An Analysis of Contemporary Associative Memory Models

Anton V. Roupassov-Ruiz

Department of Psychology, University of Alberta

PSYCH 396: Individual Research

Dr. Jeremy B. Caplan

June 12, 2024

Abstract

Human memory is still not very well understood. There exist many models that attempt to approximate how the mind stores and retrieves memories—days, weeks, or even years later. One such kind of memory relates to associations—how the brain links different knowledge representations together (Kahana, 2012). The focus of this paper is to acquaint those who may be new to associative memory—beginning with a gentle introduction to common associative memory tasks and slowly progressing through increasingly complex association representations—all while describing key terminology along the way. Matrix, convolution, local trace, and concatenation models are discussed using supporting examples where helpful. Similarities and differences between how the models represent, store, and retrieve associations are examined. Consider that although the material presented may be difficult at times to understand, asking Dr. Caplan any questions you may have is—from my experience—the most effective way to go about the process.

There are various tasks that are commonly used to test associative memory. Kahana provides a comprehensive overview in his 2012 paper, from which the following is derived. The *cued-recall* task is the classic task used to research associative memory. Participants are shown a list of paired-items (which are often words but can also be letters, phonemes, or graphemes) or *paired associates,* ranging from $A_1 - B_1, A_2 - B_2, \ldots, A_L - B_L$ with L representing the number of pairs in the list.

In the *study-test method* of cued-recall, participants are first presented with a list of paired associates (called a *study period*). They are then presented with a *probe* and asked to recall the probe's *pair associate*—the *target* item. Either *forward recall* (recall B_i when presented with A_i) or *backward recall* (recall A_i when presented with B_i) is tested (where *i* denotes the index of an item in a list).

Associative recognition tasks are also common. Like in cued recall, participants study a list of $A_i - B_i$ pairs. Participants then recognize whether a pair is *intact* ($A_i - B_i$) or *rearranged* ($A_i - B_j$ or $A_j - B_i$) by responding *yes* to the intact pairs and *no* to the rearranged pairs when prompted (Kahana, 2012).

We will now begin our discussion of associative models. Remember that the overarching goal of a model is to represent—as accurately as possible—how the brain stores associations. A good model mimics patterns that have been noticed in human subjects during experiments.

Model performance compared to human subjects on memory tasks falls outside the scope of this paper, but it is still something to keep in mind while learning about them.

Matrix Model

The matrix model is named the way it is due to how associations are stored—in matrices. Items are modeled as vectors; each dimension represents an item's specific *feature*, which can include emotional tones or sensory features like colours or odors that occurred during learning (Hintzman, 1986). Feature values are usually generated randomly for simplicity (Caplan, personal communication). Because each item consists of multiple features, the matrix model is considered *distributed* (Kahana, 2012). Liu et. al provide an overview of the matrix model in MATLAB in their 2012 tutorial, from which the following demonstration of the matrix model is based off of.

First, suppose we have an association between two items **a** and **b** (lowercase boldface denotes vectors). The item vectors are normalized and orthogonal because it makes the calculations easier to work with:

```
a = [5;4]
b = [-4;5]
```

Associations are stored in an *association matrix*. Let's create a matrix *M* to store the association **a**-**b**. The association is stored by taking the outer product of vector **a** with vector **b**:

```
M = a*b'
```

where the * performs matrix multiplication and ' denotes transpose.

Let's try retrieving an item. We'll try retrieving item **a** first. A target **a_retrieve** is retrieved by calculating the product between the association matrix and the target's paired associate. Thus, we probe *M* with **b**:

```
a_retrieve = M*b
```

The dot product between a retrieved vector and its corresponding stored item vector returns the similarity between the two, assuming they are normalized and mean-centered,

```
a'*a_retrieve / (norm(a) * norm(a_retrieve))
```

where a maximum value of 1 implies the vectors are maximally similar and a minimum value of 0 implies maximal dissimilarity. We get a value of 1, implying that vectors **a** and **a_retrieve** are identical, which is what we want! We have retrieved item **a** successfully.

Let's now try modeling another association. Suppose we have an association **c**-**d** between items **c** and **d**.

```
c = [6;7]
d = [-7;6]
```

Multiple associations are modeled by adding them to the association matrix. To store the association **c**-**d**, we calculate the outer product of **c** and **d** and add it to our matrix *M*:

```
M = M + c*d'
```

We have now stored two associations in our association matrix! If we want to keep adding associations we simply continue adding them to *M*.

Let's try retrieving an item from our new stored association. We can try retrieving **c**. Like before, we retrieve target **c_retrieve** by probing *M* with **c**'s pair associate **d**:

```
c_retrieve = M*d
```

Let's confirm **c_retrieve** was retrieved correctly by calculating the dot product between **c** and **c_retrieve**:

```
c'*c_retrieve / (norm(c)*norm(c_retrieve))
```

We get a dot product of 1! We successfully retrieved **c_retrieve** and modeled multiple associations in our current matrix model!

So far, we have only tried retrieving in the backwards direction. In the stored association **a**-**b**, probing with **b** to retrieve **a** is probing in the backwards direction, while probing with **a** to retrieve **b** is in the forwards direction.

Let's now try retrieving in the forwards direction. Let's retrieve **b** by probing with **a**:

```
b_retrieve = M*a
```

As before, we calculate the dot product between **b_retrieve** and **b** to confirm we retrieved correctly.

```
b'*b_retrieve / (norm(b) * norm(b_retrieve))
```

We get a value approximately equal to 0, not 1! The current model is *unidirectional*, meaning that we must probe from the right side or noise is returned. If we want a *bidirectional* model, we simply store the backwards and forwards associations in the association matrix. Let's create a bidirectional model using that stores both directions of the associations **a** and **b:**

```
M = a*b'+b*a'
```

We now store the outer products of **a** with **b** and **b** with **a** in a new matrix *M*.

Let's try retrieving in the forwards direction again! We probe with **a** to retrieve **b**:

```
b_retrieve = M*a
```

We check the dot product between **b_retrieve** and **b** and we get a value of 1!

Similarly, let's try retrieving in the backwards direction now. We probe with **b** to retrieve **a**:

```
a_retrieve = M*b
```

We check the dot product between **a_retrieve** and **a** and we get a value of 1!

Either item from an association can now be retrieved when probed with its pair associate.

To summarize, items in the matrix model are vectors composed of many features. Associations are stored as outer products of items in an association matrix. Multiple associations are represented by summating outer products. We can create a model with a unidirectional representation—probing must occur from the right side—or one with a bidirectional representation—probing can be done from either side.

Convolution Model

Convolution models are different from matrix models in the way associations are stored. Although items are also represented as vectors with random feature values—like in the matrix model—associations are created by *convolving* items instead of calculating their outer product (Murdock, 1982; Metcalfe 1982). Items are convolved either *linearly* or *circularly*. The linear convolution of items **f** and **g** (row vectors with *n* features each) produces a *2n-1* element row vector (Murdock, 1982) and (Metcalfe, 1982) whereas the circular convolution of **f** and **g** produces an *n* element row vector (Plate, 1995).

Note that prior to Plate's 1995 paper, convolution models of episodic memory used linear convolution. He noted several advantages of circular convolution that have popularized its use in convolution models since then—reasons that are outside the scope of this paper.

Let's visualize what circular convolution looks like—the illustration below is great:

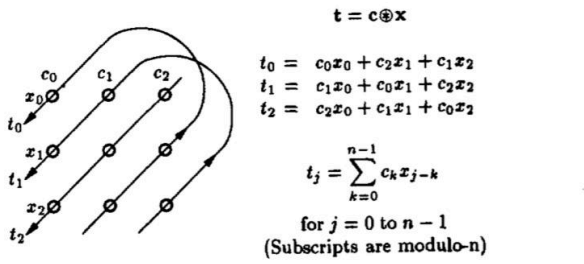

Fig. 4. Circular convolution represented as a compressed outer product for $n = 3$.

Figure 1 [Plate 1995]

Where the **t** output vector results from circularly convolving the **c** and **x** vectors. Intuitively, convolution can be thought of as a compressed outer product (Plate, 1995).

To demonstrate the structure of a convolution model, suppose we have items **f** and **g** from before and we want to associate them. Let's convolve them together:

$$\vec{f} * \vec{g} = \vec{c}$$

Where $*$ denotes convolution and **c** is the convolution of **f** and **g**. Note that arrow-notation is used to denote vectors in equations going forwards.

Like in the matrix model, targets are retrieved by probing with a *cue*. To do so, we *correlate*, which is the approximate inverse operation to convolution, the probe with the convolution vector.

A visualization of circular correlation is illustrated in Figure 2 below:

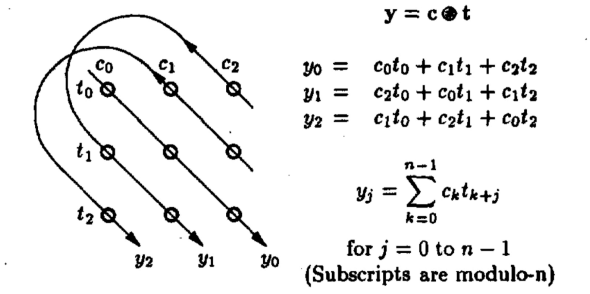

Fig. 5. Circular correlation represented as a compressed outer product for $n = 3$.

Figure 2 [Plate 1995]

Where the **y** output vector results from circularly correlating the **t** and **c** vectors.

Thus, to retrieve **g** we correlate **f** with **c**:

$$\vec{f} \# \vec{c} = \vec{g}_r$$

Where # denotes correlation and $\mathbf{g_r}$ is retrieved.

Because correlation is not an exact inverse of correlation (Plate, 1995) but rather an approximate inverse, $\mathbf{g_r}$ is a *distorted* (or approximate) version of **g**. Like in the matrix model, we use the dot product to calculate the similarity between retrieved and stored items.

The dot product between $\mathbf{g_r}$ and **g** is shown:

$$cos(\Theta) = \frac{\vec{g}_r \cdot \vec{g}}{|\vec{g}_r||\vec{g}|} \approx 1$$

Where Θ is the angle between $\mathbf{g_r}$ and **g.** The dot product is approximately equal to 1—since $\mathbf{g_r}$ is an approximation of **g**.

In a convolution model, multiple associations are stored by summing individual associations—like in the matrix model. Thus, convolutions of paired associates—which are vectors—are summed together to produce a vector of the same size as the convolution vectors.

Suppose we have items **h** and **p** that we want to associate. We also want to store them alongside the **f**-**g** association.

We must convolve **h** with **p** first:

$$\vec{h} * \vec{p} = \vec{d}$$

Where **d** is the convolution of **h** with **p**.

Now, by summating **c** and **d** (which are both vectors) we store the **f**-**g** and **h**-**p** associations in a single vector:

$$\vec{c} + \vec{d} = \vec{e}$$

where **e** stores the **f**-**g** and **h**-**p** associations.

We can now try retrieving an item. Note that circular correlation is distributive. Let's probe with **h** to retrieve **p**:

$$\vec{h} \# \vec{e} = \vec{h} \# (\vec{c} + \vec{d}) = \vec{h} \# (\vec{f} * \vec{g} + \vec{h} * \vec{p}) = \vec{h} \# (\vec{f} * \vec{g}) + \vec{h} \#(\vec{h} * \vec{p}) = noise + \vec{p}_r$$

Where $\mathbf{p_r}$ is the retrieved item.

As before, we calculate the dot product between $\mathbf{p_r}$ and **p** to confirm their similarity:

$$cos(\Theta) = \frac{\vec{p}_r \cdot \vec{p}}{\left|\vec{p}_r\right|\left|\vec{p}\right|} \approx 1$$

Because we have not actually provided any values to our vectors, we cannot calculate the dot product, but it should be approximately equal to 1.

Like in the matrix model, items in the concatenation model are modeled as feature vectors. To associate them, however, we convolve them instead of calculating their outer product. To calculate the similarity between a retrieved vector and a cue we use the dot product, like in the

matrix model. If we want to retrieve a target, we correlate the cue with the convolution vector. Finally, multiple associations are represented as sums of individual associations, just like in the matrix model.

Concatenation and Local Trace Models

In *concatenation* models, associations are modeled—like the name suggests—as concatenations of items (Shiffrin and Steyvers 1997; Criss and Shiffrin, 2004). Like in the matrix and concatenation models, items are usually modeled as vectors;associations are simply concatenated vectors. *Local trace* models are unique in the way that memory consists of a set of separate items—not conjoined into one representation—unlike in matrix and convolution models (Caplan, personal communication). There is a strong overlap between concatenation and local trace models—many models are both—which is why they are discussed together in this paper. We begin our discussion with Hintzman's (1986) MINERVA 2 model, moving towards Shiffrin and Steyvers (1997) REM.2 model and finishing with Rizzuto and Kahana's (2001) autoassociative neural network model.

Note that the section on MINERVA 2 is based on Hintzman's 1986 paper. With that in mind, we can begin learning about the model.

To begin with, MINERVA 2 is a distributed, concatenation-based, local-trace model. *Experiences*—or items—are strings of features (*primitive properties* as Hintzman refers to them).

There are much less primitive properties than there are *experiences*—implying that experiences overlap (either more or less based on the number of shared properties).

Two memory systems make up the model—a *primary* memory (PM) system representing the current experience and a *secondary* memory (SM) system which holds pre existing memory traces. The memory systems communicate by two mechanisms: 1.) a probe—or new experience—is sent from PM to the *traces*—or stored memories—in SM in parallel, and 2.) an *echo*—which has an intensity and content property—is sent from SM to PM. (Hinztman, 1986). The former varies based upon the similarity of all traces to the probe while the latter represents the traces' specific activation pattern. Activation of a trace activates all of its primitive properties, even those not shared with the probe, forming the foundation for associative learning.

Features in MINERVA 2 are represented as strings of integers. Feature j equaling -1 represents excitation, -1 is inhibition, and 0 indeterminate. Feature values are generated randomly for simplicity, just like in the matrix model and concatenation models. Learning happens by copying experience features into a new trace, where each feature has probability P of being encoded and probability F of being forgotten (reverting from -1 or +1 to 0).

SM traces are activated by the following equation:

$$S_i = (1/N_R) \sum_{j=1}^{n} P(j)T(i,j)$$

where N_R denotes the number of relevant properties such that $P(j)$ and $T(i,j)$ are non-zero (as otherwise the sum is 0). S_i is similar to a Pearson *r* (or dot product), which measures the linear relationship between two normally distributed variables (Newcastle University, n.d), such that S_i ranges from $-1 \leq 0 \leq +1$ (S_i is 0 if the probe and trace are orthogonal). Feature position is indexed by j, which ranges from 1, 2, …, n (where n is the total number of primitive properties. Traces are indexed by i, which ranges from 1, 2, …, m (where m is the number of traces in SM). The value of the *j*th feature of the probe is denoted by $P(j)$ while the value of the *j*th feature of trace *i* is represented by $T(i,j)$. Finally, the similarity of trace *i* to the probe is denoted $S(i)$.

Hintzman does not explicitly discuss associations in his 1986 paper and thus modeling them within a MINERVA 2 framework is not within the scope of this paper. Discussion will now shift to the REM.2 model.

REM.2, like MINERVA 2, is a distributed local-trace concatenation model. Criss and Shiffrin (2004) discussed storing associative memory within a REM.2 model framework, meaning that we can demonstrate with examples what it would look like.

Suppose we have association **a**-**b** between items **a** and **b**. Note that items in a REM.2 framework are typically modeled with 20 features each (Shiffrin and Steyvers, 1997) but we will use 2 for simplicity. Vectors **a** and **b** are shown below:

$$\vec{a} = [c, d]$$

$$\vec{b} = [f, g]$$

Where c, d, f, g are feature values.

Let's concatenate our items together to model the association. The concatenation of **a** with **b** is:

$$\vec{a} \,\&\, \vec{b} = [c, d, f, g]$$

Where concatenation is denoted &.

We have modeled an association! Note that Shiffrin and Steyvers introduce a probability parameter u of a feature being stored after it is concatenated. The possibility of *limited capacity* is also discussed where there exists a maximum number of features that can be stored, as well as the option to vary the rate of storage inversely with the number of features. Furthermore, some features may be stored incorrectly and some features may not copy over at all.

We will avoid further discussion of these modifications for the sake of simplicity.

Let's now try storing multiple associations in a concatenation model. Suppose there exists another association **h-i** between items **h** and **i**:

$$\vec{h} = [j, k]$$

$$\vec{i} = [l, m]$$

Where j, k, l, m are feature values.

Like before, we concatenate **h** and **i**:

$$\vec{h} \,\&\, \vec{i} = [j, k, l, m]$$

We have now modeled multiple associations in a REM.2 model!

Modeling retrieval in a cued recall context falls outside the scope of Shiffrin and Steyver (1997) since the associative memory application of REM is not the paper's focus. However, Criss and Shiffrin (2004) build upon the associative memory application of concatenation models.

In their paper, they describe how associative recognition can be modeled using a concatenation model: a probe—an association modeled by the concatenation of two item vectors— is simply compared with each stored concatenation. The higher the similarity between a cue and a stored vector—what Criss and Schiffrin refer to as a *familiarity measure* that is calculated as a "product of evidence from each feature"—the more likely they represent the same association. Note that similarity is measured here using a *likelihood ratio*, which measures how likely the observed data for each feature (match or mismatch between the probe and memory image) would occur if the memory image represents the same image (s-image) versus a different d-image. A bayesian decision process is used to determine if a test item has been previously seen or not. Modeling this falls outside of the scope of this paper but we will show a simple abstraction nonetheless.

Let's try modeling it!

Consider the association **a**-**b**—modeled as the concatenation of items **a** and **b**:

$$\vec{a}\,\&\,\vec{b} \;=\; [j, k, l, m]$$

where j, k are feature values of **a** and l, m are feature values of **b**. Concatenation is denoted by &.

Now suppose we have another association **c-d**—modeled as the concatenation of items **c** and **d**:

$$\vec{c}\,\&\,\vec{d} \;=\; [n, o, p, q]$$

where n, o are feature values of **c** and p, q are feature values of **d**.

Let's try modeling retrieval! Suppose we have a probe **a-b**. We compare the feature values of our probe with our stored associations to confirm whether this association exists in memory (since this is an associative recognition task).

Let's calculate the similarity between probe **a-b** and each stored association with the likelihood ratio.

The likelihood ratio of probe **a-b** is calculated by comparing its feature values against each stored association. The likelihood ratio between **a-b** and stored association **c-d** would be low, reflecting low similarity due to a lack of shared feature values. However, the likelihood ratio between probe **a-b** and stored association **a-b** would be high (since they share many feature values)

.

A key difference between MINERVA 2 and the convolution and matrix models is how experiences are stored: the former assumes a new memory trace for each experience while the latter models simply create a compound trace—a matrix in the matrix model and a convolution vector in the convolution model.

We will now discuss Kahana and Rizzuto's autoassociative neural network model. The strength of their model lies in the ability to accommodate two very extreme views of associative learning. The first is the *independent associations hypothesis* (IAH) that says that an association's strength is based on the *temporal order*, or order in which information was presented to the brain while encoding (Ebbinghaus, 1885/1913; Robinson, 1932).

Let's show what this looks like with an example.

Suppose there exist two items **a** and **b**. If they are encoded in the order **a** then **b**, then the forwards association **a-b** will be stronger than the backwards association (Kahana and Rizzuto, 2001).

The other theory is the *associative symmetry hypothesis* (ASH), which says that forwards and backwards associations are close to equal (Kahana and Rizzuto, 2001).

Let's show what this would look like with the same items **a** and **b**:

If they are encoded in the order **a** then **b**, ASH says that the forwards association **a**-**b** will be equal to the backwards association (Kahana and Rizzuto, 2001).

How can a model accommodate both these views, each the polar opposites of one another?

Kahana and Rizzuto actually did exactly that using a concatenation model:

$$W = \sum_{V=1}^{L} (a^{v} \oplus b^{v})(a^{v} \oplus b^{v})^{T}$$

This is the storage equation for a list of L pairs where a^{v} and b^{v} are binary $\pm$ 1 N element vectors representing items that will be associated. $\oplus$ denotes concatenation and W is a $2N \times 2N$ weight matrix.

Although this equation may seem complicated at first, it simply describes how item vectors are concatenated to represent an association. The outer product of an association **j** with itself—where **j** denotes any given association—is then summated with the outer products of each association in the list from V to L—where V is the first association and L is the length of the list. These are summated into a memory matrix with four quadrants. Quadrants 1 and 3 have *autoassociative* information—an item is associated with itself—while 2 and 4 contain

heteroassociative information—an item is associated with a different item (Caplan, personal communication).

Two random variables, γ_f and γ_b, are also introduced to control forwards and backwards directional learning. The idea is that based on how correlated these two variables are to one another the model implements either ASH or IAH, determining the extent to which the forwards or backwards association is learned for any given **a**-**b** pair (Kahana and Rizzuto, 2001). Further statistical explanation is outside the scope of this paper.

Retrieval occurs by comparing the state of the network with a target item—both represented by vectors. Their similarity is calculated as the cosine of the angle between them such that if it is greater than a threshold Θ then a target is said to be recovered (Kahana and Rizzuto, 2001).

MINERVA 2, REM.2, and the autoassociative neural network model all modeled items as feature vectors—just like the matrix and concatenation models. Associations were modeled in REM.2 by concatenating item vectors. The autoassociative neural network model also concatenated item vectors but the outer product of an association with itself was then taken and stored into a matrix model. Despite REM.2 and the autoassociative neural network model both storing associations as concatenations, associations were stored in a modified matrix model in the autoassociative neural network.

Conclusion

We have learned about memory tasks and gone over various kinds of memory models. We can now discuss some of the similarities and differences between them. Notice how items were represented as vector features in *every single model* we discussed. Representing items in this way makes them very easy to work with. For one, we can use a straightforward operation—the dot product—to calculate similarity between a cue and a target.

Secondly, it allows us to apply all kinds of useful operations on them while not losing the inherent properties of an item that allows us to differentiate them. We took outer products, convolved, concatenated, and even had a model do a mix of both concatenation and taking the outer product. We began with a collection of n association vectors that we weren't sure what to do with but we were able to reduce their representations to just one representation—a matrix in the matrix model, a convolution vector in the convolution model, and a matrix in the autoassociative neural network.

Although the focus of this paper was initially meant to solely be on modeling associative memory, I think it simultaneously (and non-intentionally) showcases the brilliant versatility of linear algebra. My hope is that you, the reader, is just as amazed as I am by how powerful of a language it is and will also appreciate it a lot more than simply having to learn it for a course that needs to be taken as a computing science prerequisite.

References

Criss, A. H., & Shiffrin, R. M. (2004). Pairs do not suffer interference from other types of pairs or single items in associative recognition. *Memory & Cognition*, *32*(8), 1284–1297. https://doi.org/10.3758/BF03206319

Eich, J. M. (1982). A composite holographic associative recall model. *Psychological Review*, *89*(6), 627–661. https://doi.org/10.1037/0033-295X.89.6.627

Hintzman, D. L. (1986). "Schema abstraction" in a multiple-trace memory model. *Psychological Review*, *93*(4), 411–428. https://doi.org/10.1037/0033-295X.93.4.411

Kahana, M. J. (2012). *Foundations of human memory. [Electronic resource]* (University of Alberta - Online Resources Internet Access). Oxford University Press.

Murdock, B. B. (1982). A theory for the storage and retrieval of item and associative information. *Psychological Review*, *89*(6), 609–626. https://doi.org/10.1037/0033-295X.89.6.609

Numeracy, Maths and Statistics—Academic Skills Kit. (n.d.). Retrieved June 12, 2024, from https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/statistics/regression-and-correlation/strength-of-correlation.html

Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks*, *6*(3), 623–641. https://doi.org/10.1109/72.377968

Rizzuto, D. S., & Kahana, M. J. (2001). An autoassociative neural network model of paired-associate learning. *Neural Computation*, *13*(9), 2075–2092. https://doi.org/10.1162/089976601750399317

Shiffrin, R. M., & Steyvers, M. (1997). A model for recognition memory: REM—retrieving effectively from memory. *Psychonomic Bulletin & Review*, *4*(2), 145–166. https://doi.org/10.3758/BF03209391