general, for a k -dimensional feature space, the local encoding yields an accuracy proportional to the k^{th} root of the number of units. Achieving high accuracy without running into the binding problem is thus very expensive.

The use of one unit for each discriminable feature may be a reasonable encoding if a very large number of features are presented on each occasion, so that a large fraction of the units are active. However, it is a very inefficient encoding if only a very small fraction of the possible features are presented at once. The average amount of information conveyed by the state of a binary unit is 1 bit if the unit is active half the time, and it is much less if the unit is only rarely active.⁴ It would

⁴ The amount of information conveyed by a unit that has a probability of p of being on is $-p \log p - (1 - p) \log(1 - p)$.

therefore be more efficient to use an encoding in which a larger fraction of the units were active at any moment. This can be done if we abandon the idea that each discriminable feature is represented by activity in a single unit.

Coarse Coding

Suppose we divide the space into larger, overlapping zones and assign a unit to each zone. For simplicity, we will assume that the zones are circular, that their centers have a uniform random distribution throughout the space, and that all the zones used by a given encoding scheme have the same radius. The question of interest is how accurately a feature is encoded as a function of the radius of the zones. If we have a given number of units at our disposal is it better to use large zones so that each feature point falls in many zones, or is it better to use small zones so that each feature is represented by activity in fewer but more finely tuned units?

The accuracy is proportional to the number of different encodings that are generated as we move a feature point along a straight line from one side of the space to the other. Every time the line crosses the boundary of a zone, the encoding of the feature point changes because the activity of the unit corresponding to that zone changes. So the number of discriminable features along the line is just twice the number of zones that the line penetrates.⁵ The line penetrates every zone whose center lies within one radius of the line (see Figure 3). This number is proportional to the radius of the zones, r , and it is also proportional to their number, n . Hence the accuracy, a , is related to the number of zones and to their radius as follows:

$$a \propto nr.$$

In general, for a k -dimensional space, the number of zones whose centers lie within one radius of a line through the space is proportional to the volume of a k -dimensional hypercylinder of radius r . This volume is equal to the length of the cylinder (which is fixed) times its $(k - 1)$ -dimensional cross-sectional area which is proportional to r^{k-1} .

⁵ Problems arise if you enter and leave a zone without crossing other zone borders in between because you revert to the same encoding as before, but this effect is negligible if the zones are dense enough for there to be many zones containing each point in the space.

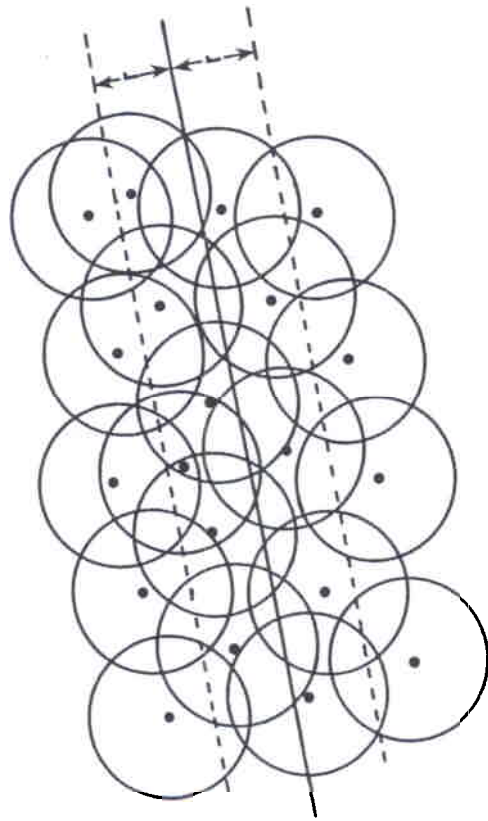


FIGURE 3. The number of zone boundaries that are cut by the line is proportional to the number of zone centers within one-zone radius of the line.

Hence, the accuracy is given by

$$a \propto nr^{k-1}.$$

So, for example, doubling the radius of the zones increases by a factor of 32, the *linear* accuracy with which a six-dimensional feature like a stopped oriented three-dimensional edge is represented. The intuitive idea that larger zones lead to sloppier representations is entirely wrong because distributed representations hold information much more efficiently than local ones. Even though each active unit is less specific in its meaning, the combination of active units is far more specific. Notice also that with coarse coding the accuracy is proportional to the number of units, which is much better than being proportional to the k th root of the number.

Units that respond to complex features in retinotopic maps in visual cortex often have fairly large receptive fields. This is often interpreted as the first step on the way to a translation invariant representation. However, it may be that the function of the large fields is not to achieve translation invariance but to pinpoint accurately where the feature is!

Limitations on coarse coding. So far, only the advantages of coarse coding have been mentioned, and its problematic aspects have been ignored. There are a number of limitations that cause the coarse coding strategy to break down when the "receptive fields" become too

large. One obvious limitation occurs when the fields become comparable in size to the whole space. This limitation is generally of little interest because other, more severe, problems arise before the receptive fields become this large.

Coarse coding is only effective when the features that must be represented are relatively sparse. If many feature points are crowded together, each receptive field will contain many features and the activity pattern in the coarse-coded units will not discriminate between many alternative combinations of feature points. (If the units are allowed to have integer activity levels that reflect the number of feature points falling within their fields, a few nearby points can be tolerated, but not many.) Thus there is a resolution/accuracy trade-off. Coarse coding can give high accuracy for the parameters of features provided that features are widely spaced so that high resolution is not also required. As a rough rule of thumb, the diameter of the receptive fields should be of the same order as the spacing between simultaneously present feature points.⁶

The fact that coarse coding only works if the features are sparse should be unsurprising given that its advantage over a local encoding is that it uses the information capacity of the units more efficiently by making each unit active more often. If the features are so dense that the units would be active for about half the time using a local encoding, coarse coding can only make things worse.

A second major limitation on the use of coarse coding stems from the fact that the representation of a feature must be used to affect other representations. There is no point using coarse coding if the features have to be recoded as activity in finely tuned units before they can have the appropriate effects on other representations. If we assume that the effect of a distributed representation is the *sum* of the effects of the individual active units that constitute the representation, there is a strong limitation on the circumstances under which coarse coding can be used effectively. Nearby features will be encoded by similar sets of active units, and so they will inevitably tend to have similar effects. Broadly speaking, coarse coding is only useful if the required effect of a feature is the average of the required effects of its neighbors. At a fine enough scale this is nearly always true for spatial tasks. The scale at which it breaks down determines an upper limit on the size of the receptive fields.

⁶ It is interesting that many of the geometric visual illusions illustrate interactions between features at a distance much greater than the uncertainty in the subjects' knowledge of the position of a feature. This is just what would be expected if coarse coding is being used to represent complex features accurately.

Another limitation is that whenever coarse-coded representations interact, there is a tendency for the coarseness to increase. To counteract this tendency, it is probably necessary to have lateral inhibition operating within each representation. This issue requires further research.

Extension to noncontinuous spaces. The principle underlying coarse coding can be generalized to noncontinuous spaces by thinking of a set of items as the equivalent of a receptive field. A local representation uses one unit for each possible item. A distributed representation uses a unit for a set of items, and it implicitly encodes a particular item as the intersection of the sets that correspond to the active units.

In the domain of spatial features there is generally a very strong regularity: Sets of features with similar parameter values need to have similar effects on other representations. Coarse coding is efficient because it allows this regularity to be expressed in the connection strengths. In other domains, the regularities are different, but the efficiency arguments are the same: It is better to devote a unit to a set of items than to a single item, provided that the set is chosen in such a way that membership in the set implies something about membership in other sets. This implication can then be captured as a connection strength. Ideally, a set should be chosen so that membership of this set has strong implications for memberships of other sets that are also encoded by individual units.

We illustrate these points with a very simple example. Consider a microlanguage consisting of the three-letter words of English made up of *w* or *l*, followed by *i* or *e*, followed by *g* or *r*. The strings *wig* and *leg* are words, but *weg*, *lig*, and all strings ending in *r* are not. Suppose we wanted to use a distributed representation scheme as a basis for representing the words, and we wanted to be able to use the distributed pattern as a basis for deciding whether the string is a word or a nonword. For simplicity we will have a single "decision" unit. The problem is to find connections from the units representing the word to the decision unit such that it fires whenever a word is present but does not fire when no word is present.⁷

⁷ Note that the problem remains the same if the decision unit is replaced by a set of units and the task of the network is to produce a different pattern for the word and nonword decisions. For when we examine each unit, it either takes the same or a different value in the two patterns; in the cases where the value is the same, there is no problem, but neither do such units differentiate the two patterns. When the values are different, the unit behaves just like the single decision unit discussed in the text.

Figure 4 shows three representation schemes: a distributed scheme that does not work, a distributed scheme that does work, and a local scheme. In the first scheme, each letter/position combination is represented by a different unit. Since there are only five letter/position possibilities, only five units have connections to the output unit. Each word and nonword produces a different and unique pattern over these five units, but the connections from the five units to the decision unit cannot be set in such a way as to make the decision unit fire whenever one of the words is present and fail to fire whenever one of the nonwords is present.

The reason for the problem is simply that the connections between the letter/position units and the decision units can only capture the degree to which each letter indicates whether the string is a word or not. The *g* tends to indicate that a word is present, whereas the *r* indicates that the item is not a word; but each of the other letters, taken individually, has absolutely no predictive ability in this case.

Whether a letter string is a word or not cannot be determined conclusively from the individual letters it contains; it is necessary to consider also what combinations of letters it contains. Thus, we need a representation that captures what combinations of letters are present in a way that is sufficient for the purposes of the network. One could capture this by using local representations and assigning one node to each word, as in the third panel of Figure 4. However, it is important to see that one need not go all the way to local representations to solve the

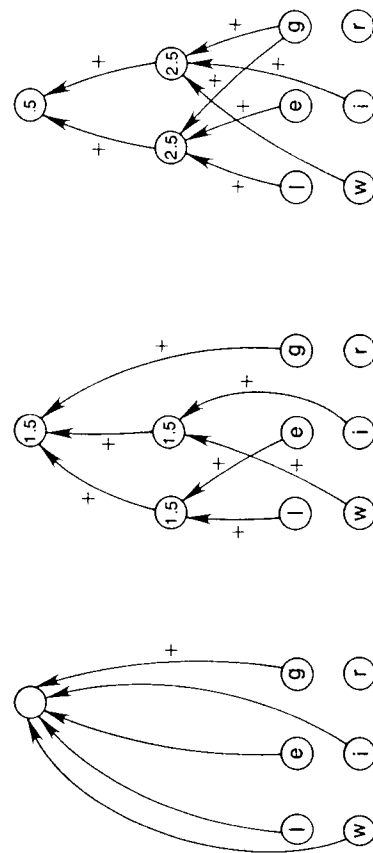


FIGURE 4. Three networks applied to the problem of determining which of the strings that can be made from *w* or *l*, followed by *i* or *e*, followed by *g* or *r* form words. Numbers on the connections represent connection strengths; numbers on the units represent the units' thresholds. A unit will take on an activation equal to 1 if its input exceeds its threshold; otherwise, its activation is 0.