The Memory Tesseract:  
Distributed MINERVA and the Unification of Memory

Matthew A. Kelly (matthew_kelly2@carleton.ca) 1  
Douglas J. K. Mewhort (mewhortd@queensu.ca) 2  
Robert L. West (robert_west@carleton.ca) 1

1 Institute of Cognitive Science, Carleton University  
1125 Colonel By Drive, Ottawa, Ontario, K1S 5B6 Canada  
2 Department of Psychology, Queen’s University  
62 Arch Street, Kingston, Ontario, K7L 3N6 Canada

Abstract
We prove that MINERVA 2, a widely-used model of biological long-term memory, is mathematically equivalent to an auto-associative memory implemented as a fourth order tensor. We further propose an alternative implementation of MINERVA 2 as a holographic lateral inhibition network. Our work clarifies the relationship between MINERVA 2 and other memory models, and shows that MINERVA 2 and derivative models can be neurally implemented and scaled-up to long-term learning tasks.

Keywords: memory; cognitive modelling; MINERVA 2; vectors; tensors; holographic reduced representations; HRRs.

Introduction
Most memory phenomena can be explained by unified, computational memory models (e.g., Franklin & Mewhort, 2013; Hintzman, 1984; Humphreys, Bain, & Pike, 1989; Jamieson & Mewhort, 2011). Simulation has led to parsimonious theories of memory, but at a cost of a profusion of competing models. As different models focus on different phenomena, there is no best model.

Simulation models share many characteristics indicating wide agreement about the mathematics of how memory works. Here, we argue that memory models, including the MINERVA 2 (Hintzman, 1984) model and variants (e.g., Jamieson, Crump, & Hannah, 2012; Jamieson & Mewhort, 2011; Kwantes, 2005; Thomas et al., 2008), as well as holographic models of short/long term memory (e.g., Eich, 1982; Franklin & Mewhort, 2013; Murdock, 1993), and the DSHM model of declarative memory (Rutledge-Taylor & West, 2008) which uses the BEAGLE learning algorithm (Jones & Mewhort, 2007), can be understood in terms of a single neurally plausible memory framework.

This effort at unification is based on the MINERVA 2 model (Hintzman, 1984), a computational model of biological memory, intended by Hintzman to describe long-term memory (both episodic and semantic). It has been applied to several experimental paradigms, including judgement of frequency tasks (Hintzman 1984), recognition tasks (Hintzman, 1984), “schema-abstraction” or category learning (Hintzman, 1984; 1986), implicit learning tasks such as artificial grammar learning (Jamieson & Mewhort, 2011), as well as speech perception (Goldinger, 1998), and naming words from print (Kwantes & Mewhort, 1999).

How does MINERVA work?

Hintzman (1986) describes MINERVA’s key assumptions: (1) only episode traces are stored in memory, (2) repetition produces multiple traces of an item, (3) a retrieval cue contacts all traces simultaneously, (4) each trace is activated according to similarity to the cue, (5) all traces respond in parallel, the retrieved information reflecting their summed output.

Variations on the MINERVA 2 model address an even broader range of phenomena. MINERVA-AL makes and corrects predictions to capture numerous associative learning phenomena from both the animal and human learning literature (Jamieson, Crump, & Hannah, 2012). Kwantes (2005) used MINERVA to study how semantic similarity can be learned from word co-occurrence in the language. Thomas et al. (2008) used MINERVA to study hypothesis generation and probability judgement in humans. In this paper, we use the term MINERVA 2 to refer specifically to Hintzman’s model, and MINERVA to refer collectively to MINERVA 2 and any model based on it.

Our central contribution is to prove that MINERVA is mathematically equivalent to an auto-associative fourth order tensor memory, or memory tesseract. We further demonstrate that MINERVA is approximately equivalent to a holographic lateral inhibition network (Levy & Gayler, 2009). These demonstrations have three implications for memory modelling: (1) the relationship between MINERVA and other memory models is clarified, suggesting that MINERVA may be suitable as a basis for all memory modelling, (2) MINERVA is scalable to arbitrarily long-term learning, and (3) MINERVA is neurally plausible.
vector in the table “resonates” with the cue in proportion to its similarity to the cue. Similarity is computed as a cosine (i.e., normalized dot-product) of the cue with the stored vector. Each vector is activated by its cubed similarity to the cue. Information is retrieved from memory in the form of a new vector, called an echo. The echo is a weighted sum of the vectors in the table, each vector weighted by its activation. By computing activation as the cube of similarity, the contribution of the most similar vectors (or experiences) is emphasized and that of the least similar (and least relevant) is minimized. The echo is used by the model to respond to the cue as appropriate for the given task.

Abstract, conceptual, and categorical information reflect aggregate retrieval over many episode traces. The blending of experiences in the echo is one source of our ability to use abstractions (e.g., Goldinger, 1998).

According to Hintzman (1990), MINERV A 2 can be understood as an artificial neural network. A layer of input nodes represent the cue. A layer of output nodes represent the echo. Between the two is a hidden layer of nodes. In the hidden layer, each node corresponds to an episode trace. It follows that MINERV A’s hidden layer is a localist network: specific nodes represent specific pieces of information.

Modellers using MINERV A are generally agnostic as to how the model is related to the brain. No one claims that for each new experience, one grows a new neuron that is forever singly dedicated to that particular experience. But no other interpretation of how MINERV A is related to the brain has been previously proposed, leaving open the question of MINERV A’s neural and, hence, theoretical plausibility.

A comparison of memory models

The memory models discussed here use vector and tensor representations to simulate the processes of storage and retrieval. Tensors are a generalization of matrices. A vector is a first order tensor. A matrix is a second order tensor. A third order tensor is a “3D matrix” or a stack of matrices.

Memory models can use either localist or distributed representations and have either localist or distributed memory stores. Vector-based memory models use distributed representations, that is to say, an item to be remembered is represented as a high dimensional vector, which can be thought of as a pattern of activation across nodes in a network. Conversely, in a network that uses localist representation, an item is represented by the activation of a single node, as opposed to a pattern of activation across a group of nodes.

In vector-based memory models, the memory store can be either localist or distributed. By a localist store, we mean that a model stores different data in different places. By a distributed store, we mean a model that stores all data in a single place or all data in all places. For our purposes, vector-based memory models can be divided into four categories: vector memory, where all memories are stored in a single vector; matrix memory, where all memories are stored in a single matrix; tensor memory, where all memories are stored in a single, higher-order tensor (e.g., a “3D” or “4D” matrix), and multi-vector memory, where multiple vectors are used to store memories. Multi-vector memory models, such as MINERV A, and DSHM (Rutledge-Taylor & West, 2008) use localist stores because different vectors are used to store different memories, whereas vector (e.g., Eich, 1982; Franklin & Mewhort, 2013; Murdock, 1993), matrix (e.g., Humphreys, Pike, Bain, & Tehan, 1989; Howard & Kahana, 2002), and tensor memory models (e.g., Humphreys, Bain, & Pike, 1989; Smolensky, 1990) use distributed stores. Across these four categories, there are strong similarities between models, as we will now discuss.

Matrix and tensor models

Humphreys et al. (1989) note that their matrix memory, MINERV A 2, and TODAM (Murdock, 1993) retrieve information as a sum of all traces in memory, each trace weighted by its similarity to the cue. As we will show, these models are not different in kind, and so we elect to use the term echo, normally reserved for MINERV A, to refer to the retrieved vector in all of these memory models.

A key point of comparison is how vector-based models represent associations between items. Smolensky (1990) notes that the tensor product, a generalization of the matrix product, can be used to form associations between an arbitrary number of items, though at the cost of producing progressively larger and more unwieldy tensors.

The order of the tensor used to store memories indicates the number of vectors the memory model associates together. An association between a pair of items is represented as the tensor product of the items’ vectors, which is a second order tensor, or matrix. A matrix memory is the sum of those associations. Howard and Kahana’s (2002) matrix memory is context \( x \) item, that is, it associates a vector representing an item with a vector representing a context. Humphreys et al. (1989) matrix memory is item a \( x \) item b, that is, it associates two different items together.

If three items are being associated, the result is a “3D matrix”, or third order tensor. Humphreys, Bain, and Pike’s (1989) third order tensor memory is context \( x \) item \( a \) \( x \) item b, that is, it associates two different items together with a representation of context.

There is a problem here. Across these models, the architecture of memory is being modified to suit the particulars of the tasks being modelled. If we just need one cue (be it an item or context), we use a matrix memory. If we use two cues (be it an item and context, or two items) we use a third order tensor. But what if we need to use three cues? Do we then need to use a fourth order tensor? What about four cues? Using this approach, not only does the architecture of memory need to be changed depending on the particulars of the task, but the architecture becomes increasingly unwieldy as the task becomes more complex.

Vector models

Holographic vectors can represent arbitrarily complex associations of items and context, making it unnecessary to use matrices or higher order tensors to represent association, allowing modellers to adopt a memory architecture that is invariant with respect to the complexity of the associations.

In a vector memory, an association between a set of items is the convolution of the vectors representing those items. Convolution is a noisy compression of the tensor-product (Plate, 1995; for discussion see Kelly, Blostein, & Mewhort, 2013), thus vector models differ from matrix and tensor...
models only in that the highly lossy nature of holographic vector storage adds noise to the echo, and that individual items and associations between sets of any size can all be stored together as a sum in a single vector memory.

Holographic vectors do have one weakness: they are highly lossy. This is the only reason one might prefer the aforementioned matrix or tensor memories over a holographic one. However, combining holographic vectors with MINERVA creates a system that can store arbitrarily complex associations between items and contexts, and retrieve them with fidelity (Jamieson & Mewhort, 2011).

MINERVA

MINERVA 2 differs from vector or matrix models in that:
1. MINERVA 2 associates items by either adding together or concatenating the vectors representing those items, rather than using the tensor-product or convolution.
2. All traces in the echo of MINERVA 2 are weighted by the cubed similarity to the cue. In a vector or matrix model, the similarity is not raised to an exponent.

The first difference is not essential. The Holographic Exemplar Model (Jamieson & Mewhort, 2011) is a MINERVA that uses convolution rather than concatenation, gaining the ability to represent arbitrarily complex associations.

The second difference is critical. Raising similarity to an exponent of 3 sets the MINERVA models apart from the vector model by using nonlinearity. In particular, the exponent of 2 is equivalent to a third order tensor. Finally, we prove that the MINERVA 2 model, which uses an exponent of 3, is equivalent to a fourth order tensor.

Consider a variant on the MINERVA model that uses dot product (denoted by •) to measure similarity and weights each trace by its similarity raised to the exponent of 1. Each episode trace in memory is represented by a vector \( v \) where \( i = 1 \ldots m \) and \( m \) is the number of traces in memory. When the model is presented with a cue \( x \), the echo \( y \) is:

\[
y = (xv_1)v_1 + \ldots + (xv_m)v_m
\]

This is equivalent to an auto-associative matrix memory.

In an auto-associative matrix memory, each episode trace is represented by a vector \( v \). To store a trace in memory, the trace is associated with itself (hence auto-associative) by taking the outer-product of the vector with itself, \( vv^T \), then taking the sum of all the outer-product matrices to create the memory matrix, \( M \):

\[
M = vv^T + \ldots + mm^T
\]

The echo, \( y \), is the inner-product of the cue and the matrix:

\[
y = Mx
\]

\[
y = (v_1v_1^T + \ldots + v_mv_m^T)x
\]

\[
y = v_1v_1^Tx + \ldots + v_mv_m^Tx
\]

Because \( vv^T \) is the dot-product of \( v \) and \( x \):

\[
y = (xv_1)v_1 + \ldots + (xv_m)v_m
\]

which is identical to MINERVA with an exponent of 1 or to a matrix memory (e.g., Humphreys et al., 1989). Consider a MINERVA that raises similarity to the exponent of 2:

\[
y = (xv_1)^2v_1 + \ldots + (xv_m)^2v_m
\]

This variant of MINERVA, as we shall demonstrate, is mathematically equivalent to an auto-associative third order tensor memory. Using the tensor product, denoted by \( \otimes \), we
can store each trace as \( v_i \otimes v_i \otimes v_i \), which is a third order tensor. The memory tensor \( M \) is the sum of the third order tensor outer-products of each episode trace:
\[
M = v_i \otimes v_i \otimes v_i + ... + v_m \otimes v_m \otimes v_m
\]
The echo, \( y \), can be computed from the cue, \( x \), by taking the inner product twice:
\[
y = (M \cdot x) \cdot x
\]
If each \( v_i \) is a vector of \( n \) dimensions, then \( M \) is an \( n \times n \times n \) tensor. \( M \) can be thought of as \( n \) matrices of \( n \times n \) dimensions. When we compute the inner-product of the \( M \) with the \( x \), we compute the inner product of \( x \) with each of those \( n \) matrices. This results in \( n \) vectors that can be rearranged into a new \( n \times n \times n \) matrix. The second inner product with \( x \) then produces a vector, the echo \( y \).

To illustrate, let us break \( M \) into its components:
\[
y = (M \cdot x) \cdot x
y = ((v_1 \otimes v_i \otimes v_i + ... + v_m \otimes v_m \otimes v_m) \cdot x) \cdot x
y = (v_i \otimes v_i \otimes v_i \cdot x + ... + v_m \otimes v_m \otimes v_m \cdot x) \cdot x
\]
The tensor product \( v_i \otimes v_i \otimes v_i \) can be understood as \( n \) matrices, where each matrix is the outer-product \( v_i v_i^T \) weighted by a different element \( j \) of \( v_i \), for all \( j = 1 \) \ldots \( n \).
\[
v_i \otimes v_i \otimes v_i = \{v_i v_i^T, ..., v_n v_n^T\}
\]
Taking the inner-product of the cue \( x \) with each \( v_i \otimes v_i \otimes v_i \), we get \( n \) vectors, each weighted by the dot-product of \( x \) with \( v_i \):
\[
v_i \otimes v_i \otimes v_i \cdot x = \{v_i v_i^T \cdot x, ..., v_n v_n^T \cdot x\}
\]
If we factor out the dot-product of \( x \) and \( v_i \), the result is \( n \) vectors, or rather, the outer-product matrix of \( v_i v_i^T \):
\[
v_i \otimes v_i \otimes v_i \cdot x = (x \cdot v_i \cdot v_i) \{v_i v_i^T, ..., v_n v_n^T\}
\]
Thus, when we take the outer-product of \( x \) with \( M \), the result is a sum of \( m \) matrices \( v_i v_i^T \), each matrix weighted by the dot-product of \( x \) with \( v_i \).
\[
y = ((x \cdot v_i) \cdot v_i v_i^T) \cdot x + ... + (x \cdot v_m) \cdot v_m v_m^T \cdot x
\]
By then taking the second inner-product with \( x \), we reduce each matrix to a vector weighted by the squared similarity to \( x \), producing an echo like MINERVA with an exponent of 2:
\[
y = (x \cdot v_i) v_i v_i^T \cdot x + ... + (x \cdot v_m) v_m v_m^T \cdot x
\]
The dot product of two vectors, where each vector has a magnitude of one, produces a value in the range of one, if the vectors are identical, to zero, if the vectors are orthogonal, to negative one, if one vector is the negation of the other. Thus it is important to preserve the sign of the dot product. By taking the square of the dot product, the sign is lost. For this reason, MINERVA 2 uses an exponent of 3.

MINERVA 2 is equivalent to an auto-associative memory implemented as a fourth order tensor. Memory is constructed as a sum of fourth order tensors:
\[
M = v_i \otimes v_i \otimes v_i \otimes v_i + ... + v_m \otimes v_m \otimes v_m \otimes v_m
\]
Given a cue \( x \), an echo \( y \) is computed by taking the inner product three times:
\[
y = ((M \cdot x) \cdot x) \cdot x
\]
\[
y = ((( v_i \otimes v_i \otimes v_i \otimes v_i + ... + v_m \otimes v_m \otimes v_m \otimes v_m) \cdot x) \cdot x) \cdot x
\]
\[
y = ((( v_i \otimes v_i \otimes v_i \otimes v_i + ... + (x \cdot v_i) v_i v_i^T \cdot v_m v_m^T \cdot v_m v_m^T) \cdot x) \cdot x)
\]
\[
y = ((x \cdot v_i \cdot v_i \cdot v_i \cdot v_i)^3 + ... + (x \cdot v_m \cdot v_m \cdot v_m \cdot v_m)^3) \cdot x
\]
\[
y = (x \cdot v_i)^3 \cdot v_i + ... + (x \cdot v_m)^3 \cdot v_m
\]

**Is the memory tesseract practical?**

MINERVA is equivalent to a distributed memory system implemented as an auto-associative fourth order tensor, or *memory tesseract*. Unfortunately, fourth order tensors are very large. For most applications of MINERVA, the dimensionality \( n \) of a vector will be larger than the number of memories \( m \) stored in the model. MINERVA, as standardized implemented, is an \( m \times n \) table, whereas a memory tesseract is an \( n^4 \) data structure. A typical MINERVA 2 has \( 10 \leq n \leq 200 \). In general, the number of memories stored is smaller than \( n \) and *much* smaller than \( n^3 \). For applications where \( m < n^3 \), the implementation of MINERVA as a table is more efficient. However, for large scale applications where \( m \geq n^3 \), the fourth order tensor is more efficient.

Alternatively, a holographic approximation to the memory tesseract can be implemented as an \( n \times p \) data structure for the \( p \) of your choice, as is discussed in the next section.

**Using holographic vectors rather than tensors**

In a holographic vector system, trying to clean-up the echo by iteratively using the echo as a cue to retrieve a new echo is like trying to clean a pair of glasses with an oily cloth: the more you try to clean it, the worse it becomes. Yet Levy and Gayler (2009) have demonstrated that this is possible using a lateral inhibition network implemented as a fully distributed vector architecture. To store a trace \( v_i \) in Levy and Gayler’s model, the trace is associated with itself twice, then each trace is added to the memory vector \( m \):
\[
m = v_i \cdot v_i \cdot v_i + ... + v_m \cdot v_m \cdot v_m
\]
where * is a binding operation. In holographic reduced representations (Plate, 1995), binding uses circular convolution and unbinding uses circular correlation. Given a cue, \( x \), we can unbind, denoted by #, to recover an echo:
\[
y = x # (x \otimes m)
\]
Unbinding is such that given a bound pair, \( v_i v_j \),
\[
x \otimes v_i v_j = (x \cdot v_i v_j) \cdot v_j + \text{noise}
\]
Thus:
\[
y = x # (x \otimes (v_i \cdot v_i \cdot v_i + ... + v_m \cdot v_m \cdot v_m))
\]
\[
y = x # ((x \cdot v_i v_i) \cdot v_i + ... + (x \cdot v_m v_m) \cdot v_m + \text{noise})
\]
\[
y = (x \cdot v_i)^3 \cdot v_i + ... + (x \cdot v_m)^3 \cdot v_m + \text{noise}
\]
However, if we wish to imitate MINERVA 2 as closely as possible (and preserve the sign of the similarity) we need to add another association to Levy and Gayler’s model. We compute the memory vector, \( m \), and unbind the echo, \( y \), as:
\[
m = v_i \cdot v_i \cdot v_i + ... + v_m \cdot v_m \cdot v_m
\]
\[
y = x # (x \otimes m)
\]
\[
y = x # ((x \otimes (v_i \cdot v_i \cdot v_i + ... + v_m \cdot v_m \cdot v_m)))
\]
\[
y = (x \cdot v_i)^3 \cdot v_i + ... + (x \cdot v_m)^3 \cdot v_m + \text{noise}
\]
While this allows the most similar traces to the cue to dominate the echo, the noise term threatens to overwhelm the signal. Because holographic vectors use lossy compression, by iterating, the noise will only grow.

Levy and Gayler solve this problem by using random permutations. Given three random permutation matrices \( P_1, P_2, P_3 \), we can permute the vectors as follows:
\[
\begin{align*}
m &= (P_1 v_i) \cdot (P_2 v_i) \cdot (P_3 v_i) \cdot v_i + ... + (P_1 v_m) \cdot (P_2 v_m) \cdot (P_3 v_m) \cdot v_m
\end{align*}
\]
To recover an echo, we can use inverse permutations:
**The semantic tesseract: Scaling MINERVA up**

Applying MINERVA to large semantic memory tasks, such as learning the meaning of words (Kwantes, 2005) or learning how to sound-out written words (Kwantes & Mewhort, 1999), requires abandoning a key assumption of MINERVA for purely pragmatic reasons, namely, that repetition of an item produces multiple traces of that item. But if these models were re-implemented using the memory tesseract (or its holographic approximation), we would not have to abandon that key assumption because the tesseract does not grow with the addition of new memories.

**BEAGLE (Jones & Mewhort, 2007)** is a learning algorithm that models how people abstract the meaning of words from their lifetime language experience. DHSM (Rutledge-Taylor & West, 2008) uses a similar approach to BEAGLE but re-purposes the algorithm as a general memory model. DHSM is a multi-vector memory system, like the standard implementation of MINERVA, but unlike MINERVA, each vector stands for a concept rather than an individual experience. In DHSM, each experience updates the vector for each concept that comprises the experience. An experience is stored in the associations between concept vectors.

In MINERVA, concepts are echoes, artifacts of aggregate retrieval across experiences. We speculate that the vectors in BEAGLE and DHSM are akin to echoes. The memory tesseract now provides a means of re-implementing BEAGLE or DHSM as MINERVA models.

In BEAGLE, an experience is a sentence. To re-implement BEAGLE in the memory tesseract, each time a word is encountered in a sentence, memory is updated with a new episode trace by summing a new fourth order tensor with the memory tensor:

\[ M_i = M_i + v_{i\text{orth}} \otimes v_i \otimes v_i \otimes v_i \]

Each episode trace is the sum of a word’s orthographic information as calculated in Cox et al. (2011) with the word’s semantic information gleaned from that particular sentence, as calculated in Jones and Mewhort (2007):

\[ v_i = v_{i\text{orth}} + v_{i\text{semantic}} \]

However, holographic vectors typically have \( n \geq 512 \). 5124 is about 1678 times larger than the BEAGLE model. If we assume that BEAGLE and MINERVA use vectors of the same size, given a literate adult with a vocabulary of 80 000 words, the memory tesseract will be larger than BEAGLE for vectors with more than 43 dimensions.

If we were to run a memory tesseract model that learns word meaning from context by processing a large corpus, as BEAGLE does, we would need to keep \( n \) as small as possible without sacrificing too much fidelity. We could also use the holographic approximation to the memory tesseract, which is, again, scalable depending on the desired fidelity.

The proposed model escapes being just a less efficient version of BEAGLE because it raises similarity to the exponent of 3. BEAGLE stores all experiences of a particular word in a single vector. This storage is highly lossy and the individual experiences are difficult to recover. MINERVA retains more of each experience, information that can be recovered by iterating to “clean up” the echo.

A memory tesseract implementation of BEAGLE would have another benefit. Individual vectors in BEAGLE are sensitive to first-order (word co-occurrence) and second order (synonymy) associations. Kwantes (2005) found that aggregating across vectors with first-order associations produces an echo with second order associations. Likewise, we suspect that aggregating across BEAGLE’s vectors would produce an echo with third order associations. Third-order associations (and higher) may be useful for identifying part of speech. This change to BEAGLE is a benefit of adopting the MINERVA memory retrieval mechanism.

The preceding discussion is intended only as an extended example of how MINERVA, when implemented as a tensor, can be applied to larger scale problems.
Conclusion

We demonstrate that the influential MINERVA 2 (Hintzman, 1984) model is mathematically equivalent to an autoassociative fourth order tensor memory system, or memory tesseract. We further show that this is approximately equivalent to a variant of the holographic lateral inhibition network proposed by Levy and Gayler (2009). These demonstrations have three theoretical implications:

1. Viewing MINERVA 2 and its variants (collectively, MINERVA models) as a fourth order tensor clarifies the relationship between MINERVA and third order, second order (i.e., matrix), and compressed tensor (i.e., holographic vector) memory models, allowing us to move toward a unified understanding of memory.

2. MINERVA can be scaled up to model long term learning, broadening the scope of tasks to which MINERVA can be applied and allowing for unification with models of semantic learning, such as BEAGLE.

3. A naïve neural interpretation of MINERVA might suggest that a new neuron is grown for each new experience, corresponding to the addition of another row to MINERVA’s memory table. Understood as a memory tesseract, MINERVA is fully distributed across neural connectivity, and memories can be added by changing the connectivity without requiring additional neural resources. Eliasmith’s (2013) neural engineering framework provides a system for translating linear algebra computations (e.g., convolution, permutation) into spiking neuron models. The memory tesseract, or its holographic approximation, could be easily implemented as a realistic neural model using Eliasmith’s framework.

References


