Convolution-based memory models

Memory, convolution, distributed representation, holography, chunking

Tony A Plate tplate@acm.org

Black Mesa Capital, Santa Fe, New Mexico USA

Definition: Convolution-Based Memory Models are a mathematical model of neural storage of complex data structures using distributed representations. Data structures stored range from lists of pairs through sequences, trees and networks.

Contents

and CBMMs: information about each element of an item or region of an image is distributed across the entire storage medium.

Rapidly binding components of a memory together

CBMMs use two operations for composing patterns: superposition and binding. For patterns of real numbers, superposition is ordinary element-wise addition; for patterns of binary numbers, superposition is element-wise binary-OR. Superposition is useful for forming unstructured collections of items. However, associations or *bindings* between items cannot be represented using superposition alone because of the *binding problem*. [CROSS REF TO ARTICLE ON "BINDING PROBLEM", and section on "BINDING PROBLEM" in article on "DISTRIBUTED REPRESENTATIONS".]

CBMMs use *convolution* as a binding operation: convolution binds two patterns together into one. If **x** and **y** are *n*-dimensional pattern vectors (subscripted 0 to *n*-1), then the circular convolution of **x** and **y**, written **z** = **x**⊗**y**, is also an *n*-dimensional pattern vector and has elements

$$z_i = \sum_{k=0}^{n-1} x_k\, y_{(i-k)\bmod n} \ .$$

Circular convolution can be viewed as a compression of the outer (or tensor) product of the two vectors, where compression is achieved by summing particular elements, as shown in Figure 1. (Other variants of convolution can be viewed as slightly different ways of compressing the outer product.)
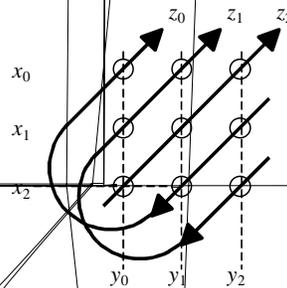
One the most important properties of convolution is *similarity preservation*: if patterns **red** and **pink** are similar, then the bindings **red⊗square** and **pink⊗square** will also be similar, to approximately the same degree.

Rapid retrieval and interference effects

Convolution bindings can be easily decoded using inverse convolution operations. For example, using exact inverses, **red**$^{-1}$**⊗red⊗circle = circle**. However, the exact inverse can be numerically unstable and is not always the best choice for decoding. For many vectors, e.g., those whose elements have independent Gaussian statistics with mean zero and variance 1/n, an approximate inverse can be used. The approximate inverse of **x** is denoted by **x**^{T} (this notation is chosen because it is closely related to the matrix transpose.) It is a simple rearrangement of the elements of **x**: $x_i^T = x_{(-i) \bmod n}$. Reconstruction using the approximation inverse is noisy, i.e., **red**^{T}**⊗red⊗circle** is only approximately equal to **circle**, but is usually more stable in the presence of noise than reconstruction using the exact inverse. If necessary, exact reconstructions can be provided passing the noisy result through a cleanup memory, which returns the closest matching pattern among the patterns it contains.

Decoding still works when multiple associations are superimposed. For example,
$$\mathbf{blue}^T \otimes ((\mathbf{red} \otimes \mathbf{circle}) + (\mathbf{blue} \otimes \mathbf{square}))$$
$$= (\mathbf{blue}^T \otimes \mathbf{red} \otimes \mathbf{circle}) + (\mathbf{blue}^T \otimes \mathbf{blue} \otimes \mathbf{square})$$
$$\approx \mathbf{square}.$$

Because of the randomising properties of convolution, the first term on the right in the expansion (**blue**^{T} **⊗ red ⊗ circle**) is not similar to any of **blue**, **red**, **circle**, or **square** and can be regarded as noise,. The second term on the right (**blue**^{T} **⊗ blue⊗square**) is a noisy version of **square**. The sum of these two terms is an even noisier, but still recognisable, version of **square**. When larger numbers of bindings are superimposed together the interference effects can become significant, though increasing the vector dimension can reduce interference effects. For further discussion and quantitative analysis, see Murdock 1982, Metcalf 1982, or Plate 1994.

TODAM

Murdock's (1982) "Theory of Distributed Associative Memory" model (TODAM) is intended to model patterns of human performance on memorization tasks, focussing on tasks involving lists of paired associates. For example, a subject might be asked to memorize the list "cow-horse, car-truck, dog-cat, and pen-pencil" and then answer such questions as "Did `car' appear in the list?" (recognition), or "What was `cat' associated with?" (cued recall). Subjects' relative abilities to perform these and other tasks under different conditions, and the types of errors they produce, give insight into the properties of human memory. Some of the conditions commonly varied are the number of pairs, the familiarity of items, the similarity of items, and the position of recall or recognition targets within the list.

The TODAM formula for sequentially constructing a memory trace for a list of pairs $(\mathbf{x}_i, \mathbf{y}_i)$ of item patterns is as follows:

$$\mathbf{T}_j = \alpha\, \mathbf{T}_{j-1} + \gamma_1 \mathbf{x}_j + \gamma_2 \mathbf{y}_j + \gamma\ (\mathbf{x}_j * \mathbf{y}_j) \qquad \mathbf{T}_j = \alpha \mathbf{T}_{j-1} + \gamma_1 \mathbf{x}_j + \gamma_2 \mathbf{y}_j + \gamma_3 \mathbf{x}_j \otimes \mathbf{y}_j$$

where $\mathbf{T}_j$ is the memory trace pattern (a vector) representing pairs 1 through j (with $\mathbf{T}_0 = \mathbf{0}$). The scalars α, γ_1, γ_2, and γ_3 are adjustable parameters of the model, taking values between 0 and 1.

TODAM uses an "unwrapped" version of convolution which expands the size of vectors each time it is applied, but TODAM could use any convolution operation.

For example, the memory trace for the list of three pairs: (**a,b**), (**c,d**), and (**e,f**) is built as follows:

$$\mathbf{T}_1 = \gamma_1\mathbf{a} + \gamma_2\mathbf{b} + \gamma_3\mathbf{a} \otimes \mathbf{b}$$
$$\mathbf{T}_2 = \gamma_1\mathbf{c} + \gamma_2\mathbf{d} + \gamma_3\mathbf{c} \otimes \mathbf{d} + \alpha(\gamma_1\mathbf{a} + \gamma_2\mathbf{b} + \gamma_3\mathbf{a} \otimes \mathbf{b})$$
$$\mathbf{T}_3 = \gamma_1\mathbf{e} + \gamma_2\mathbf{f} + \gamma_3\mathbf{e} \otimes \mathbf{f} + \alpha(\gamma_1\mathbf{c} + \gamma_2\mathbf{d} + \gamma_3\mathbf{c} \otimes \mathbf{d}) + \alpha^2(\gamma_1\mathbf{a} + \gamma_2\mathbf{b} + \gamma_3\mathbf{a} \otimes \mathbf{b})$$

Item recognition is done by comparing an item with the trace: item **x** was stored in trace **T** if $\mathbf{x} \cdot \mathbf{T} > t$ (i.e., if the dot product of **x** and **T** is greater than some threshold t.)

Cued recall is accomplished by decoding the trace with the cue: if item **x** was stored in trace **T**, then $\mathbf{x}\#\mathbf{T}$ is a noisy reconstruction of the partner of **x** (where $\mathbf{x}\#\mathbf{T}$ is another way of writing $\mathbf{x}^{\mathrm{T}} \otimes \mathbf{T}$).

Some of the predictions of TODAM that are supported by evidence in the psychological literature are as follows:

o   Performance decreases with increasing list length;

o   Cued recall is symmetric: the recall of **x** given **y** from a trace containing the pair $\mathbf{x} \otimes \mathbf{y}$ is as accurate as the recall of **y** given **x** from the same trace;

o   There is no primacy effect, only a recency effect, because forgetting is geometric in α;

o   Cued recall for a particular item can be superior to recognition for that same item – it can be possible to recall an item that cannot be recognized. This is because weights can be defined so that associative information is stronger than item information.

CHARM

The "Composite Holographic Associative Recall Model" (CHARM) of Metcalfe (Metcalfe-Eich 1982) was specifically intended to address the effects of similarity among items in cued recall from lists of paired associates. CHARM uses an even simpler storage method than TODAM – it stores only associative information and no item information. The memory trace for a list of pairs ($\mathbf{x}_i,\mathbf{y}_i$) of item patterns is constructed as follows:

$$\mathbf{T} = \sum_{i=1}^{k} \mathbf{x}_i \otimes \mathbf{y}_i .$$

CHARM uses a truncated version of the non-wrapped convolution used in TODAM so that the patterns for memory traces are the same size as for items.

As with TODAM, the process for performing cued-recall in CHARM begins by correlating a composite memory trace with the cue, e.g., to find the item corresponding to $\mathbf{x}_1$ in $\mathbf{T}$, $\mathbf{x}_1 \# \mathbf{T}$ is computed. The resulting pattern will be a noisy version of the pattern associated with $\mathbf{x}_1$ in $\mathbf{T}$, which is passed through a cleanup memory. For the purposes of Metcalfe's experiments, the cleanup memory contained patterns for items stored in the memory trace, and patterns for some other items not stored in the memory trace.

One type of retrieval phenomena modelled with CHARM is the reduced ability to accurately recall items from a list whose members are similar, versus from a list whose members are dissimilar. For example, performance on a pair such as NAPOLEON—ARISTOTLE is worse when the pair is embedded in a list of pairs of names of other famous people (a homogenous list) than when it is embedded in a list containing items conceptually unrelated to it, such as RED-BLUE. Furthermore, with homogenous lists, incorrect recall of an item that is similar to the correct response and that was also in the list with an associated item similar to the cue is a frequent type of error in both CHARM and with human subjects.

Binding via full tensor products

A list of paired items is a very simple set of relationships. Many cognitive tasks demand the ability to store more complicated relationships. For example, understanding language requires the ability to work with recursive structures: a phrase can have a verb, a subject and an object, but the object could be a phase itself, which could even contain further subphrases. For example, the sentence "I believe that politicians will say whatever will help them to get elected" contains at least three levels of recursion.

One of the first concrete descriptions of such a scheme was given by Smolensky (1990). Smolensky used tensor products to bind roles and fillers together in a recursive manner. For example, the sentence "Politicians tell stories" could be represented as the rank-2 tensor $\mathbf{T} = \mathbf{politicians} \otimes \mathbf{tell_{agent}} + \mathbf{stories} \otimes \mathbf{tell_{object}}$, where $\mathbf{politicians}$ is a pattern representing politicians, $\mathbf{tell_{agent}}$ is a pattern for the agent role of "tell", etc, and $\otimes$ is the tensor product (a generalization of the outer product). Tensors can be superimposed and decoded in a manner similar to convolution traces; the role pattern $\mathbf{tell_{agent}}$ can be used to decode the tensor $\mathbf{T}$ to retrieve the pattern $\mathbf{politicians}$. What makes the use of tensors interesting is that the rank-2 tensor $\mathbf{T}$ can be used as the filler in some higher-level role-filler binding, e.g., to represent the meaning of the sentence "I know politicians tell stories."

Holographic Reduced Representations

Holographic Reduced Representations (HRRs) (Plate, 2000) use convolution-based role-filler bindings to construct patterns representing a recursive structures.

The HRR for the proposition "Politicians tell stories" is constructed as follows:

$$\mathbf{P}_{tell} = \mathbf{tell} + \mathbf{politicians} + \mathbf{stories} + \mathbf{tell}_{agt} \otimes \mathbf{politicians} + \mathbf{tell}_{obj} \otimes \mathbf{stories}$$

If we have the pattern $\mathbf{P}_{tell}$ and know the role patterns, then we can reconstruct a filler pattern by convolving $\mathbf{P}_{tell}$ with the approximate inverse of a role pattern. For example, $\mathbf{tell}^{\mathbf{T}}_{agt} \otimes \mathbf{P}_{tell}$ gives a noisy version of **politicians** which can be put through a clean-up memory to provide an accurate reconstruction.

The HRR pattern $\mathbf{P}_{tell}$ a *reduced representation* [CROSS REFERENCE TO "REDUCED REPRESENTATIONS" IN ARTICLE ON "DISTRIBUTED REPRESENTATIONS"] for the proposition "Politicians tell stories" and can be used as a filler in a higher-order proposition. For example, the HRR $\mathbf{P}_{know}$, representing "Bill knows politicians tell stories" is constructed as follows:

$$\mathbf{P}_{know} = \mathbf{know} + \mathbf{bill} + \mathbf{P}_{tell} + \mathbf{know}_{agt} \otimes \mathbf{bill} + \mathbf{know}_{obj} \otimes \mathbf{P}_{tell}$$

Such higher-level HRRs can be decoded in the same way as first order HRRs. For example, the filler of the know-object role is decoded as follows:

$$\mathbf{P}_{know} \otimes \mathbf{know}^{T}_{obj} \approx \mathbf{P}_{tell}$$

This reconstructed filler is a proposition. To discover its fillers it could be cleaned up and then decoded again.

HRRs are similar if they merely involve similar entities or predicates. Because of the similarity preserving properties of convolution, they will be even more similar if the entities are involved in similar roles. Thus it turns out that the similarity of HRRs can reflect both superficial and structural similarity in a way that neatly corresponds to the data on human analog retrieval (Plate 2000).

Implementing convolution-based memories in connectionist networks

The various operations used in convolution-based memory models, i.e., convolution, correlation, approximate inverse, dot-product, and cleanup memory, are easily implemented in connectionist networks. Convolution encoding and decoding can be implemented by suitably connected networks of "sigma-pi" neurons. [CROSS REFERENCE TO ARTICLE DISCUSSING "sigma-pi" neurons] Figure 2 shows a network that computes the circular convolution of two 3-element vectors.
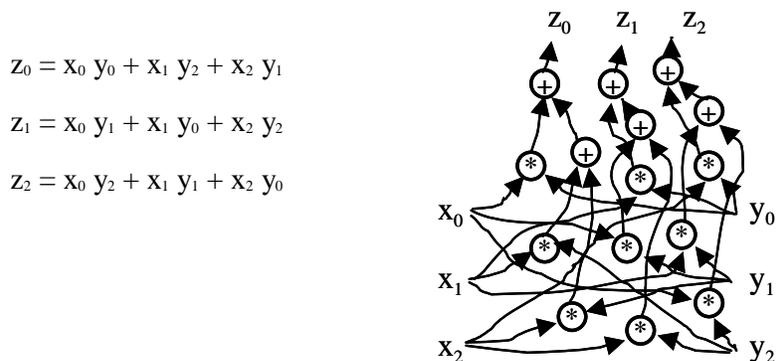
$$z_0 = x_0\ y_0 + x_1\ y_2 + x_2\ y_1$$

$$z_1 = x_0\ y_1 + x_1\ y_0 + x_2\ y_2$$

$$z_2 = x_0\ y_2 + x_1\ y_1 + x_2\ y_0$$

Figure 2: The *circular convolution* z of vectors x and y drawn as a network of three sigma-pi neuron. Each sigma-pi neuron computes the sum of three products as shown on the left.

The pattern of connections in the sigma-pi network that computes circular convolution may seem unrealistically intricate and precise for a biological circuit. However, Plate (2000) shows that sigma-pi networks that sum random products of pairs of elements from x and y can also function as encoding and decoding networks with similar properties to convolution.

For computation of similarity, a dot product can be computed by a single sigma-pi neuron. Clean-up memory can be implemented in a several ways, e.g., Kanerva's (1988) Sparse Distributed Memory, or Baum, Moody and Wilczek's (1988) various associative content addressable memory schemes.

References

Borsellino, A. and T. Poggio (1973). Convolution and correlation algebras. Kybernetik 13, 113–122.

Metcalfe Eich, J. (1982). A composite holographic associative recall model. Psychological Review 89, 627–661.

Murdock, B. B. (1982). A theory for the storage and retrieval of item and associative information. Psychological Review 89 (6), 316–338.

Plate, T. (2000a). Randomly connected sigma-pi neurons can form associator networks. Network: Computation in Neural Systems 11 (4), 321–332.

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

Plate, T. A. (1995). Holographic reduced representations. IEEE Transactions on Neural Networks 6 (3), 623–641.

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

Further reading

Anderson, J. A. (1973). A theory for the recognition of items from short memorized lists. Psychological Review 80 (6), 417–438.

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

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

Kanerva, P. (1996). Binary spatter-coding of ordered k-tuples. In C. von der Malsburg, W. von Seelen, J. Vorbruggen, and B. Sendhoff (Eds.), Artificial Neural Networks–ICANN Proceedings, Volume 1112 of Lecture Notes in Computer Science, Berlin, pp. 869–873. Springer.

Murdock, B (1993) TODAM2: A Model for the Storage and Retrieval of Item, Associative, and Serial-Order Information. Psychological Review, 100(2), 183-203.

Rachkovskij DA and Kussul EM (2001) Binding and Normalization of Binary Sparse Distributed Representations By Context-Dependent Thinning. *Neural Computation* 13(2), 411-452.

Willshaw, D. (1981). Holography, associative memory, and inductive gener-alization. In G. E. Hinton and J. A. Anderson (Eds.), Parallel models of associative memory. Hillsdale, NJ: Erlbaum.

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