Distributed representations Article No 59

This is a final draft. Please cite from the published version, details at http://www.cognitivescience.net

ENCYCLOPEDIA OF COGNITIVE SCIENCE 2002 ©Macmillan Reference Ltd

Distributed representations

Memory, local representations, distributed representations, coarse codes, conjunctive codes, reduced representations, chunking

Tony Plate

Black Mesa Capital, Santa Fe, New Mexico USA

Definition: Distributed representations are a way of representing information in a pattern of activation over a set of neurons, in which each concept is represented by activation over multiple neurons, and each neuron participates in the representation of multiple concepts.

Contents

The importance of representation

The way information is represented has a large impact on the type of operations that are easy or practical to perform with it. Researchers working on neural models of cognitive tasks have taken representational issues especially seriously. It is a great challenge to work out how to effectively utilize the potentially vast computational power of the human brain, in the face of the unreliability and slowness of individual neurons. It does seem clear that any neural computational scheme that performs at near-human levels must make use of neural representations that make it possible to perform relatively high-level tasks in just a few "steps" of neural computation. There is simply not enough time for many steps of computation in the time that people take to act when having a conversation, playing sports, etc.

Properties of distributed representations

In distributed representations concepts are represented by patterns of activity over a collection of neurons. This contrasts with local representations, in which each neuron represents a single concept, and each concept is represented by a single neuron. Researchers generally accept that a neural representation with the following two properties is a distributed representation (e.g., Hinton et al, 1986):

- Each concept (e.g., an entity, token, or value) is represented by more than more neuron (i.e., by a pattern of neural activity in which more than one neuron is active.)
- Each neuron participates in the representation of more than one concept.

Another equivalent property is that in a distributed representation one cannot interpret the meaning of activity on a single neuron in isolation: the meaning of activity on any particular neuron is dependent on the activity in other neurons (Thorpe 1995).

The distinction between local and distributed representations is not always as clear as it might initially seem (see van Gelder, 1991, for discussion). Is a standard 8-bit binary encoding for numbers between 0 and 255 a local or a distributed code? At the level of numbers each "concept" (number) is represented by multiple "neurons" (bits), and each neuron participates in representing many concepts. However, this encoding can also be viewed as a local representation of powers of 2: 1, 2, 4, 8, etc. This is an example of where a representation is distributed at one level of interpretation but in which individual neurons represent finer-grained features or "micro-features" in a localist fashion.

Similarity and generalization

The similarity of two patterns is very important in distributed representations.

patterns. One sees the same principle at work when a network develops distributed representations for itself, e.g., as in Hinton's (1986) family-tree-learning network. Note that domain similarity can depend upon the task to be performed: two inputs that can be treated as the same for one task may need to be treated as different for another task. At the most basic level, learning is about determining which similarities and differences between inputs are and are not important for a particular task. A network that is either provided with, or that can learn, a good set of features to represent its input will often be able to generalize well from limited data.

Superposition, multiple concepts, interference & ghosting

Using distributed representations, multiple concepts can be represented at the same time on the same set of neurons by superimposing their patterns together. Mathematically, superposition means some sort of addition, possibly followed by a thresholding or a normalizing operation.

How can we tell whether a particular pattern $\mathbf{x}$ is part of a superposition of patterns, denoted by $\mathbf{y}$? The simplest, and most commonly used way is to check whether the similarity between $\mathbf{x}$ and $\mathbf{y}$, exceeds some predetermined threshold. Another technique is discussed later in this article.

As more patterns are superimposed together, it can become difficult to tell whether or not a particular pattern is part of the superposition. When patterns appear in the result of a superposition, but were in fact not part of the superposition, this is known as interference or ghosting. The number of patterns that can be superimposed before ghosting becomes a problem depends on several aspects of the representation: the number of neurons (more neurons means less ghosting); the number of distinct patterns (more patterns means more ghosting); the degree of noise tolerance we wish the representation to have (higher noise tolerance means more ghosting); and the density of patterns.

Density and sparse representations

The total level of activity in a pattern is referred to as the sparseness, or alternatively, the density, of the representation. For a binary representation, this is the fraction of neurons that are active. For a continuous representation the Euclidean length of vectors is often used as the density. Patterns chosen to represent concepts in a model are often restricted to have the same density. Density of patterns in a representation can be tuned to optimise the properties of the representation, e.g., minimizing ghosting and maximizing capacity.

Sparse binary distributed representations have the attractive property that they can be superimposed using the binary-OR rule and with little ghosting until the number of superimposed patterns becomes large. Sparse representations are also of interest because neurophysiological evidence suggests that representations used in the brain are quite sparse [CROSS REFERENCE?].

Advantages of distributed representations

Distributed representations are often held to have many advantages compared to symbolic (e.g., Lisp data structures) and local representations. The most common and important are as follows:

1. Representational efficiency: distributed representations form a more efficient code than localist representations, provided that only a few concepts are to be represented at once. A localist representation using n neurons can represent just n different entities. A distributed representation using n binary neurons can represent up to 2^n different entities (using all possible patterns of zeros and ones).

2. Mapping efficiency: a micro-feature-based distributed representation often allows a simple mapping (that uses few connections or weights) to solve a task. For example, suppose we wish to classify 100 different colored shapes as to whether or not they are yellow. Using a localist representation, this would require a connection from each neuron representing a yellow shape to the output neuron. Using a feature-based distributed representation, all that is required is a single connection from the neuron encoding "yellow" to the output neuron. In general, a mapping that can be encoded in relatively few weights operating on a feature-based distributed representation can be learned from a relatively small training sample and will generalize correctly to examples not in the training set.

3. Continuity (in the mathematical sense): representing concepts in continuous vector spaces allows powerful gradient-based learning techniques such as backpropagation to be applied to many problems, including ones that might otherwise be seen as discrete symbolic problems.

4. Soft capacity limits and graceful degradation: distributed representations typically have soft limits on how many concepts can be represented simultaneously before ghosting or interference becomes a serious problem. Also, the performance of neural networks using distributed representations tends to degrade gracefully in response to damage to the network or noise added to activations. Many researchers find these properties compellingly similar to performance observed in people.

On the other hand, local representations are far simpler to understand, implement, interpret, and work with. If distributed representations do not provide significant advantages for a particular application, it may be more appropriate to use a local representation. Page (2000) argues forcefully that this is the case in many cognitive modelling applications.

Major areas of research in distributed representations

The major areas of research in distributed representations are (a) techniques for representing data more complex than simple tokens, e.g., data with compositional structure, continuous data, probability distributions; (b) properties of representational schemes, e.g., capacity, scaling, reconstruction accuracy; and (c) techniques for learning distributed representations.

Coding problems and techniques

The binding problem

The problem of keeping track of which features or components belong to which objects is know as the "binding problem". Consider trying to represent the presence of several coloured shapes. A simple local representation could have one set of features for colour and another set of units for shape. A red circle would be represented by activity on the red unit and on the circle unit (Figure 1a). However, when we try to represent two coloured objects simultaneously, we encounter a binding problem: does red+blue+circle+triangle mean red circle and blue triangle or blue circle and red triangle (Figure 1b)? This is illustrated in Figure 1 (a) and (b). The binding problem can arise with local or distributed representations when features or components are represented independently and can belong to different objects.

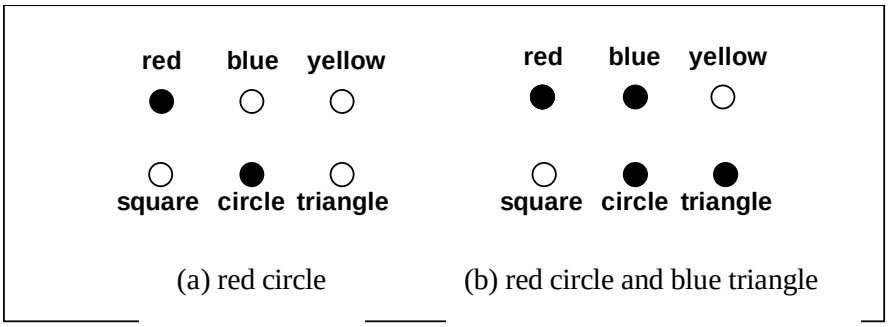

Figure 1. The binding problem: representations of multiple objects on independent feature sets can lose information about which features belong to which objects.

Conjunctive coding

Conjunctive codes are a general approach to solving the binding problem in neural representations. A simple local conjunctive code has one neuron for every possible combination (conjunction) of a single value from each feature set. For example, to represent coloured shapes with 10 possible colours and 5 possible shapes we would use 50 units. Figure 2 shows an example of this with just three colours and three shapes. This technique can also be used when the value on each feature dimension is encoded with a distributed representation: form the outer product of the patterns for each feature dimension. This outer-product operation underlies such well known associative memory schemes as the Willshaw net (Willshaw, 1989) and the Hopfield net (Hopfield, 1982).

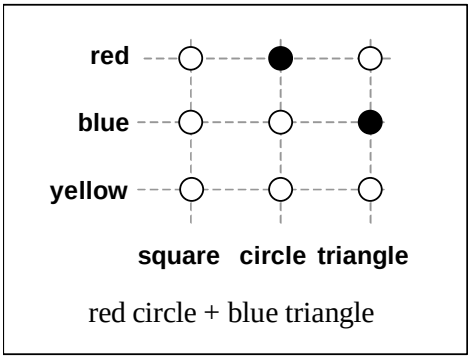

Figure 2. A conjunctive code representing two objects simultaneously.

This simple kind of conjunctive code is an example of how a neural code for a composite object can be computed in a systematic manner from the codes for its constituent parts or features. This is an important aspect of Hinton's influential idea of reduced representations, discussed later in this article.

There are two serious problems that occur with simple outer-product conjunctive codes. The first is inefficiency: resource requirements grow exponentially with the number of feature classes. With *k* feature classes (e.g., colour, shape, size, etc.) each having *n* possible values, the total number of neurons required for a complete conjunctive code is n^k. The informational efficiency of such a code is very poor if only a handful of objects are to be represented simultaneously. Another way of looking at this is that the same set of neurons could represent far more information about each object, or about more objects, if a more efficient code were used. The second problem is that the conjunctive code can hide what makes objects similar, which places high resource demands on mapping and can make learning difficult. If the independent features of the input are useful for solving the problem or constructing the mapping, learning and mapping are simpler when input to a network represents each feature dimension independently rather than in a conjunctive manner..

Coarse coding

One of the apparent paradoxes of neural codes is that a stimulus or entity can be represented more accurately by a collection of neurons with broad or coarse response functions than by a collection of neurons with more finely-tuned response functions. This applies to the representation of both discrete and continuous stimuli or entities.

For representing continuous data, e.g., a position in space (with 2 or more dimensions), this means that one can increase the overall accuracy of coding scheme by decreasing the accuracy with which individual neurons represent a data point (i.e., by making the neurons code the data more coarsely). Consider a simple representation for a point in space, where each of *n* neurons has a randomly chosen centre (in input space), and responds to a data-point within a radius *r* of its centre (its *receptive field*). A two dimensional version of this is shown in Figure 3. Hinton, McClelland and Rumelhart (1986) show that for neurons representing regions in a *k*–dimensional space the inaccuracy of a distributed representation like this is proportional to $1/r^{k-1}$. For example, in a 6-dimensional space, doubling the radius of

each receptive field reduces the inaccuracy of the representation by a factor of 32 on each dimension.

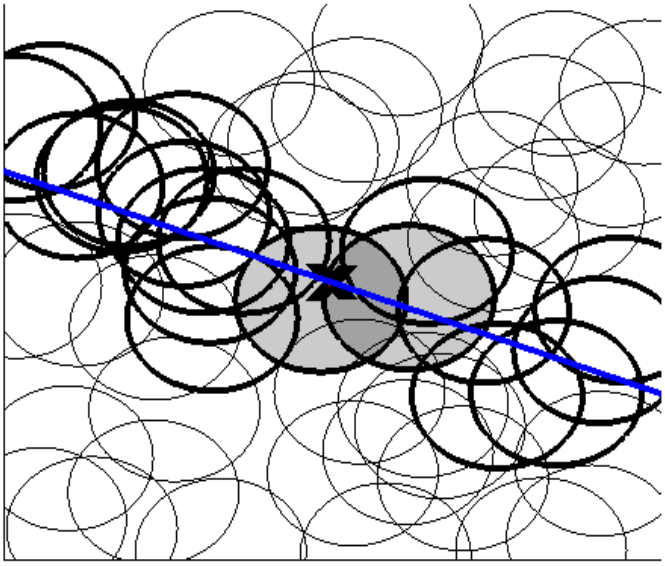

Figure 3. Coarse coding of 2-d positions. Each neuron is represented by a circle showing its receptive field. Neurons that respond to the point X have dark grey receptive fields. Neurons that respond to some point along the solid line have bold

have a compositional nature, and quite often, the composition is hierarchical. In the sentence "Joan watched Sam cook the eggs.", there are a number of different concepts involved: "Joan", "watching", "Jack", "cooking", "eggs". Representing the overall concept communicated by the above sentence involves representing its components and their relationships. If the relationships are not represented we are not able to distinguish between "Joan watched Sam cook the eggs" and "Sam watched Joan cook the eggs". This is another example of a binding problem; we need to bind the concepts "Sam" and "eggs" to the relational concept "cooking". One possibility is to allocate a set of neurons for each role or aspect of the relation: a relation name set containing the code for "cooking", a subject set containing the code for "Sam", and an object set containing the code for "eggs". However, there is also a hierarchical, or recursive, aspect to sentence "Joan watched Sam cook the eggs." Once we have constructed an appropriate representation for "Sam cooking eggs" we then need to bind it and "Joan" to the relational concept "watching." An approach utilizing a fixed-size set of neurons for each role prevents any neat solution to representing the recursive aspects of the whole problem because the number of neurons in the representation of a relation is larger than the number of neurons allocated to represent the filler of a role.

The compositional and recursive nature of concepts is not limited to obviously linguistic tasks, but is widespread in human cognition; visual scene understanding and analogy recall and matching are two examples. Although representing hierarchical compositional structures in the activations and/or weights of a neural network is not easy, it is important as it could eventually allow the application of powerful neural-network learning techniques to difficult problems such as language understanding and acquisition.

Reduced descriptions

Hinton (1990) introduced the idea of "reduced description" as way of representing hierarchical compositional structure in fixed-size distributed representations. The basic idea is that a relation can be represented in two different ways: (a) as an expanded representation in which the fillers of each role are represented on separate groups of neurons (e.g., a relation with two arguments and a relation name could use three groups of n neurons); and (b) as a compressed or reduced description over just n neurons. Since the reduced description for a relation occupies just n neurons, it can be used as a filler in some other relation, which in turn could have a reduced description, and so on, allowing arbitrary levels of nesting. A reduced representation behaves like a pointer in symbolic data structures in that it gives a way of referring to another structure. An important difference is that unlike a pointer, a reduced description should carry some information about its contents that can be accessed without expanding the reduced description into a full description. This allows processing to be sensitive to components nested within reduced descriptions without having to unpack multiple levels of nested relations.

In the 1990's researchers developed a number of concrete neural schemes for implementing reduced descriptions. Many use some form of conjunctive role-filler bindings in which both the roles and fillers of a relation are represented with distributed patterns. Many of these schemes differ mainly in the binding operation they use; it turns out that there are many alternatives to a straight outer-product for forming a conjunctive code of two patterns.

RAAMs

Pollack (1990) used a bottleneck auto-encoder network to learn reduced descriptions for relations, which he called "Recursive Auto-Associative Memory" (RAAM). A full, expanded relation was represented across the input units of the network: each filler on one group of input units, and possibly the relation name on a final group of input units. The hidden layer was the same size as one of the groups of input units and was intended to contain a reduced description of the full relation. The network learnt, using backpropagation, to compress the full relation down to a reduced description, which could then be expanded back out to a full relation. During learning the network had to discover simultaneously how to create and how to decode reduced descriptions for relations. Pollack showed that a network like this could reliably learn to represent hierarchically structured relations that were several levels deep.

Tensor Product Representations

Smolensky (1990) proposed a tensor-product formalism for representing recursive structure. This extends the outer-product role-filler binding operation to higher dimensions. For example, suppose a role and a filler are each represented by a pattern over a line of n neurons. Then their outer-product binding is a pattern over a square of n^2 neurons. This can be bound with another role vector (again a pattern over n neurons) by forming the tensor product, which in this situation is a pattern over a cube of n^3 neurons. This can be taken to arbitrarily deep levels of nesting. However, the number of neurons required increases exponentially with the depth of conceptual nesting.

Holographic Reduced Representations

Holographic Reduced Representations (HRRs, Plate 1995) are a role-filler binding scheme for recursive compositional structure based on conjunctive coding implemented using circular convolution [cross ref to article on CONVOLUTION BASED MEMORY MODELS]. In HRRs, roles, fillers, labels, and entire relations are all represented as patterns over n neurons. Circular convolution is defined as $\mathbf{z} = \mathbf{x} \otimes \mathbf{y}$, where $z_i = \sum_{k=0}^{n-1} x_k y_{(i-k) \bmod n}$ (where n is the length of the vectors). Circular convolution is a conjunctive code that keeps dimensionality constant: given role and filler patterns over n neurons each, their circular convolution is also a pattern over n neurons. This makes it simple to build recursive structures. The encoding of a relation is the superposition of role-filler bindings (and possibly a relation label) and thus is also a pattern over n neurons and is easily used as a filler in another relation. For example, a reduced representation for the sentence "Joan watched Sam cooking eggs" can be constructed as follows:

$$\mathbf{S}_1 = \mathbf{cook} + \mathbf{cook}_{\text{agt}} \otimes \mathbf{Sam} + \mathbf{cook}_{\text{obj}} \otimes \mathbf{eggs}$$

$$\mathbf{S}_2 = \mathbf{watch} + \mathbf{watch}_{\text{agt}} \otimes \mathbf{Joan} + \mathbf{watch}_{\text{obj}} \otimes \mathbf{S}_1$$

where all the variables are patterns over n neurons (**cook** and **watch** are relation labels; **Sam**, **eggs**, etc. are patterns representing entities; and $\mathbf{cook}_{\text{agt}}$, etc. are patterns representing the roles of the relations.)

A filler of a role in a relation in a reduced representation may be decoded by convolving with the approximate inverse of the role pattern. The approximate inverse of **x**, denoted by $\mathbf{x}^{\mathrm{T}}$, is a simple permutation of elements: $x_i^{\mathrm{T}} = x_{-i \bmod n}$, and has the property that $\mathbf{x}^{\mathrm{T}} \otimes (\mathbf{x} \otimes \mathbf{y}) \approx \mathbf{y}$

For example, to recover the filler of the cook-agent role in $\mathbf{S}_1$, we compute $\mathbf{S_1} \otimes \mathbf{cook}_{\mathrm{agt}}^{\mathrm{T}}$, which results in the vector **Sam+noise**, because

$$\begin{aligned}\mathbf{cook}_{\mathrm{agt}}^{\mathrm{T}} \otimes \mathbf{S}_1 &= \mathbf{cook}_{\mathrm{agt}}^{\mathrm{T}} \otimes (\mathbf{cook} + \mathbf{cook}_{\mathrm{agt}} \otimes \mathbf{Sam} + \mathbf{cook}_{\mathrm{obj}} \otimes \mathbf{eggs}) \\ &= \mathbf{cook}_{\mathrm{agt}}^{\mathrm{T}} \otimes \mathbf{cook} + \mathbf{cook}_{\mathrm{agt}}^{\mathrm{T}} \otimes \mathbf{cook}_{\mathrm{agt}} \otimes \mathbf{Sam} + \mathbf{cook}_{\mathrm{agt}}^{\mathrm{T}} \otimes \mathbf{cook}_{\mathrm{obj}} \otimes \mathbf{eggs} \\ &= \mathbf{noise} + \mathbf{Sam} + \mathbf{noise}\end{aligned}$$

In order to recognize **Sam+noise** as the pattern **Sam**, we must pass it through an auto-associative clean-up memory that can take **Sam+noise** as input and return the pattern **Sam**. All potential decoding targets, such as lower-level patterns such as **cook**, $\mathbf{cook}_{\mathrm{agt}}$, **Sam**, and higher-level patterns representing chunks, such as $\mathbf{S}_1$ and $\mathbf{S}_2$, must be stored in this long-term clean-up memory. A clean-up memory is also necessary to identify when a decoding target is not present. Without a clean-up memory it would be impossible to tell whether or not there was anything bound to $\mathbf{watch}_{\mathrm{agt}}$ in $\mathbf{S_1}$ because $\mathbf{S_1} \otimes \mathbf{watch}_{\mathrm{agt}}^{\mathrm{T}}$ is a pattern with similar statistical properties to any other pattern. With a clean-up memory containing all potential decoding targets, $\mathbf{S_1} \otimes \mathbf{watch}_{\mathrm{agt}}^{\mathrm{T}}$ can be identified as noise because it almost certainly will not be similar to anything in the clean-up memory.

Binary Spatter Codes

Kanerva's (1996) "Binary Spatter Code" is a scheme for encoding complex compositional structures in binary distributed representations. Binary Spatter Codes use binary vectors as patterns, element-wise exclusive-or for encoding and decoding bindings, and a thresholded sum for superposition. They have similar properties to HRRs.

Holistic Processing

A major reason for interest in distributed representations of complex structure is the potential for performing structure-sensitive processing without having to unpack hierarchical structures.

Determining similarity is one of the simplest types of processing. Plate (2000) shows that the dot-product of HRRs reflects both superficial similarity (similarity of components) and structural similarity (similarity of structural arrangement of components). The dot-product of HRRs composed of similar entities is higher if the entities are arranged in an analogical (isomorphic) structure. This means that HRRs can be used for fast (but approximate) detection of structural similarity. HRR computations can also be used to rapidly but imperfectly identify corresponding entities in isomorphisms (Plate 2000, Eliasmith and Thagard 2001).

Various authors have demonstrated that a variety of structure-sensitive manipulations can be performed on distributed representations without unpacking them. Pollack (1990) trained a feedforward network to transform reduced descriptions for propositions like (LOVED X Y) to ones for (LOVED Y X) where the reduced descriptions where found by a RAAM. Chalmers (1990) trained a feedforward network to transform reduced descriptions of simple passive sentences to reduced descriptions of active sentences, where the reduced descriptions were found by a RAAM. Niklasson and van Gelder (1994) trained a feedforward network to do material conditional inference, and its reverse, on reduced descriptions found by a RAAM. Legendre, Miyata and Smolensky (1991) showed how tensor product representations for active sentences could be transformed to ones for passive sentences (and vice-versa) by a pre-calculated linear transform. Neumann (2001) trained networks to perform holistic transformations on a variety of representations, including RAAMs, HRRs, and Binary Spatter Codes.

Interpretation of distributed representations

It is straightforward to tell what is represented in a model that uses symbolic or local neural representations. This is not the case with models that use distributed representations. It is important to note that for the purposes of processing the information present in a distributed representation, it is usually NOT necessary to identify what is represented in terms easily understandable to people. Indeed, the whole point of having a distributed representation is that it makes further processing simpler than performing the same computations on a more easily interpretable representation. Interpretation of a distributed representation is typically only necessary either when outputs need to be computed in a readily interpretable form or when a person wants to gain insight into the internal workings of a model.

There are two quite different senses in which a distributed representation can be interpreted: (a) determine what items are represented in a pattern of activation, where patterns for individual items are known; and (b) determine what features a network has learned to use to represent a set of items.

Algorithms for determining what items are represented

A simple and common algorithm for determining the items present in a distributed pattern of activation is to compute the dot-product of each item with the pattern of activation. A feed-forward neural networks with a localist output representation performs this computation in its final layer of weights. Often it is useful to apply a thresholding or a winner-take-all function to the outputs to cut down the noise.

It is also possible to take an inferential approach to identifying the items present in a distributed code. The idea is to find the best explanation for the observed pattern of activities, in terms of the items that could be present. Zemel, Dayan and Pouget (1998) do this in analysing the potential of sparse distributed representations for representing probability distributions. This requires three pieces of knowledge, which can be combined using Baye's rule: (1) the set of all possible probability distributions that could be represented, and the respective prior probability of each probability distribution (prior to knowledge of current activities); (2) the currently observed pattern of activity; and (3) for each possible probability distribution, the probability

that it would generate the currently observed pattern of activity. In Zemel et al's approach, the probability distribution represented by a pattern of activity is the one with the maximum a posteriori (MAP) probability (the posterior probability of a distribution is the product of its prior probability and the probability that it would have generated the currently observed pattern of activity.)

Understanding learned representations

Principal Components Analysis is often used to gain understanding into a learned distributed representation. Elman (1991) trained a network to predict the next word in sentences generated from a simple recursive English-like language. The predictions made by the network indicated that it had managed to learn something about the recursive nature of the language. Elman used Principal Components Analysis to gain some understanding of how the network represented state (i.e., its memory of what it had seen so far); he showed that a particular phrase caused a similar shape of trajectory through the principal component space independent of its level of embedding, and that level of embedding determined the position of the trajectory.

Conclusion

Distributed representations offer powerful representational principles that can be used in neural network approaches to learning and to modelling human cognitive performance. Research continues on three main aspects of distributed representation: schemes for representing data; properties of representational schemes; and ways of learning distributed representations. The ability to learn and use distributed representations of concepts, and to compose complex data structures out of these representations offers the potential of building general-purpose computers that combine the power of symbolic manipulation with the robustness, learning and generalization ability of neural networks.

References

Chalmers DJ (1990) Syntactic transformations on distributed representations. *Connection Science*. 2(1-2): 53-62.

Eliasmith E and Thagard P (2001) Integrating structure and meaning: a distributed model of analogical mapping. *Cognitive Science* 25:245-286.

van Gelder T (1991) What is the 'D' in 'PDP'? An Overview of the Concept of Distribution. In *Philosophy and Connectionist Theory,* Stich S, Rumelhart D and Ramsey W (eds). Lawrence Erlbaum, Hillsdale NJ.

Hinton GE (1990) Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, 46:47-75.

Hinton GE (1986) Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science*, 1-12 Hillsdale, NJ: Erlbaum. (Reprinted in *Parallel Distributed Processing: Implications for Psychology and Neurobiology*, Ed. Morris RGM (1989), pp46-61, Oxford University Press, Oxford, UK.)

Hinton GE, McClelland JL and Rumelhart DE (1986) Distributed Representations. In *Parallel distributed processing: Explorations in the Microstructure of Cognition*, Volume 1, eds, Rumelhart DE and McClelland JL and the PDP research group, pp77-109, MIT Press, Cambridge, MA.

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

Kanerva P (1996) Binary spatter-coding of ordered k-tuples. In *Artificial Neural Networks–ICANN Proceedings,* Volume 1112 of Lecture Notes in Computer Science, von der Malsburg C, von Seelen W, Vorbruggen J, and Sendhoff B (Eds.), pp869–873. Springer, Berlin.

Legendre G, Miyata Y and Smolensky P (1991) Distributed recursive structure processing. In Touretzky DS and Lippman, R (Eds) *Advances in Neural Information Processing Systems 3*, Morgan Kaufmann, San Mateo, CA, pp 591-597.

Niklasson LF and van Gelder T (1994) Can Connectionist Models Exhibit Non-Classical Structure Sensitivity? *Proceedings of the Sixteenth Annual Conference of The Cognitive Science Society*. Erlbaum, Hillsdale, NJ, pp 664-669.

Page (2000) Connectionist modelling in psychology: A localist manifesto. *Behavioral and Brain Sciences*, 23:443-512

Plate TA (2000) Structured operations with vector representations. *Expert Systems: The International Journal of Knowledge Engineering and Neural Networks: Special Issue on Connectionist Symbol Processing*, 17(1):29–40.

Plate TA (1995) Holographic reduced representations. *IEEE Transactions on Neural Networks* 6(3):623–641.

Pollack JB (1990) Recursive distributed representations. *Artificial Intelligence*. 46(1-2): 77-105.

Smolensky P (1990) Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence,* 46(1-2):159–216.

Thorpe S (1995) Localized versus distributed representations. In *The Handbook of brain theory and neural networks*, ed Arbib MA. MIT Press, Cambridge MA.

Willshaw D (1989) Holography, associative memory, and inductive generalization. In Hinton GE and Anderson JA (Eds.), *Parallel models of associative memory*. (Updated Edition) pp 99-127, Erlbaum, Hillsdale, NJ.

Zemel R, Dayan P and Pouget A (1998) Probabilistic Interpretation of Population Codes. *Neural Computation*, 10(2): 403-430.

Further reading

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

Baum EB, Moody J and Wilczek F (1988) Internal Representations for Associative Memory. *Biological Cybernetics*, 59:217-228.

Bourlard H and Kamp Y (1988) Auto-association by multilayer perceptrons and singular value decomposition, *Biological Cybernetics* 59:291-294.

Deerwester S, Dumais ST, Furnas GW, Landauer TK, and Harshman R (1990) Indexing By Latent Semantic Analysis. *Journal of the American Society For Information Science*, 41:391-407.

Elman JL (1991) Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning* 7:194-220.

Halford G, Wilson WH, and Phillips S (1998) Processing capacity defined by relational complexity: Implications for comparative, developmental, and cognitive psychology. *Behavioral and Brain Sciences* 21(6), 803–831.

Hinton GE, Dayan P, Frey BJ and Neal R (1995) The wake-sleep algorithm for unsupervised Neural Networks. *Science*, 268:1158-1161.

Hinton GE and Ghahramani Z (1997) Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society of London*, B, 352:1177-1190.

Hummel JE and Holyoak KJ (1997) Distributed Representations of Structure: A theory of Analogical Access and Mapping. *Psychological Review* 104(3):427-466.

LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, and Jackel LD (1989) Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541-551.

Neumann J (2001) Holistic Processing of Hierarchical Structures in Connectionist Networks. PhD thesis, University of Edinburgh, http://www.cogsci.ed.ac.uk/~jne/holistic_trafo/thesis.pdf.

Olshausen BA and Field DJ (1996) Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607-609.

Rachkovskij DA (2001) Representation and processing of structures with binary sparse distributed codes. *IEEE Transactions on Knowledge and Data Engineering*, 13(2):261-276.

Rumelhart DE and McClelland JL (1986) On Learning the Past Tenses of English Verbs. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Volume 2, McClelland JL, Rumelhart DE and the PDP research group (eds) pp216-271, MIT Press, Cambridge, MA.

Zemel RS and Hinton GE (1995) Learning Population Codes by Minimizing Description Length. *Neural Computation*, 7(3): 549-564.

Glossary

Neural activation#The state of activity of a neuron, often summarized as the rate and/or phase of firing of the neuron.

Pattern, activation pattern#A pattern of activation across a set of neurons, often represented as a vector of binary or real numbers

Trace, memory trace#The neural activations that constitute a particular memory

Vector#A mathematical term for a list of numbers

Binding#A record of an association between two or more concepts

Receptive field#The set of stimuli to which a neuron responds