Journal of Mathematical Psychology 77 (2017) 142–155

Contents lists available at ScienceDirect

Journal of Mathematical Psychology

journal homepage: www.elsevier.com/locate/jmp

The memory tesseract: Mathematical equivalence between composite and separate storage memory models

Matthew A. Kelly[a,*], D.J.K. Mewhort[b], Robert L. West[a]

[a] *Carleton University, Canada*

[b] *Queen's University, Canada*

HIGHLIGHTS

- Comparative analysis points towards a unified mathematical basis for memory models.
- MINERVA 2 is proven to be equivalent to a fourth order tensor associative memory.
- A holographic lateral inhibition network approximates MINERVA 2.
- MINERVA 2 can be implemented as a fully distributed neural model.
- MINERVA 2 can be scaled up arbitrarily assuming an arbitrarily parallel computer.

ARTICLE INFO

Article history:
Received 11 March 2016
Received in revised form
25 September 2016
Available online 17 November 2016

Keywords:
Memory
Cognitive modelling
Computational modelling
MINERVA 2
Vectors
Tensors
Holographic reduced representations
HRRs
Associative memory
Neural networks

ABSTRACT

Computational memory models can explain the behaviour of human memory in diverse experimental paradigms. But research has produced a profusion of competing models, and, as different models focus on different phenomena, there is no best model. However, by examining commonalities among models, we can move towards theoretical unification. Computational memory models can be grouped into composite and separate storage models. We prove that MINERVA 2, a separate storage model of long-term memory, is mathematically equivalent to composite storage memory implemented as a fourth order tensor, and approximately equivalent to a fourth-order tensor compressed into a holographic vector. Building of these demonstrations, we show that MINERVA 2 and related separate storage models can be implemented in neurons. Our work clarifies the relationship between composite and separate storage models of memory, and thereby moves memory models a step closer to theoretical unification.

Computational memory models can explain the behaviour of human memory in diverse experimental paradigms—whether it be recall or recognition, short-term or long-term retention, implicit or explicit learning. But research has produced a profusion of competing models, and, as different models focus on different phenomena, there is no best model. However, computational models of memory share many characteristics indicating wide agreement about the mathematics of how memory works. These shared characteristics can lead us towards developing a unified basis for computational models of human memory.

We argue that the class of memory models that use high dimensional vectors can be understood as belonging to a single mathematical framework. These memory models include *composite vector models* (e.g., Anderson, 1973; Johns, Jones, & Mewhort, 2012; Murdock, 1989), *matrix models* (e.g., Farrell & Lewandowsky, 2002; Humphreys, Pike, Bain, & Tehan, 1989b; Howard & Kahana, 2002; Lewandowsky & Farrell, 2008), *tensor models* (e.g., Humphreys, Bain, & Pike, 1989a; Osth & Dennis, 2015; Smolensky, 1990), *holographic vector models* (e.g., Eich, 1982; Franklin & Mewhort, 2015; Murdock, 1993), and *multi-vector models*such as the MINERVA 2 (Hintzman, 1984) model and variants (e.g., Dougherty, Gettys, & Ogden, 1999; Jamieson, Crump, & Hannah, 2012; Jamieson & Mewhort, 2011; Kwantes, 2005; Thomas, Dougherty, Sprenger, & Harbison, 2008), the Generalized Context Model of categorization (GCM; Nosofsky, 1986, 1991), and the BEAGLE model of distributional semantics (Jones & Mewhort, 2007; see also Jones, Kintsch, & Mewhort, 2006) and variants (Kelly, Kwok, & West, 2015; Rutledge-Taylor, Kelly, West, & Pyke, 2014).

* Correspondence to: 9 Glen Lawrence CRES., Kingston, ON, Canada, K7L 4V1.
E-mail address: matthew.kelly2@carleton.ca (M.A. Kelly).

http://dx.doi.org/10.1016/j.jmp.2016.10.006
0022-2496/© 2016 Elsevier Inc. All rights reserved.

We use MINERVA 2 (Hintzman, 1984) as a starting point for developing our theoretical framework. MINERVA 2 (Hintzman, 1984) is a computational model of long-term memory (both episodic and semantic). We choose MINERVA 2 because it captures a wide variety of human memory phenomena across differing experimental paradigms and as such seems a good candidate for a basis for theoretical unification.

MINERVA 2 has been applied to a number of 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, 2009, 2011), the production effect (Jamieson, Mewhort, & Hockley, 2016) as well as speech perception (Goldinger, 1998), and naming words from print (Kwantes & Mewhort, 1999).

While there are many experimental phenomena that MINERVA 2 cannot account for, our interest is not in MINERVA 2 per se, but in the broader class of models based on MINERVA 2. We refer to this broader class as MINERVA models and use MINERVA 2 to refer specifically to the model proposed by Hintzman (1984). While MINERVA 2 has many limitations, the MINERVA framework as a whole has proved a fruitful research paradigm.

Variations on MINERVA 2 address a broader range of phenomena. MINERVA-AL captures numerous associative learning phenomena from both the animal and human learning literature (Jamieson et al., 2012). Kwantes (2005) uses a MINERVA variant as a model of distributional semantics. Johns, Jamieson, Crump, Jones, and Mewhort (2016) use a MINERVA variant to model the production of natural language syntax given sentence exemplars. MINERVA-DM models judgements of likelihood to account for heuristics and biases in decision-making (Dougherty et al., 1999). The HyGene model (Thomas et al., 2008) extends MINERVA-DM to hypothesis generation and accounts for how errors in hypothesis generation lead to errors in judgement and decision-making.

In this paper, we begin by providing an introduction to MINERVA 2, followed by a comparison of the various memory models that use high-dimensional vectors. These models belong to two broad classes: composite storage and separate storage models. Composite storage models have a clear neural implementation and are invariant in scale with respect to the number of memories stored. Conversely, distributed storage models grow with the number of memories or concepts stored in the model and do not have an established neural interpretation.

To address concerns about the scalability and neural realization of separate storage models, and to move towards a unified theoretical framework for memory models, we present a proof of exact mathematical equivalence between MINERVA 2 (a separate storage model) and an auto-associative fourth-order tensor memory (a composite storage model). We refer to the tensor as a "memory tesseract" as it is a matrix with four equal dimensions. We also prove that MINERVA is approximately equivalent to a holographic approximation to the memory tesseract.

To illustrate the behaviour of the memory models, we present a set of simulations on artificial data. We also compare performance of the holographic approximation to Johns et al.'s (2016) MINERVA model of a sentence production task. We find that the holographic approximation provides a means of implementing MINERVA as a memory system that is invariant in scale with respect to the number of experiences stored, but does so at the cost of increased noise from the compression of the memory traces into a smaller data structure. Also, to be feasibly implemented on very large-scale tasks, the holographic approximation needs to be simulated on a massively parallel computer, such as a neuromorphic computer.

This work clarifies the relationship between MINERVA and other memory models that use high dimensional vectors. This work serves to demonstrate that MINERVA can potentially be used as a basis for unifying high dimensional vector memory modelling, that MINERVA is scalable to arbitrarily long-term learning if implemented on a massively parallel computer, and that MINERVA can be plausibly realized in neurons.

1. How does MINERVA work?

In MINERVA, each individual experience, or episode, is represented by a high dimensional vector, a list of features represented by numerical values. Memory is a table where each row is a vector representing an *episodic trace*, a stored experience. New experiences are stored as new rows in the memory table. New experiences do not need to be novel. A repeated experience is also stored as a new row, separate from previous instances of that experience.

In MINERVA, memory retrieval is not a look-up process, it is a reconstruction process. In the words of Tulving and Watkins (1973, p. 744), a retrieval cue "combines or interacts with the stored information to create the memory of a previously experienced event". When a retrieval cue is presented, each vector in the table "resonates" with the cue in proportion to its similarity to the cue (Hintzman, 1986).

Similarity is computed as a normalized dot-product of the cue's vector with the stored vector. Each stored 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 model uses the echo to respond as appropriate for the given task. Abstract, conceptual, semantic, and categorical information reflect aggregate retrieval over many episodic traces (e.g., Goldinger, 1998; Kwantes, 2005).

Hintzman (1984, p. 96) summarizes MINERVA 2's key assumptions:

(1) only episodic 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, retrieved information reflects their summed output.

According to Hintzman (1990), MINERVA 2 can be understood as an artificial neural network (see Fig. 1). A layer of input nodes represent the cue, a layer of output nodes represent the echo, and between the two is a hidden layer of nodes. In the hidden layer, each node corresponds to an episodic trace. It follows that MINERVA's hidden layer is a localist network: specific nodes represent specific pieces of information.

Modellers using MINERVA are generally agnostic as to how the model is related to the brain. No one claims that for each new experience the brain grows a new neuron that is forever singly dedicated to that particular experience. But no other interpretation of how MINERVA can be implemented in neurons has been previously proposed, leaving open the question of MINERVA's neural plausibility.

2. 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, and we use the term tesseract to refer to a fourth order tensor or "4D matrix".

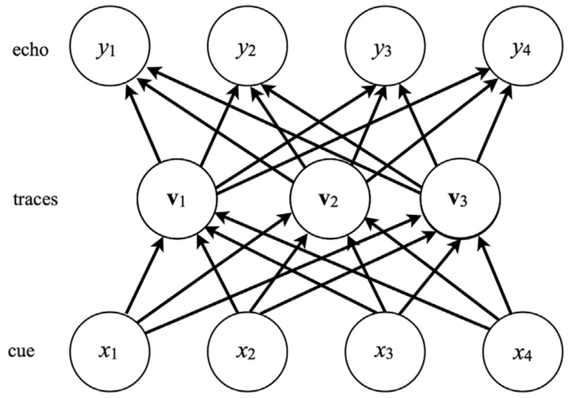


Fig. 1. MINERVA represented as a neural network that operates on four-dimensional vectors ($n = 4$) and has three memory traces stored ($m = 3$). Given a cue $\mathbf{x}$, the network retrieves an echo $\mathbf{y}$, a weighted sum of the traces $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$. The input and output layers can alternatively be represented by a single input/output layer with recurrent connections from the hidden layer, as in Hintzman (1990).

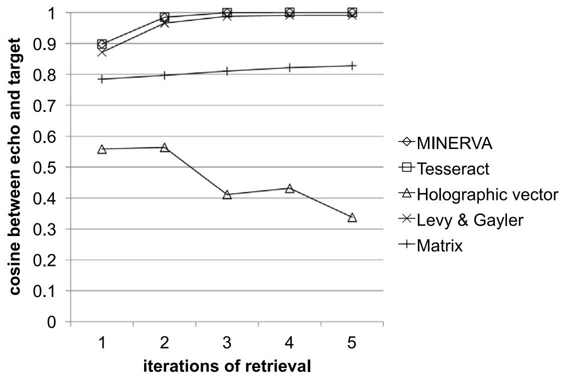


Fig. 2. Iterative retrieval from five memory models: MINERVA 2, the memory tesseract, an auto-associative holographic vector, a variant of Levy and Gayler's (2009) model that approximates MINERVA 2 / the memory tesseract, and an auto-associative matrix memory. The 64-dimensional vectors $\mathbf{a}$ and $\mathbf{b}$ are stored in memory. On the first iteration, $\mathbf{a} + 0.8\mathbf{b}$, normalized to a magnitude of one, is used to retrieve an echo. On successive iterations, the echo from the previous iteration is used to retrieve a new echo. Cosine is measured between each echo and the target, $\mathbf{a}$. Results averaged over 50 runs of each model.

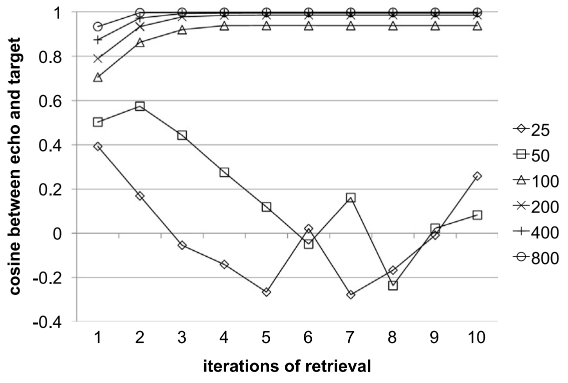


Fig. 3. Iterative retrieval from the holographic approximation to the tesseract. As the number of memory vectors, p, increases from 25 to 800, the model more reliably and more rapidly cleans-up the echo. The 64-dimensional vectors $\mathbf{a}$ and $\mathbf{b}$ are stored in memory, $\mathbf{c}$ adds noise to the probe. On the first iteration, $\mathbf{a}+0.8\mathbf{b}+\mathbf{c}$, normalized to a magnitude of one, is used to retrieve an echo. On successive iterations, the echo from the previous iteration is used to retrieve a new echo. Cosine is measured between each echo and the target, $\mathbf{a}$. Results are from a single run of each model.

For our purposes, vector-based memory models can be divided into five categories: *composite 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); *holographic vector memory*, which compress tensors into a vector; and *multi-vector memory*, where multiple vectors are used to store memories. Composite vector, matrix, tensor, and holographic vector memory are all examples of what Clark and Gronlund (1996) refer to as *composite* or *distributed storage* models because all stored experiences are represented as distributed across a shared set of units. Conversely, multi-vector memory is what Clark and Gronlund refer to as a *separate storage* model because different experiences are represented in different sets of units.

Humphreys et al. (1989b) note that their matrix memory, the multi-vector model MINERVA 2, and the holographic vector model TODAM (Murdock, 1993) all 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 MINERVA, to refer to the retrieved vector in all of the memory models we discuss in this paper. We will use the term *probe* to refer to the vector used as a retrieval cue.

In vector-based memory models, a to-be-remembered item is represented as a vector. These vectors are typically high dimensional with randomly generated values. For example, in the simulations illustrated in Figs 2, 3, and 4, we use vectors of 64 dimensions. The 64 values for each vector are randomly sampled from a normal distribution.

Here we compare *composite vector, matrix, tensor, holographic vector*, and *multi-vector* models of memory (see Table 1).

2.1. Composite vector models

A composite vector memory is simple: it is the sum of the vectors that represent the to-be-remembered items. To store an item in memory, the vector representing that item is weighted by how well it is encoded and then added to the memory vector. The familiarity of a given item is the dot product of that item with the memory vector. Anderson (1973), Johns et al. (2012), and Murdock (1989) use composite vector models to account for performance on recognition tasks.

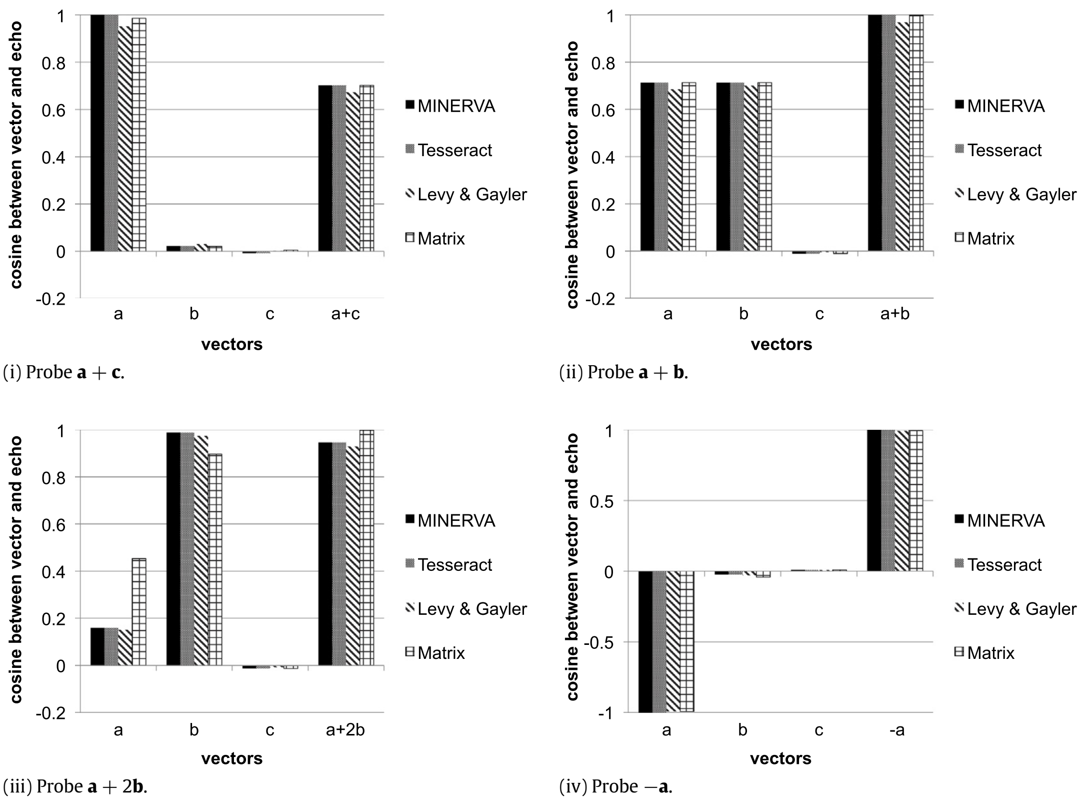
Composite vector memories do not store associations. Instead, items are stored individually. Unlike the other models, a composite vector memory does not retrieve an echo given a probe. Rather, the memory vector itself is a linear combination of item vectors, like the echoes retrieved by the more complicated memory models.

2.2. Matrix and tensor models

A key point of comparison is how vector-based models represent associations (see Table 1). Using the Hebbian learning rule, for a pair of items, each represented as a vector, an association between the pair is represented by the matrix outer-product of the vectors. Smolensky (1990) notes that the tensor product, a generalization of the matrix product, can be used to form associations among an arbitrary number of items (pairs, triples, quadruples, etc.), 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. Items that are not associated can be represented in a composite vector memory, which is a first order tensor. 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 such associations between pairs of vectors. A matrix memory is *auto-associative* if each item is



(i) Probe **a** + **c**.

(ii) Probe **a** + **b**.

(iii) Probe **a** + 2**b**.

(iv) Probe −**a**.

Fig. 4. A comparison of four memory models: MINERVA, the memory tesseract, the holographic approximation to the memory tesseract, and an auto-associative matrix memory. Four different vectors (probes) are presented to all models: (i) **a** + **c**, (ii) **a** + **b**, (iii) **a** + 2**b**, (iv) −**a**. Four corresponding vectors (echoes) are retrieved from each model. Plotted is the cosine between each echo and the vectors **a**, **b**, **c**, and the probe that elicited the echo.

Table 1
Comparison of memory models.

Model by category	Stored associations	Trace activation
Composite vector (1st order tensor)		
Anderson (1973)	*item* (no association)	Dot product
Murdock (1989)	*item*	Dot product
Johns et al. (2012)	*item*	Dot product
Matrix (2nd order tensor)		
Humphreys et al. (1989b)	*item a* × *item b*	Dot product
Howard and Kahana (2002)	*context* × *item*	Dot product
SOB (Farrell & Lewandowsky, 2002)	*item a* × *item a*	Dot product
C-SOB (Lewandowsky & Farrell, 2008)	*item a* × *item a*; *context* × *item*	Dot product
3rd order tensor		
Humphreys et al. (1989a)	*context* × *item a* × *item b*	Dot product
2nd/3rd order tensor		
Osth and Dennis (2015)	*context* × *item*; *context* × *item a* × *item b*	Dot product
Holographic vector (compressed tensor)		
Eich (1982)	*item a* × *item b*	Dot product
TODAM (Murdock, 1993)	*item; item a* × *item b*	Dot product
Franklin and Mewhort (2015)	*item*; *item a* × *item b*	Dot product
Multi-vector (separate storage)		
GCM (Nosofsky, 1986, 1991)	*item* × *category*	Exponential weighted distance
MINERVA	*trace*[a]	Cubed normalized dot product
REM (Shiffrin & Steyvers, 1997)	*item*	Likelihood ratio
BEAGLE (Jones & Mewhort, 2007)	*word a* × *word b* × *word c*…	Vector cosine
DSHM (Rutledge-Taylor et al., 2014)	*item a* × *item b* × *item c*…	Vector cosine
HDM (Kelly et al., 2015)	*item a* × *item b* × *item c*…	Vector cosine

[a] Traces are not associated to each other, but each trace may represent a set of associated items and contexts.

associated with itself (e.g., SOB; Farrell & Lewandowsky, 2002). A matrix memory is *hetero-associative* if each item is associated with a different item (e.g., Humphreys et al., 1989b) or a contextual cue (e.g., Howard & Kahana, 2002).

Associating three vectors results in a "3D matrix", or third order tensor (e.g., Humphreys et al., 1989a; Osth & Dennis, 2015). Osth and Dennis's (2015) model uses two tensors: a matrix for items with a context and a third-order tensor for pairs of items with a context. The two tensors are separate strictly due to mathematical necessity. Osth and Dennis note that they are not committed to singletons and pairs of items having distinct neural substrates or memory systems.

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 are testing for the familiarity of an item, we can use a composite vector memory. If we need a 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 also the architecture becomes increasingly unwieldy as the task becomes more complex.

2.3. Holographic vector models

Given k vectors, each of n dimensions, an association of those vectors can be represented as the tensor product of the vectors, which is a tensor of n^k values (Smolensky, 1990). A tensor is a potentially unwieldy representation if k is large. Furthermore, using tensors necessitates a separate memory store for each value of k because tensors of different orders cannot be added together (e.g., one cannot add together a vector and a matrix).

In a holographic vector memory, the association of a set of vectors is the convolution of those vectors. If circular convolution is used instead of the tensor product, the association of k vectors of n dimensions is itself a vector of n dimensions, irrespective of k (Plate, 1995).

Holographic vectors can represent arbitrarily complex associations of items and context. Using holographic vectors makes it unnecessary to use matrices or higher order tensors to represent associations and allows modellers to adopt a memory architecture that is compact and invariant with respect to the complexity of the associations.

Holographic vectors are part of a family of related computational memory systems called vector-symbolic architectures (Gayler, 2003) that also includes MAP codes (Gayler, 2003), square matrix representations (Kelly, 2010) and binary spatter codes (Kanerva, 1996; see Kelly, Blostein, & Mewhort, 2013, for a review). Vector-symbolic architectures are so named because they provide a means of representing symbolic expressions of arbitrary complexity using vectors. Vector-symbolic architectures address the question of how the complex data structures necessary for reasoning and language can be realized in neural networks (for discussion, see Gayler, 2003; Plate, 1995). All vector-symbolic architectures can be understood as compressing the tensor product of a set of vectors into a single vector.

As holographic vectors are the most common vector-symbolic architecture in the literature, we restrict our discussion to holographic vectors, and in particular, holographic reduced representations (Plate, 1995), which use circular convolution to encode associations. As vector-symbolic architectures all have similar properties, our discussion pertains to all memory models that use vector-symbolic architectures.

Cognitive models of memory that use holographic vectors can explain and predict a variety of human memory phenomena (e.g., Eich, 1982; Franklin & Mewhort, 2015; Murdock, 1993). The majority of models that use holographic vectors are purely cognitive and, as such, do not concern themselves with the question of how the models could be implemented in the brain. However, the Neural Engineering Framework (Eliasmith, 2013) uses holographic vectors as the representation scheme for their neurocognitive model of the brain Eliasmith et al. (2012). Thagard and Stewart (2011) suggest that holographic vectors are the neural underpinning of creativity. Holographic vectors present a plausible means by which the brain could recursively combine patterns of neural activations representing simple concepts to generate new patterns of activation that represent novel, complicated concepts.

However, circular convolution is a *lossy compression* of the tensor-product (Plate, 1995; for discussion see Kelly et al., 2013). Information is lost in the act of compressing a tensor of size n^k down to a vector of size n. Information loss is the only[1] reason one might prefer matrix or tensor memories to a holographic memory. However, combining holographic vectors with MINERVA creates a system that can store arbitrarily complex associations of items and contexts, and retrieve them with fidelity (Jamieson & Mewhort, 2011).

2.4. MINERVA versus vector and matrix models

Raising similarity to an exponent of 3 sets the MINERVA models apart from the vector and matrix models, as Hintzman explains (1990, p. 116):

> This model escapes being just a less efficient version of the vector model by using nonlinearity. In particular, the activation of each hidden unit is a positively accelerated function of its match to the input vector, limiting the number of units that will respond significantly to any input, and thereby reducing noise.

By weighting each episodic trace by the cube of its similarity, the traces that are most similar to the cue contribute much more to the echo than traces that have only partial similarity to the cue, or traces that have tiny, incidental similarity to the cue.

Raising the similarity to any exponent larger than 1 introduces non-linearity. However, when using even-numbered exponents, the sign of the similarity is lost. MINERVA 2 uses a normalized dot product to measure similarity, which ranges from $+1$ to -1. To preserve the sign of the similarity, MINERVA 2 uses an exponent 3 rather than 2. Larger odd-numbered exponents (5, 7, 9, ...) also preserve the sign of the similarity and can be used to further reduce the amount of noise in the echo (e.g., Johns et al., 2016).

Non-linearity also allows MINERVA to “clean-up” the echo by iteratively using the echo as a cue to produce a new echo (Hintzman, 1986). With each pass through MINERVA, the contributions of the most similar traces grow. This process can be repeated until the echo reaches a steady state where it no longer changes, at which point the echo will closely resemble the trace most similar to the initial cue (see Fig. 2).

The clean-up process serves as a possible explanation for why we are faster to remember some things than others: echoes formed from frequently occurring and distinctive episodic traces reach a steady state more quickly. The number of iterations until the echo reaches a steady state is used to produce response latency predictions in the SOB (Farrell & Lewandowsky, 2002) and C-SOB (Lewandowsky & Farrell, 2008) models of memory.

Linear systems require an external clean-up memory. Unless some form of non-linearity is introduced, the echo from a holographic vector memory (e.g., TODAM; Murdock, 1993) or, to a lesser extent, a matrix memory (e.g., Humphreys et al., 1989b) is noisy (see figure) and requires an external memory for items in order to identify which item the echo most resembles (Murdock, 1993). Raising similarity to an exponent is not the only way to introduce non-linearity. For example, restricting the output of a model to binary or bipolar values can introduce the necessary non-linearity to iteratively clean up the echo (e.g., Farrell & Lewandowsky, 2002).

MINERVA needs to be nonlinear, as it cannot rely on an external clean-up memory. MINERVA is a model of both episodic and semantic memory and thus it would “violate the spirit of MINERVA 2” (Hintzman, 1986, p. 416) to have an external store of items (i.e., a separate semantic memory) to clean up the echo.

[1] Another difference is that convolution is commutative whereas the matrix product and tensor product are non-commutative. A non-commutative variant of convolution can be used if required (Plate, 1995, p. 12).

2.5. MINERVA and other multi-vector models

In *multi-vector* memory models, memories are stored as a collection of vectors in a table. Multi-vector models include MINERVA models, the Retrieving Effectively from Memory model (REM; Shiffrin & Steyvers, 1997) the Generalized Context Model of categorization (GCM; Nosofsky, 1986, 1991), the BEAGLE model of distributional semantics (Jones & Mewhort, 2007), and variants of BEAGLE, such as the DSHM (Rutledge-Taylor et al., 2014) and HDM (Kelly et al., 2015) models of memory.

MINERVA and GCM store each *experience* as a separate vector, such that the number of vectors grows with each additional memory trace stored. DSHM, HDM, and BEAGLE store each distinct *concept* as a separate vector, such that the number of vectors grows only with the addition of new concepts. REM takes a hybrid approach and stores vectors for both individual experiences and individual concepts. REM makes the argument that concepts are frequently revisited experiences, such that vectors in memory exist on a continuum from experience to concept. Conversely, MINERVA holds that concepts or categories are emergent from aggregate retrieval across experiences and so do not require a distinct storage mechanism.

In both MINERVA and GCM, when a probe is presented to memory, each vector in memory is activated according to its similarity to the probe raised to some power. The GCM calculates similarity as e to the power of the negative Euclidean distance, which is 1 when the vectors are identical and asymptotically approaches 0 as the vectors move further apart. As similarity in the GCM is always positive, the GCM can raise similarity to even-numbered powers without losing the sign of the similarity. In the GCM, similarity is raised to the power of c, where c is a sensitivity parameter set by the modeller. Conversely, in MINERVA 2, the exponent is always 3, though some variants of MINERVA dynamically vary the exponent (e.g., Mewhort & Johns, 2005) or use a larger exponent to minimize noise (e.g., Johns et al., 2016).

The GCM differs from MINERVA in that the GCM is a model of categorization judgements whereas MINERVA is a more general model of memory. MINERVA retrieves from memory an echo, a vector representing an inexact recollection. Conversely, the GCM retrieves from memory the amount of evidence for each possible categorization, computed as the sum of the activations of the exemplars of each category.

However, the GCM can be used to model recognition judgements by computing a familiarity score for a probe as a sum of the activations of each category (Nosofsky, 1991). Likewise, MINERVA 2 can model category learning (e.g., Hintzman, 1986) and can be made to imitate the GCM. MINERVA 2 can store exemplar-category pairs by concatenating a vector representing an exemplar with a vector representing the corresponding category. Given an exemplar as a probe, MINERVA 2 retrieves an echo where the latter half of the echo will be a weighted sum of category vectors. The cosine between the latter half of the echo and each category vector will be the amount of evidence for each categorization, as in the GCM.

The REM model (Shiffrin & Steyvers, 1997) differs from MINERVA in that the activation of a vector is computed as a likelihood ratio: the probability of the observed similarity given the probe and trace are a match is divided by the probability of the similarity given the probe and trace are a mismatch. Computing activation using Bayesian probability allows REM to model a number of list strength effects (Shiffrin & Steyvers, 1997) that are problematic for MINERA and composite memory models (see Shiffrin, Ratcliff, & Clark, 1990 for a discussion), though more recent work has suggested alternate approaches to accounting for list strength effects in composite (Johns et al., 2012) and MINERVA models (Jamieson et al., 2016).

BEAGLE (Jones & Mewhort, 2007) and variants (Kelly et al., 2015; Rutledge-Taylor et al., 2014) are collections of holographic vectors. BEAGLE and variants differ from MINERVA in that each item is represented in memory by a single vector that is sum of all experiences with that item. The memory vectors stored in BEAGLE, DSHM and HDM can be thought of as analogous to the echoes retrieved from memory by MINERVA. A MINERVA model that uses the MINERVA 2 architecture and the representation assumptions of BEAGLE can generate echoes that closely approximate the memory vectors in BEAGLE (Kelly, 2016, p. 61, p. 203).

Thus, the GCM model of categorization, the BEAGLE model of distributional semantics, and the DSHM and HDM memory models based on BEAGLE, can all be instantiated within the MINERVA architecture and are compatible with MINERVA theory. However, in practical terms, the BEAGLE model cannot be implemented as a MINERVA model as described, as it would require storing one vector in the memory table for each word in the corpus. The size of a corpus may range from a few million words, to 100 million words (e.g., the British National Corpus), to 1.9 billion words (e.g., Global Web-Based English). Thus, a strict MINERVA instantiation of BEAGLE is not feasibly computable.

3. Scalability of MINERVA

Applying MINERVA to large scale 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 computational efficiency, namely, that repetition of an item produces multiple traces of that item.

This might seem like an assumption that can be abandoned without affecting the model. If an item is repeated *exactly*, adding another row to MINERVA's table to represent the second occurrence of the item is mathematically equivalent to storing only one trace for that item and doubling that trace's weighting. Instead of adding additional traces, exact repetition of an item can be simulated by increasing the weight of the item's trace. While this approach suffices for modelling many experimental tasks, for real world tasks, *exact* repetition is an unrealistic assumption. In the words of Heraclitus, you cannot step twice into the same river.

For example, Kwantes and Mewhort's (1999) MINERVA model of sounding-out written words has one vector per word. Each vector stores a prototypical pronunciation of the word, rather than any particular experience of having heard the word pronounced, which would vary depending on context and speaker. BEAGLE (Jones & Mewhort, 2007) and Kwantes' (2005) MINERVA model, both models of learning the meaning of words, also have one vector per word, such that each experience of a word's use in the language is summed into a single vector.

For the sake of computational feasibility, as MINERVA models are scaled up, the models shift from storing experiences to concepts. Thus, there are two kinds of MINERVA model: models that store episodes (i.e., episodic memory), and models that store generic knowledge (i.e., semantic memory). Modelling episodic and semantic memory using different assumptions violates the spirit of MINERVA 2, as the model is intended as an account of semantic and episodic memory as a single, integrated system (Hintzman, 1986).

4. MINERVA as a fourth order tensor

In what follows, we prove equivalence between the MINERVA 2 model and an auto-associative fourth order tensor memory. To do so, we first prove that a variant of MINERVA that raises similarity to an exponent of 1 is equivalent to an auto-associative second order tensor (i.e., a matrix) memory. Then, we prove that a MINERVA that uses an 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.

4.1. MINERVA with an exponent of 1

Consider a variant on the MINERVA model that uses dot product (denoted by $\bullet$)[2] to measure similarity and weights each trace by its similarity raised to the exponent of 1. Each trace in memory is represented by a vector $\mathbf{v}_i$ where $i = 1 \ldots m$ and m is the number of traces in memory. When the model is presented with a cue $\mathbf{x}$, the echo $\mathbf{y}$ is:

$$\mathbf{y} = (\mathbf{x} \bullet \mathbf{v}_1)\mathbf{v}_1 + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)\mathbf{v}_m.$$

This is equivalent to an auto-associative matrix memory (e.g., Farrell & Lewandowsky, 2002).

In an auto-associative matrix memory, each trace is represented by a vector $\mathbf{v}_i$. 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, $\mathbf{v}_i\mathbf{v}_i^{\mathrm{T}}$, then taking the sum of all the outer-product matrices to create the memory matrix, $\mathbf{M}$:

$$\mathbf{M} = \mathbf{v}_1\mathbf{v}_1^{\mathrm{T}} + \cdots + \mathbf{v}_m\mathbf{v}_m^{\mathrm{T}}.$$

The echo, $\mathbf{y}$, is the inner-product of the cue and the matrix:

$$\mathbf{y} = \mathbf{M}\mathbf{x}$$
$$\mathbf{y} = (\mathbf{v}_1\mathbf{v}_1^{\mathrm{T}} + \cdots + \mathbf{v}_m\mathbf{v}_m^{\mathrm{T}})\mathbf{x}$$
$$\mathbf{y} = \mathbf{v}_1\mathbf{v}_1^{\mathrm{T}}\mathbf{x} + \cdots + \mathbf{v}_m\mathbf{v}_m^{\mathrm{T}}\mathbf{x}.$$

Because $\mathbf{v}_i^{\mathrm{T}}\mathbf{x}$ is the dot-product of $\mathbf{v}_i$ and $\mathbf{x}$:

$$\mathbf{y} = (\mathbf{x} \bullet \mathbf{v}_1)\mathbf{v}_1 + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)\mathbf{v}_m$$

which is identical to the echo from a MINERVA with an exponent of 1.

We note that on a recognition task, the behaviour of a MINERVA with an exponent of 1 is equivalent to the behaviour of a composite vector memory, depending on how familiarity is computed. A composite vector memory $\mathbf{m}$ is a sum of item vectors:

$$\mathbf{m} = \mathbf{v}_1 + \cdots + \mathbf{v}_m.$$

Whether or not an item $\mathbf{x}$ is recognized by $\mathbf{m}$ is a function of the familiarity, f, computed as the dot product of $\mathbf{x}$ with $\mathbf{m}$,

$$f = \mathbf{x} \bullet \mathbf{m} = (\mathbf{x} \bullet \mathbf{v}_1) + \cdots + (\mathbf{x} \bullet \mathbf{v}_m).$$

In MINERVA 2, familiarity can be computed in one of two ways. These two methods for computing familiarity are common across vector-based models of memory and are referred to as *local match* and *global match* (Kahana, Rizzuto, & Schneider, 2005). When using a *global match*, familiarity is computed as the similarity between the probe and all of memory. When using *local match*, familiarity is computed as the similarity between the probe and the echo.

When using *global match*, Hintzman (1986) computes familiarity as the sum of similarities between the probe and each trace in memory. In MINERVA 2, the familiarity f would be as follows:

$$f = (\mathbf{x} \bullet \mathbf{v}_1)^3 + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)^3.$$

However, when using a MINERVA with an exponent of 1, the familiarity would equal to the familiarity computed by a composite vector model:

$$f = (\mathbf{x} \bullet \mathbf{v}_1) + \cdots + (\mathbf{x} \bullet \mathbf{v}_m).$$

Thus, on recognition tasks, when computing familiarity as in Hintzman (1986), a MINERVA with an exponent of 1 is equivalent to a composite vector model.

Alternatively, Hintzman (1988, p. 546) suggests that familiarity can be computed using the *local match*method as the similarity between the echo and the probe. Local match is used in some MINERVA 2 models (e.g., Jamieson & Mewhort, 2009). Because computing familiarity as the similarity between the probe and the echo allows for easier comparison to other memory models, it is the approach we adopt in our simulations (see Figs. 2, 3, and 4).

For either MINERVA with an exponent of 1 or, equivalently, an auto-associative matrix memory, given an echo,

$$\mathbf{y} = (\mathbf{x} \bullet \mathbf{v}_1)\mathbf{v}_1 + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)\mathbf{v}_m$$

we can compute the familiarity of $\mathbf{x}$ as $\mathbf{x} \bullet \mathbf{y}$,

$$f = \mathbf{x} \bullet \mathbf{y} = \mathbf{x} \bullet ((\mathbf{x} \bullet \mathbf{v}_1)\mathbf{v}_1 + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)\mathbf{v}_m)$$
$$f = \mathbf{x} \bullet (\mathbf{x} \bullet \mathbf{v}_1)\mathbf{v}_1 + \cdots + \mathbf{x} \bullet (\mathbf{x} \bullet \mathbf{v}_m)\mathbf{v}_m$$
$$f = (\mathbf{x} \bullet \mathbf{v}_1)(\mathbf{x} \bullet \mathbf{v}_1) + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)(\mathbf{x} \bullet \mathbf{v}_m)$$
$$f = (\mathbf{x} \bullet \mathbf{v}_1)^2 + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)^2$$

which differs from the familiarity in a composite vector model as each term is squared. For MINERVA 2, i.e., a MINERVA with an exponent of 3, computing familiarity in this manner results in each term being raised to the power of four:

$$f = (\mathbf{x} \bullet \mathbf{v}_1)^4 + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)^4.$$

In general, it seems that the difference between the *local match* and *global match* methods for computing familiarity is that if the global match method for the model in question computes familiarity as,

$$f = (\mathbf{x} \bullet \mathbf{v}_1)^n + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)^n$$

then the local match model computes familiarity as

$$f = (\mathbf{x} \bullet \mathbf{v}_1)^{n+1} + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)^{n+1}.$$

Thus, these two methods for computing familiarity, the *global* match between the probe and memory or the *local* match between probe and the echo, are not as different as they might seem from their initial description. However, the local match can be calculated as a cosine of the probe and the echo (i.e., a normalized dot product), as in Jamieson and Mewhort (2009). Normalizing either the global or local match may produce substantively different behavioural predictions than using the un-normalized match values.

4.2. MINERVA with an exponent of 2

Consider a MINERVA that raises similarity to the exponent of 2:

$$\mathbf{y} = (\mathbf{x} \bullet \mathbf{v}_1)^2\mathbf{v}_1 + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)^2\mathbf{v}_m.$$

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 $\mathbf{v}_i \otimes \mathbf{v}_i \otimes \mathbf{v}_i$, which is a third order tensor. The memory tensor $\mathbf{M}$ is the sum of the third order tensor outer-products of each trace:

$$\mathbf{M} = \mathbf{v}_1 \otimes \mathbf{v}_1 \otimes \mathbf{v}_1 + \cdots + \mathbf{v}_m \otimes \mathbf{v}_m \otimes \mathbf{v}_m.$$

The echo, $\mathbf{y}$, can be computed from the cue, $\mathbf{x}$, by taking the inner product *twice*:

$$\mathbf{y} = (\mathbf{M}\,\mathbf{x})\mathbf{x}.$$

If each $\mathbf{v}_i$ is a vector of n dimensions, then $\mathbf{M}$ is an $n \times n \times n$ tensor. $\mathbf{M}$ can be thought of as n matrices of $n \times n$ dimensions. When we compute the inner-product of $\mathbf{M}$ with the cue $\mathbf{x}$, we compute the inner product of $\mathbf{x}$ with each of those n matrices. This results in n

[2] To simplify the proof, we assume that MINERVA uses the dot product to measure similarity. Using the dot product is equivalent to using the cosine (i.e., a normalized dot product) to measure similarity if all vectors, the cue and traces, are normalized to a magnitude of one. This assumption merely moves normalization from being part of running the model to part of generating the vectors before running the model, and so has no effect on the model's behaviour.

vectors that can be rearranged into a new $n \times n$ matrix. The second inner product with $\mathbf{x}$ then produces a vector, the echo $\mathbf{y}$.

To illustrate, let us break $\mathbf{M}$ into its components:

$$\mathbf{y} = (\mathbf{M}\,\mathbf{x})\mathbf{x}$$
$$\mathbf{y} = ((\mathbf{v}_1 \otimes \mathbf{v}_1 \otimes \mathbf{v}_1 + \cdots + \mathbf{v}_m \otimes \mathbf{v}_m \otimes \mathbf{v}_m)\mathbf{x})\mathbf{x}$$
$$\mathbf{y} = (\mathbf{v}_1 \otimes \mathbf{v}_1 \otimes \mathbf{v}_1\mathbf{x} + \cdots + \mathbf{v}_m \otimes \mathbf{v}_m \otimes \mathbf{v}_m\mathbf{x})\mathbf{x}.$$

The tensor product $\mathbf{v}_i \otimes \mathbf{v}_i \otimes \mathbf{v}_i$ can be understood as n matrices, where each matrix is the outer-product $\mathbf{v}_i\mathbf{v}_i^{\mathrm{T}}$ weighted by a different element j of $\mathbf{v}_i$, for all $j = 1 \ldots n$.

$$\mathbf{v}_i \otimes \mathbf{v}_i \otimes \mathbf{v}_i = \{v_{i1}\mathbf{v}_i\mathbf{v}_i^{\mathrm{T}}, \ldots, v_{in}\mathbf{v}_i\mathbf{v}_i^{\mathrm{T}}\}.$$

Taking the inner-product of the cue $\mathbf{x}$ with $\mathbf{v}_i \otimes \mathbf{v}_i \otimes \mathbf{v}_i$, we get n vectors, each weighted by the dot-product of $\mathbf{x}$ with $\mathbf{v}_i$:

$$\mathbf{v}_i \otimes \mathbf{v}_i \otimes \mathbf{v}_i\mathbf{x} = \{v_{i1}\mathbf{v}_i\mathbf{v}_i^{\mathrm{T}}\mathbf{x}, \ldots, v_{in}\mathbf{v}_i\mathbf{v}_i^{\mathrm{T}}\mathbf{x}\}$$
$$\mathbf{v}_i \otimes \mathbf{v}_i \otimes \mathbf{v}_i\mathbf{x} = \{v_{i1}(\mathbf{x} \bullet \mathbf{v}_i)\mathbf{v}_i, \ldots, v_{in}(\mathbf{x} \bullet \mathbf{v}_i)\mathbf{v}_i\}.$$

If we factor out the dot-product of $\mathbf{x}$ and $\mathbf{v}_i$, the result is n vectors, or rather, the outer-product matrix of $\mathbf{v}_i\mathbf{v}_i^{\mathrm{T}}$:

$$\mathbf{v}_i \otimes \mathbf{v}_i \otimes \mathbf{v}_i\mathbf{x} = (\mathbf{x} \bullet \mathbf{v}_i)\{v_{i1}\mathbf{v}_i, \ldots, v_{in}\mathbf{v}_i\}$$
$$\mathbf{v}_i \otimes \mathbf{v}_i \otimes \mathbf{v}_i\mathbf{x} = (\mathbf{x} \bullet \mathbf{v}_i)\mathbf{v}_i\mathbf{v}_i^{\mathrm{T}}.$$

Thus, when we take the outer-product of $\mathbf{x}$ with $\mathbf{M}$, the result is a sum of m matrices $\mathbf{v}_i\mathbf{v}_i^{\mathrm{T}}$, each matrix weighted by the dot-product of $\mathbf{x}$ with $\mathbf{v}_i$.

$$\mathbf{y} = (\mathbf{v}_1 \otimes \mathbf{v}_1 \otimes \mathbf{v}_1\mathbf{x} + \cdots + \mathbf{v}_m \otimes \mathbf{v}_m \otimes \mathbf{v}_m\mathbf{x})\mathbf{x}$$
$$\mathbf{y} = ((\mathbf{x} \bullet \mathbf{v}_1)\mathbf{v}_1\mathbf{v}_1^{\mathrm{T}} + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)\mathbf{v}_m\mathbf{v}_m^{\mathrm{T}})\mathbf{x}.$$

By then taking the second inner-product with $\mathbf{x}$, we reduce each matrix to a vector weighted by the squared similarity to $\mathbf{x}$, producing an echo like MINERVA with an exponent of 2:

$$\mathbf{y} = (\mathbf{x} \bullet \mathbf{v}_1)\mathbf{v}_1\mathbf{v}_1^{\mathrm{T}}\mathbf{x} + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)\mathbf{v}_m\mathbf{v}_m^{\mathrm{T}}\mathbf{x}$$
$$\mathbf{y} = (\mathbf{x} \bullet \mathbf{v}_1)(\mathbf{x} \bullet \mathbf{v}_1)\mathbf{v}_1 + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)(\mathbf{x} \bullet \mathbf{v}_m)\mathbf{v}_m$$
$$\mathbf{y} = (\mathbf{x} \bullet \mathbf{v}_1)^2\mathbf{v}_1 + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)^2\mathbf{v}_m.$$

Thus, an auto-associative third order tensor memory or "memory cube" is equivalent to a MINERVA with an exponent of 2. Humphreys et al. (1989a) use a hetero-associative third-order tensor memory that associates two different items with a context: $item1 \times item2 \times context$. By contrast, the MINERVA third-order tensor memory associates an item with itself twice: $item1 \times item1 \times item1$.

4.3. MINERVA with an exponent of 3 (i.e., MINERVA 2)

Given a pair of vectors that both have a magnitude of one, the dot product of those vectors is in the range of +1 to −1. The dot product is +1 if the vectors are identical, 0 if the vectors are orthogonal, and −1 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:

$$\mathbf{M} = \mathbf{v}_1 \otimes \mathbf{v}_1 \otimes \mathbf{v}_1 \otimes \mathbf{v}_1 + \cdots + \mathbf{v}_m \otimes \mathbf{v}_m \otimes \mathbf{v}_m \otimes \mathbf{v}_m.$$

Given a cue $\mathbf{x}$, an echo $\mathbf{y}$ is computed by taking the inner product three times:

$$\mathbf{y} = ((\mathbf{M}\mathbf{x})\mathbf{x})\mathbf{x}$$
$$\mathbf{y} = (((\mathbf{v}_1 \otimes \mathbf{v}_1 \otimes \mathbf{v}_1 \otimes \mathbf{v}_1 + \cdots + \mathbf{v}_m \otimes \mathbf{v}_m \otimes \mathbf{v}_m \otimes \mathbf{v}_m)\mathbf{x})\mathbf{x})\mathbf{x}$$
$$\mathbf{y} = (((\mathbf{x} \bullet \mathbf{v}_1)\mathbf{v}_1 \otimes \mathbf{v}_1 \otimes \mathbf{v}_1 + \cdots + (\mathbf{x} \bullet \mathbf{v}_1)\mathbf{v}_m \otimes \mathbf{v}_m \otimes \mathbf{v}_m)\mathbf{x})\mathbf{x}$$
$$\mathbf{y} = ((\mathbf{x} \bullet \mathbf{v}_1)^2\mathbf{v}_1\mathbf{v}_1^{\mathrm{T}} + \cdots + (\mathbf{x}\mathbf{v}_m)^2\mathbf{v}_m\mathbf{v}_m^{\mathrm{T}})\mathbf{x}$$
$$\mathbf{y} = (\mathbf{x} \bullet \mathbf{v}_1)^3\mathbf{v}_1 + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)^3\mathbf{v}_m.$$

5. Implications of the memory tesseract

MINERVA is equivalent to a distributed memory system implemented as an auto-associative fourth order tensor, or *memory tesseract*. Re-interpreting MINERVA as a memory tesseract changes our understanding of the MINERVA model, and in turn, changes our understanding of what manipulations of the model are permissible.

If MINERVA is understood as a fourth-order tensor, cubing the similarity is not a parameter but a structural feature of the model. Understood as such, it is less permissible to change the value of the exponent. If we understand the GCM (Nosofsky, 1986, 1991) as a MINERVA model restricted to category judgements, then the GCM's c exponent in its activation function must also be understood as a structural feature of the GCM rather than a value that can be changed at whim.

Clark and Gronlund (1996) propose a MINERVA model where information about items and associations is cubed, but information about environmental context is not cubed. Implementing Clark and Gronlund's MINERVA model as a tensor would require two separate data structures: a fourth-order tensor for items and associations, and a matrix memory for contexts. Implemented in this manner, the Clark and Gronlund MINERVA would resemble (Lewandowsky & Farrell, 2008) C-SOB model, which also has an auto-associative memory for items and a separate matrix memory for context information. Note, though, that the C-SOB auto-associative memory for items is a matrix whereas in the Clark and Gronlund MINERVA it would be a fourth-order tensor.

The Iterative Resonance Model (IRM; Mewhort & Johns, 2005) is a variant of MINERVA that models retrieval from memory as an iterative process. On the first iteration, an echo is constructed as a sum of traces, each trace weighted by their similarity to the probe raised to an exponent of one. On each successive iteration, the same probe is used, but the exponent is incremented by 0.5. By increasing the exponent with each iteration, the echo is increasingly dominated by the trace most similar to the probe. If MINERVA is understood as a tensor, increasing the exponent at each time-step during retrieval corresponds to increasing the order of the tensor, which implies changing the structure of the model during retrieval. As the exponent grows with each time-step, the tensor grows exponentially in size. Thus, the memory tesseract cannot implement the IRM as described.

We suggest that IRM could be implemented using the memory tesseract, but it would need to use a different iterative retrieval mechanism. As shown in Fig. 2, iterative retrieval can be implemented in MINERVA with a fixed exponent of 3. On the first iteration, the probe is used to retrieve an echo. On the iterations that follow, the echo from the previous iteration is used as a new probe to retrieve a new echo. Across iterations, the echo increasingly approximates the vector in memory most similar to the original probe. This form of iterative retrieval has been used to predict reaction times in other memory models (e.g., Farrell & Lewandowsky, 2002; Lewandowsky & Farrell, 2008).

Just as re-interpreting MINERVA as a memory tesseract suggests that certain operations, such as changing the exponent, are impermissible, it also suggests other operations are permissible. The equivalence shows that MINERVA is not different in kind from vector, matrix, tensor, and holographic vector models. Thus, some of the techniques used in these other models to capture memory phenomena may be successfully applied to MINERVA. For example, MINERVA is not typically used to simulate primacy and recency effects. However, some of the other vector-based models capture one or both of these effects (e.g., Farrell & Lewandowsky, 2002; Franklin & Mewhort, 2015; Murdock, 1993). These models do so by applying weights to the information stored in memory.

To give a concrete example, we can adopt TODAM's (Murdock, 1993) forgetting parameter α and apply it to MINERVA. For each time t when memory is updated with a new trace $\mathbf{v}_t$, we weight the memory tensor $\mathbf{M}$ by α, where $0 < \alpha < 1$:

$$\mathbf{M}_t = \alpha \mathbf{M}_{t-1} + \mathbf{v}_t.$$

If the forgetting parameter α is less than one, newer memories are weighted more strongly during retrieval, allowing the model to simulate a recency effect. While this technique can be used regardless of whether MINERVA is implemented as a tensor or a table, the equivalence of the table to a tensor shows that these kinds of operations are permissible by MINERVA's theory.

6. Is the memory tesseract practical?

Unfortunately, fourth order tensors are very large. For most applications of MINERVA to experimental tasks, the dimensionality n of a vector will be larger than the number of memories m stored in the model. MINERVA, as standardly 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, such as modelling the lifetime learning of an agent (e.g., Jones & Mewhort, 2007), where $m > n^3$, the fourth order tensor is more space efficient. However, when implemented on a serial computer, the fourth order tensor computes storage in memory n^3 times more slowly than a memory table. This is because storage in the tensor is a matter of computing a new fourth order tensor to represent the trace and adding it to the memory tensor. The storage computation for a tensor memory is prohibitively slow for large data sets processed on a serial computer.

If negative similarities have no meaning in the model, one could use a third-order tensor instead, equivalent to MINERVA with an exponent of 2. In many MINERVA models, negative similarities occur only when two vectors have approximately zero similarity, but by chance this approximate value is negative. When squaring these small, negative similarities, the sign is lost, but the sign of these approximately zero values is unimportant to the model. Such models could be reimplemented with an exponent of 2 or, equivalently, as a third-order tensor. For large-scale applications of these models, where $m > n^2$, the tensor implementation would be more space efficient, though storage is n^2 times slower on a serial computer. However, for some MINERVA models, negative similarities play an important role in the behaviour of the model, such as MINERVA-AL (Jamieson et al., 2012), which stores the differences between expected and observed values for the purposes of modelling learning.

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.

7. Using holographic vectors rather than tensors

In a holographic vector memory, 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. Holographic vectors use lossy compression. With each pass through the holographic vector memory, the amount of loss, or noise, increases (see the holographic vector model in Fig. 2).

Yet Levy and Gayler (2009) have shown that it is possible to use a holographic vector system as a clean-up memory (see *Levy and Gayler model* in Fig. 2). Levy and Gayler use a lateral inhibition network implemented as a fully-distributed vector architecture. Gayler and Levy (2009) use their architecture to model analogical mapping, but we note that it can also be used as a memory system. Indeed, their network is approximately equivalent to MINERVA 2.

To store a trace $\mathbf{v}_i$ in Levy and Gayler's model, the trace is associated with itself twice, then each trace is added to the memory vector $\mathbf{m}$:

$$\mathbf{m} = \mathbf{v}_1 * \mathbf{v}_1 * \mathbf{v}_1 + \cdots + \mathbf{v}_m * \mathbf{v}_m * \mathbf{v}_m$$

where $*$ is a binding operation used to form associations. Levy and Gayler (2009) use MAP codes (Gayler, 2003), which use element-wise multiplication as a binding operation. We choose to use holographic reduced representations instead. In holographic reduced representations (Plate, 1995), binding uses circular convolution and unbinding uses circular correlation. Given a cue, $\mathbf{x}$, we can unbind, denoted by #, to recover an echo:

$$\mathbf{y} = \mathbf{x}\#(\mathbf{x}\#\mathbf{m})$$
$$\mathbf{y} = \mathbf{x}\#(\mathbf{x}\#(\mathbf{v}_1 * \mathbf{v}_1 * \mathbf{v}_1 + \cdots + \mathbf{v}_m * \mathbf{v}_m * \mathbf{v}_m)).$$

Unbinding is such that given a bound pair, $\mathbf{v}_i * \mathbf{v}_j$,

$$\mathbf{x}\#\mathbf{v}_i * \mathbf{v}_j = (\mathbf{x} \bullet \mathbf{v}_i)\mathbf{v}_j + \boldsymbol{\eta}$$

where $\boldsymbol{\eta}$ is a vector of noise with a mean of zero and a normal distribution (Plate, 1995). Thus,

$$\mathbf{y} = \mathbf{x}\#(\mathbf{x}\#(\mathbf{v}_1 * \mathbf{v}_1 * \mathbf{v}_1 + \cdots + \mathbf{v}_m * \mathbf{v}_m * \mathbf{v}_m))$$
$$\mathbf{y} = \mathbf{x}\#((\mathbf{x} \bullet \mathbf{v}_1)\mathbf{v}_1 * \mathbf{v}_1 + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)\mathbf{v}_m * \mathbf{v}_m + \boldsymbol{\eta})$$
$$\mathbf{y} = (\mathbf{x} \bullet \mathbf{v}_1)^2\mathbf{v}_1 + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)^2\mathbf{v}_m + \boldsymbol{\eta}.$$

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, $\mathbf{m}$, and unbind the echo, $\mathbf{y}$, as follows:

$$\mathbf{m} = \mathbf{v}_1 * \mathbf{v}_1 * \mathbf{v}_1 * \mathbf{v}_1 + \cdots + \mathbf{v}_m * \mathbf{v}_m * \mathbf{v}_m * \mathbf{v}_m$$
$$\mathbf{y} = \mathbf{x}\#(\mathbf{x}\#(\mathbf{x}\#\mathbf{m}))$$
$$\mathbf{y} = \mathbf{x}\#(\mathbf{x}\#(\mathbf{x}\#(\mathbf{v}_1 * \mathbf{v}_1 * \mathbf{v}_1 * \mathbf{v}_1 + \cdots + \mathbf{v}_m * \mathbf{v}_m * \mathbf{v}_m * \mathbf{v}_m)))$$
$$\mathbf{y} = (\mathbf{x} \bullet \mathbf{v}_1)^3\mathbf{v}_1 + \cdots + (\mathbf{x} \bullet \mathbf{v}_m)^3\mathbf{v}_m + \boldsymbol{\eta}.$$

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. A random permutation is a random re-ordering of the elements in a vector, like shuffling the cards in a deck. A permutation can be represented by a *permutation matrix*, a square matrix with an entry of 1 exactly once in each row and column and 0 s in all other entries. When a vector $\mathbf{v}$ is multiplied by a permutation matrix $\mathbf{P}$, the matrix product $\mathbf{P}\,\mathbf{v}$ is the permuted vector.

Given three random permutation matrices $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$, we can permute the vectors as follows:

$$\mathbf{m} = (\mathbf{P}_1\mathbf{v}_1) * (\mathbf{P}_2\mathbf{v}_1) * (\mathbf{P}_3\mathbf{v}_1) * \mathbf{v}_1 \\ + \cdots + (\mathbf{P}_1\mathbf{v}_m) * (\mathbf{P}_2\mathbf{v}_m) * (\mathbf{P}_3\mathbf{v}_m) * \mathbf{v}_m.$$

To recover an echo, we use the permutations again:

$$\mathbf{y} = ((\mathbf{P}_1\mathbf{x}) * (\mathbf{P}_2\mathbf{x}) * (\mathbf{P}_3\mathbf{x}))\#\mathbf{m}.$$

To eliminate the noise, Levy and Gayler use hundreds of such memory vectors in parallel, each of which uses its own pair, or in our variant, triple, of permutations. Let p be the number of memory vectors being used. The echo $\mathbf{y}$ is the sum of echoes for each p:

$$\mathbf{y} = \mathbf{y}_1 + \cdots + \mathbf{y}_p$$

where for each $\mathbf{y}_j, j = 1 \ldots p$,

$$\mathbf{y}_j = ((\mathbf{P}_{j,1}\mathbf{x}) * (\mathbf{P}_{j,2}\mathbf{x}) * (\mathbf{P}_{j,3}\mathbf{x}))\#\mathbf{m}_j$$

such that:

$$\mathbf{y} = ((\mathbf{P}_{1,1}\mathbf{x}) * (\mathbf{P}_{1,2}\mathbf{x}) * (\mathbf{P}_{1,3}\mathbf{x}))\#\mathbf{m}_1 \\ + \cdots + ((\mathbf{P}_{p,1}\mathbf{x}) * (\mathbf{P}_{p,2}\mathbf{x}) * (\mathbf{P}_{p,3}\mathbf{x}))\#\mathbf{m}_p.$$

Because each echo $\mathbf{y}_j$ is produced using a different triple of permutations, each echo's noise term will be different. Because the noise in each echo is different, a different part of the signal is preserved in each echo. By taking the sum of all these echoes, we average across them to get a close approximation to the true signal.

When averaging across a set of measurements with uncorrelated error, as the number of measurements approaches infinity, the statistical error on the mean approaches zero and the mean approaches the true value. In this case, as the number of memory vectors p increases to infinity, the mean echo approaches the exact value of the echo from a standard MINERVA 2. The parameter p thus gives the modeller the ability to control the reliability of the model as an approximation to MINERVA 2.

With large enough p, we minimize the noise sufficiently that we can clean up the echo by iterating (see Fig. 3). We divide the sum of all the echoes, $\mathbf{y}$, by its magnitude to normalize, and then use it as the new cue. By varying the number of memory vectors operating in parallel p, we can manipulate how quickly and how reliably the echo is cleaned up by iterating.

The number of iterations to clean up the echo is a measure of the time the model takes to perform a memory retrieval. For this reason, the number of iterations is an ideal candidate as a predictor for human reaction time. *Iterations-to-clean-up-the-echo* is used to predict reaction times in the SOB (Farrell & Lewandowsky, 2002) and C-SOB (Lewandowsky & Farrell, 2008) models of serial recall.

The number of iterations to clean up the echo in MINERVA 2 may be too few to provide sufficient granularity to map well onto human reaction times. As Hintzman (1986, p. 416), notes, for a typical experimental task, after three to four iterations, the echo will reach a steady state where it is identical to a trace in memory (see Fig. 2). If we instead use Levy and Gayler's model, by adjusting p we can vary the number of iterations that the model takes to clean up the echo (see Fig. 3). Using smaller p, the model takes, typically, longer to converge to a steady state than an iterated MINERVA 2, such that the iterations map onto smaller units of human reaction time. Thus, the lateral inhibition network, when coupled with a suitable decision mechanism to end retrieval, may provide more fine-grained reaction time predictions than an iterated MINERVA 2.

The lateral inhibition network is also more tractable than a fourth order tensor for large n (i.e., more space efficient for $p < n^3$ and more time efficient for $p \log n < n^3$). The lateral inhibition network is also more space efficient than the table implementation for $p < m$ and more time efficient on retrieval for $p \log n < m$ but is $p \log n$ times slower at storing information in memory on a serial computer.

8. Neural implementation of the holographic model

As we have shown, MINERVA 2 is approximately equivalent to a modified version of the Levy and Gayler (2009) model. The naïve neural implementation of MINERVA commits us to the claim that a new neuron is grown for each new episodic trace (Fig. 1). However, MINERVA 2, implemented as the Levy and Gayler model, can be constructed as a biologically plausible neural model where the number of neurons is constant with respect to the number of episodic traces. That such a model can be constructed serves as a proof that MINERVA 2 is not committed to the existence of the infamous grandmother cell—a neuron singly dedicated to representing a particular experience of one's grandmother.

Eliasmith's (2013) Neural Engineering Framework[3] provides a system for implementing linear algebra computations (e.g., circular convolution, vector addition, and permutation) in networks of biologically realistic model neurons. The Neural Engineering Framework is the basis for what is currently the largest functional brain model, the Semantic Pointer Architecture Unified Network (SPAUN; Eliasmith et al., 2012). By functional, we mean that SPAUN is the largest brain model that can perform cognitive tasks. For example, SPAUN can complete Raven's progressive matrices, a simple pattern recognition task used as a test of IQ (Rasmussen & Eliasmith, 2011). Knowledge in SPAUN is represented using holographic vectors.

Implementing a holographic vector model requires three linear algebra operations: circular convolution, permutation, and vector addition. Permutation and vector addition are trivial to implement in neural networks. Using the Neural Engineering Framework, circular convolution can be computed in a two-layer feed-forward neural network constructed from biologically realistic model neurons (Eliasmith, 2013, p. 128).

Plate (2000) notes that in a neural network, implementing circular convolution and its approximate inverse, circular correlation, requires intricate and precise patterns of neural connectivity. It seems difficult to explain how such patterns could arise naturally in the brain. However, circular convolution is just one, highly ordered, way of compressing tensors into vectors. Random compression schemes are an effective substitute for circular convolution (Plate, 1994, p. 154). Plate (2000) finds that a network of randomly connected sigma–pi neurons can bind two vectors in an association and that a second network can be trained to unbind that association. Thus, though the brain may not implement circular convolution per se, the brain could discover functionally equivalent compressions of the tensor product for the purposes of forming associations among memories.

9. Simulations

To illustrate the behaviour of the models, we perform a series of simulations. In these simulations, we see that MINERVA 2 and the memory tesseract behave exactly identically, and the Levy and Gayler model (2009) closely approximates the behaviour of MINERVA 2 and the tesseract. These simulations also illustrate how these models differ from the holographic vector and matrix memory models.

All simulations for Figs. 2, 3, and 4 were conducted using 64 dimensional vectors with values sampled randomly from a normal distribution with a mean of zero and a variance of 1/64. For the purpose of comparison, all models used the same three vectors: **a**, **b**, and **c**. All models had vectors **a** and **b** stored in memory, whereas **c** was used as a source of noise.

The MINERVA model used in all simulations is identical to Hintzman's (1984) MINERVA 2 except that it operates on vectors of normally distributed rather than binary values and correspondingly uses the cosine to measure similarity. The memory tesseract is an auto-associative fourth order tensor. Referred to as Levy and Gayler in Figs. 2 and 4, the holographic approximation to the memory tesseract is a variant of Levy and Gayler's (2009) lateral inhibition network which, as described in the previous section, differs from Levy and Gayler's original model in that it uses holographic reduced representations (Plate, 1995) and convolves items stored in memory with themselves three times (rather than twice) in order to imitate the behaviour of

[3] Models that use the Neural Engineering Framework can be constructed using the NENGO software package. For download instructions, see http://www.nengo.ca/download.

MINERVA 2 and the memory tesseract. In Figs. 2 and 4, the model uses $p = 400$ memory vectors in parallel and averages across the 400 retrieved vectors to produce an echo. The matrix memory in Figs. 2 and 4 is an auto-associative second order tensor. The holographic vector model in Fig. 2 is a compressed version of the matrix memory that uses convolution rather than the outer product to form associations.

Fig. 2 illustrates iterative retrieval from five memory models: MINERVA 2, the memory tesseract (behaviour is identical to MINERVA 2), the holographic approximation to the memory tesseract, and a matrix memory. On the first iteration, the probe $\mathbf{a} + 0.8\mathbf{b}$, normalized to a magnitude of one, is used to retrieve an echo. On successive iterations, the echo from the previous iteration is used to retrieve a new echo. Cosine is measured between each echo and the target, $\mathbf{a}$. Results are averaged over 50 runs of each model. For each run of the five models, a different $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ are randomly generated. All models use the same 50 $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$.

In Fig. 2, we can see that MINERVA 2, the memory tesseract, and the Levy and Gayler model all rapidly clean up the probe $\mathbf{a} + 0.8\mathbf{b}$, nearly perfectly reproducing $\mathbf{a}$ by iteration 3 or 4. The matrix memory grows to resemble a slightly more across successive iterations. Conversely, the echo of the holographic vector memory degrades across iterations. Fig. 2 illustrates that MINERVA 2 and equivalent (or approximately equivalent) models, do not require an external clean-up memory, whereas a holographic vector memory does.

Fig. 3 illustrates iterative retrieval from the holographic approximation to the tesseract with a varying number of memory vectors p. As the number of memory vectors, p, increases from 25 to 800, the model more reliably and more rapidly cleans-up the echo. On the first iteration, the probe $\mathbf{a} + 0.8\mathbf{b} + \mathbf{c}$, normalized to a magnitude of one, is used to retrieve an echo. On successive iterations, the echo from the previous iteration is used to retrieve a new echo. Cosine is measured between each echo and the target, $\mathbf{a}$. Results are from a single run of each model, with all models run on the same set of vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$. We chose not to average across runs because the average behaviour of the model is not representative of the actual behaviour of the model, particularly for small values of p (see $p = 25$ and $p = 50$ in Fig. 3).

In Fig. 4, we present the results of a simulation of four memory models alongside each other: MINERVA, the memory tesseract, the holographic approximation to the memory tesseract, and a matrix memory. In Fig. 4, all models were presented with four probes: $\mathbf{a} + \mathbf{c}$, $\mathbf{a} + \mathbf{b}$, $\mathbf{a} + 2\mathbf{b}$, and $-\mathbf{a}$, with results shown in Fig. 4 (i), (ii), (iii), and (iv), respectively. Given a probe, each model retrieved an echo. Fig. 4 shows the cosine similarity that each echo has with the vectors $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, and the probe that elicited the echo. Results are averaged across 10 different randomly generated vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$.

In Fig. 4, we see that the behaviour of MINERVA and the memory tesseract model is exactly identical, as expected given the previously presented proof of mathematical equivalence. The behaviour of all four models is approximately the same for probes (i), (ii), and (iv). In Fig. 4 (i), given the probe $\mathbf{a} + \mathbf{c}$, the models return an echo that is nearly identical to a and does not contain the unstudied item $\mathbf{c}$. In Fig. 4 (ii), given a probe $\mathbf{a} + \mathbf{b}$, that is an equal mix of two studied items, all models return an echo nearly identical to $\mathbf{a} + \mathbf{b}$. The echoes are caught between the two attractors, $\mathbf{a}$ and $\mathbf{b}$. In Fig. 4 (iv), we see that given the probe $-\mathbf{a}$ all models produce an echo identical to $-\mathbf{a}$. This demonstrates that all of the models preserve the sign of the cosine, as we would expect given that the models use an odd-numbered exponent (an exponent of 1 for the matrix memory and an exponent of 3 for the other models).

Fig. 4 (iii) demonstrates the difference in behaviour between the (linear) matrix memory and the other (non-linear) models. Given the probe $\mathbf{a} + 2\mathbf{b}$, the matrix memory produces an echo nearly identical to the probe (cosine $= 1.00$). Conversely, MINERVA 2 and the memory tesseract produce an echo which is nearly identical to $\mathbf{b}$ (cosine $= 0.99$) and only weakly similar to $\mathbf{a}$ (cosine $= 0.16$). The behaviour of the Levy and Gayler model closely approximates MINERVA 2 and the memory tesseract, but with added noise.

10. Approximating the exemplar production model

To provide a larger-scale demonstration of the memory tesseract, we replicate a simulation result from Johns et al. (2016). Johns et al. demonstrate that a MINERVA model trained on exemplar sentences can infer the syntactic structure of novel sentences. For example, given the unordered set of words {*be, deep, in, she, seemed, thought, to*}, the model can infer from the example sentences stored in memory that the words can be re-ordered into the grammatical sentence "*she seemed to be deep in thought*", even though the model has never seen that sentence before. Johns et al. name their model the Exemplar Production Model (EPM).

EPM represents each sentence as a pair of vectors. One vector represents the sentences as an unordered set of words. The other vector represents the words as an ordered sequence. EPM stores pairs of these n dimensional vectors as concatenated vectors of $2n$ dimensions. Given a probe that represents an unordered set of words, EPM can retrieve an echo, the latter half of which indicates the order of the words according to the model's experience. EPM is trained on a study set of 1000–125 000 sentences and then tested on 200 novel sentences. Johns et al. use sentences of three to seven words.

Johns et al. represent each word as a vector of 1024 dimensions. An *unordered* representation of a sentence is a sum of the vectors that represent the words in the sentence. An *ordered* representation of a sentence is a holographic vector, a sum of convolutions and permutations of vectors representing words and positions in the sentence. Each trace stored in EPM is a 2048 dimensional concatenation of an ordered and unordered representation. In the largest case, when EPM is trained on 125 000 sentences, EPM's memory is a $125\,000 \times 2048$ dimensional matrix.

Using the holographic approximation to the memory tesseract, we replicate Johns et al.'s simulation of syntactic inference from 125 000 seven word sentences. To create the study set, we sampled 125 000 seven word sentences from a corpus of novels selected from Project Gutenberg.[4] The test sets from Johns et al. can be downloaded from Johns' website.[5] Due to differences in how contractions are represented in the corpus versus the test set (e.g., is *haven't* two words or one?), we removed 40 sentences with contractions from the test set. We used the remaining 160 seven word sentences as the test set.

For seven word sentences and a study set of 125 000 sentences, Johns et al. report that EPM finds the correct ordering of words for 52% of the sentences in the test set. Notably, EPM uses an exponent of 9 instead of 3. Running EPM on our 125 000 word study set and the 160 sentence test set, we get 65% correct using 1024 dimensional vectors for words (2048 for traces) and an exponent of 9 (the same parameters used by Johns et al.).

We find that after 10 iterations to clean up the echo, performance decreases to 53% correct, which suggests that with this data set and parameters, iterating is introducing noise into the echo rather than cleaning up. If we decrease the exponent to 3, EPM gets only 16% correct, which decreases to 5% correct after 10 iterations. This suggests that MINERVA 2's canonical exponent of 3 is insufficient when realistically large quantities of data are stored in memory.

4 http://www.gutenberg.org.

5 http://btjohns.com/experience_sents.zip.

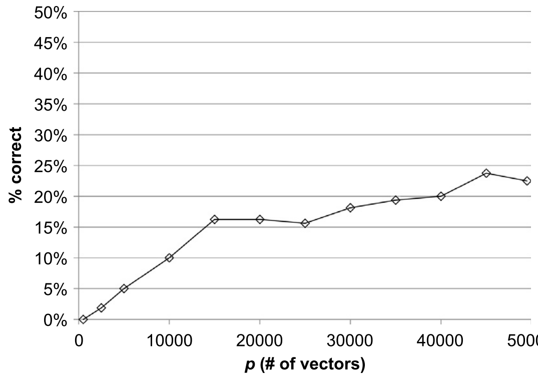


Fig. 5. Percentage of test sentences ordered correctly by the holographic approximation to EPM as a function of the number of memory vectors, *p*.

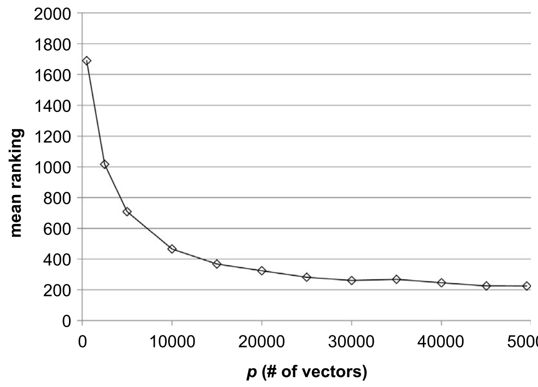


Fig. 6. Mean ranking of the correct ordering of words in test sentences. Performance of the holographic approximation to EPM as a function of the number of memory vectors, *p*.

EPM is equivalent to a tenth order tensor, a data structure of 2048^{10} values. A holographic vector implementation approximates that tensor, compressing 2048^{10} values into a vector of 2048 values. We use a variant of EPM with an exponent of 5 instead of 9 to reduce the amount of compression. With an exponent of 5, the holographic vectors compress 2048^5 into 2048. With an exponent of 5, EPM gets 42% correct. Simulating the holographic approximation to EPM with up to $p = 50\,000$ vectors yields, at best, 24% correct on the test items (see Fig. 5).

The model is judged to have selected the correct ordering when the vector representing the correct ordering is more similar to the echo than the vectors representing any other possible ordering. For a seven-word sentence, there are $7! = 5040$ different orderings of the words in the sentence. These 5040 orderings are ranked from most similar to the echo to least. If the correct ordering is assigned a rank of one, then the model is judged to be correct on that test item. The mean and median ranks provide a more sensitive measure of the model's performance. With a power of 5, EPM's echoes have a mean rank of 35.4 and a median rank of 2. For the holographic approximation to EPM, at 50 000 vectors, the mean rank is 224.5 and the median rank is 9. Performance of the holographic model seems to approach an asymptote as the number of memory vectors p increases, such that there is diminishing improvement in performance as more vectors are added (see Fig. 6).

This simulation demonstrates that larger scale MINERVA models can be re-implemented as the holographic approximation. However, representing memory traces as distributed across a set of memory vectors introduces additional noise, particularly when compressing and storing the memory traces across a set of memory vectors fewer than the number of memory traces. For applications where fine discrimination is critical (such as discriminating among 5040 highly similar alternatives), this added noise might not be inconsequential to performance of the model.

Additionally, the holographic approximation is impractical to implement on a serial computer. On a serial computer, storing a memory trace across 50 000 memory vectors takes 50 000 times longer than storing a memory trace in MINERVA. MINERVA's implementation as a table is preferable assuming a serial computer with unlimited storage.

This simulation does, however, demonstrate that MINERVA can be re-implemented as a massively parallel memory system with a fixed amount of storage. As the brain is also a massively parallel computer with limited storage, this demonstration lends support to MINERVA's validity as a theory of the memory algorithms implemented in the brain.

The holographic approximation may also be a practical implementation of MINERVA for large-scale modelling applications that use neuromorphic computers. Neuromorphic computers include the SpiNNaker architecture (Furber, Galluppi, Temple, & Plana, 2014), which has been used for simulating neural models of learning and memory (Knight, Voelker, Mundy, Eliasmith, & Furber, 2016).

11. Discussion

This work contributes to developing a unified mathematical and computational basis for modelling memory. At the heart of memory theory is the Hebbian learning rule: neurons that fire together wire together. If the activation patterns of two groups of connected neurons can be represented by a pair of vectors, then the outer-product of those vectors describes the connections that will form between the neurons, represented as a matrix of connection weights. Abstracting away from the neural details, Smolensky (1990) notes that, in general, an association of k items can be represented as a tensor of order k. We see this observation reflected in the memory models in the literature. Composite vector memories are tensors of order one that store single items. Matrix memories are tensors of order two that store pairs of representations. '3D' matrix memories are tensors of order three that store triples of representations. Collectively, we can refer to these memory models as tensor memories.

Holographic vectors (Plate, 1995) and the related vector-symbolic architectures (see Kelly et al., 2013 for review) are tensors compressed into vectors. A holographic vector of fixed dimensionality can encode associations of k representations for arbitrary k. Holographic vectors thus provide a computationally efficient basis for modelling memory for arbitrarily complex associations. Collectively, we can refer to vector-symbolic/holographic vector models and tensor models as composite memory models. All share the same Hebbian learning underpinning, although abstracted away from the details of biological neurons.

In the literature, multi-vector (or separate storage) memory models have been regarded as distinct from composite memory models (e.g., Clark & Gronlund, 1996). By proving that MINERVA is equivalent to a fourth order tensor memory, we show that these two classes of models are not different in kind. Additionally, we show that large-scale 'semantic' memory models such as BEAGLE and smaller-scale 'episodic' memory models such as MINERVA are also not different in kind. Thus, the models discussed in this paper can be understood as part of a single mathematical framework. We would expect that, for the most part, advances made for any particular one of these memory models could be translated into an advance for any other of these memory models. Thus, the profusion of memory models in the literature may not pose as much difficulty for developing a unified model of human memory as it might initially seem.

An important question in the development of a unified model of human memory will be settling the issue of representation. We have noted that different (tensor) models of memory adopt different structures (different orders of tensor) to model different tasks. Holographic vectors provide a way of building a memory model with an efficiently computable structure that is invariant with respect to changes in the task demands. But this does not solve the problem that the representation scheme used changes from task to task. Memory models do not explain how these representations arise, which allows the modeller to assume whatever representations are most convenient for modelling the given experimental task.

To some extent the *ad hoc* approach to modelling representation is justifiable, as humans encode information in different tasks differently so as to best meet the demands of the task (Hintzman, 2016). But to advance closer to a complete theoretical understanding of human memory, memory models need to be integrated with a theory of representation, and hence, perception. Some work has been done towards this goal – the SPAUN brain model (Eliasmith et al., 2012) incorporates both perception and memory – but much more work needs to be done in integrating perception and memory if we are to achieve a full understanding of either.

12. Conclusion

We demonstrate that the influential MINERVA 2 (Hintzman, 1984) model is mathematically equivalent to an auto-associative 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 among MINERVA and third order, second order (i.e., matrix), first order (i.e., composite vector), and compressed tensor (i.e., holographic vector) memory models, allowing us to move towards a unified basis for models of memory.
(2) As MINERVA can be implemented as a fully distributed memory system that is invariant in scale with respect to the number of experiences stored, MINERVA can, in principle, be scaled to arbitrarily long-term learning. However, simulating MINERVA re-implemented as a tensor or holographic vector is most feasible on a massively parallel computer.
(3) A nave neural interpretation of MINERVA might suggest that a new neuron is grown for each new experience, corresponding to the addition of 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. MINERVA can be implemented as a neural model using Eliasmith's framework.

Acknowledgments

This research is supported by an Ontario Graduate Scholarship awarded to the first author and grants AP 318 and 315656 from the Natural Sciences and Engineering Research Council of Canada to the second and third authors, respectively. The MATLAB and FORTRAN code used for the simulations in this paper can be found at: https://github.com/MatthewAKelly/Memory-Tesseract.

References

Anderson, J. A. (1973). A theory for the recognition of items from short memorized lists. *Psychological Review*, *80*, 417–438. http://dx.doi.org/10.1037/h0035486.
Clark, S. E., & Gronlund, S. D. (1996). Global matching models of recognition memory: How the models match the data. *Psychonomic Bulletin & Review*, *3*, 37–60. http://dx.doi.org/10.3758/BF03210740.
Dougherty, M. R. P., Gettys, C. F., & Ogden, E. E. (1999). MINERVA-DM: A memory processes model for judgments of likelihood. *Psychological Review*, *106*, 180–209. http://dx.doi.org/10.1037/0033-295X.106.1.180.
Eich, J. M. (1982). A composite holographic associative recall model. *Psychological Review*, *89*, 627–661. http://dx.doi.org/10.1037/0096-3445.119.2.145.
Eliasmith, C. (2013). *How to build a brain: A neural architecture for biological cognition*. New York, NY: Oxford University Press.
Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., & Rasmussen, D. (2012). A large-scale model of the functioning brain. *Science*, *338*, 1202–1205. http://dx.doi.org/10.1126/science.1225266.
Farrell, S., & Lewandowsky, S. (2002). An endogenous distributed model of ordering in serial recall. *Psychonomic Bulletin & Review*, *9*, 59–79. http://dx.doi.org/10.3758/BF03196257.
Franklin, D. R. J., & Mewhort, D. J. K. (2015). Memory as a hologram: An analysis of learning and recall. *Canadian Journal of Experimental Psychology*, *69*, 115–135. http://dx.doi.org/10.1037/cep0000035.
Furber, S. B., Galluppi, F., Temple, S., & Plana, L. A. (2014). The SpiNNaker project. *Proceedings of the IEEE*, *102*, 652–665. http://dx.doi.org/10.1109/JPROC.2014.2304638.
Gayler, R. W. (2003). Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience. In P. Slezak (Ed.), *Proceedings of the joint international conference on cognitive science* (pp. 133–138). Sydney, Australia: University of New South Wales.
Gayler, R. W., & Levy, S. D. (2009). A distributed basis for analogical mapping. In B. Kokinov, K. Holyoak, & D. Gentner (Eds.), *New frontiers in analogy research; Proceedings of the Second International Analogy Conference—Analogy 09* (pp. 165–174). Sofia, Bulgaria: New Bulgarian University Press.
Goldinger, S. D. (1998). Echoes of echoes? An episodic theory of lexical access. *Psychological Review*, *105*, 251–279. http://dx.doi.org/10.1037/0033-295X.105.2.251.
Hintzman, D. L. (1984). MINERVA 2: A simulation model of human memory. *Behavior Research Methods, Instruments, and Computers*, *16*, 96–101. http://dx.doi.org/10.3758/BF03202365.
Hintzman, D. L. (1986). "Schema abstraction" in multiple-trace memory models. *Psychological Review*, *93*, 411–428. http://dx.doi.org/10.1037/0033-295X.95.4.528.
Hintzman, D. L. (1988). Judgments of frequency and recognition memory in a multiple-trace memory model. *Psychological Review*, *95*, 528–551. http://dx.doi.org/10.1037/0033-295X.95.4.528.
Hintzman, D. L. (1990). Human learning and memory: Connections and dissociations. *Annual Review of Psychology*, *41*, 109–139. http://dx.doi.org/10.1146/annurev.ps.41.020190.000545.
Hintzman, D. L. (2016). Is memory organized by temporal contiguity? *Memory & Cognition*, *44*, 365–375. http://dx.doi.org/10.3758/s13421-015-0573-8.
Howard, M. W., & Kahana, M. J. (2002). A distributed representation of temporal context. *Journal of Mathematical Psychology*, *46*, 269–299. http://dx.doi.org/10.1006/jmps.2001.1388.
Humphreys, M. S., Bain, J. D., & Pike, R. (1989a). Different ways to cue a coherent memory system: A theory for episodic, semantic, and procedural tasks. *Psychological Review*, *96*, 208–233. http://dx.doi.org/10.1037/0033-295X.96.2.208.
Humphreys, M. S., Pike, R., Bain, J. D., & Tehan, G. (1989b). Global matching: A comparison of the SAM, Minerva II, Matrix, and TODAM models. *Journal of Mathematical Psychology*, *33*, 36–67. http://dx.doi.org/10.1016/0022-2496(89)90003-5.
Jamieson, R. K., Crump, M. J. C., & Hannah, S. D. (2012). An instance theory of associative learning. *Learning & Behavior*, *40*, 61–82. http://dx.doi.org/10.3758/s13420-011-0046-2.
Jamieson, R. K., & Mewhort, D. J. K. (2009). Applying an exemplar model to the artificial-grammar task: Inferring grammaticality from similarity. *The Quarterly Journal of Experimental Psychology*, *62*, 550–575. http://dx.doi.org/10.1080/17470210802055749.
Jamieson, R. K., & Mewhort, D. J. K. (2011). Grammaticality is inferred from global similarity: A reply to kinder (2010). *The Quarterly Journal of Experimental Psychology*, *64*, 209–216. http://dx.doi.org/10.1080/17470218.2010.537932.
Jamieson, R. K., Mewhort, D. J. K., & Hockley, W. E. (2016). A computational account of the production effect: Still playing twenty questions with nature. *Canadian Journal of Experimental Psychology*, *70*, 154–164. http://dx.doi.org/10.1037/cep0000081.
Johns, B. T., Jamieson, R. K., Crump, M. J. C., Jones, M. N., & Mewhort, D. J. K. (2016). The combinatorial power of experience. In A. Papafragou, D. Grodner, D. Mirman, & J. C. Trueswell (Eds.), *Proceedings of the 38th annual meeting of the cognitive science society* (pp. 1325–1330). Austin, TX: Cognitive Science Society.
Johns, B. T., Jones, M. N., & Mewhort, D. J. K. (2012). A synchronization account of false recognition. *Cognitive Psychology*, *65*, 486–518. http://dx.doi.org/10.1016/j.cogpsych.2012.07.002.
Jones, M. N., Kintsch, W., & Mewhort, D. J. K. (2006). High-dimensional semantic space accounts of priming. *Journal of Memory and Language*, *55*, 534–552. http://dx.doi.org/10.1016/j.jml.2006.07.003.

Jones, M. N., & Mewhort, D. J. K. (2007). Representing word meaning and order information in a composite holographic lexicon. *Psychological Review*, *114*, 1–37. http://dx.doi.org/10.1037/0033-295X.114.1.1.

Kahana, M. J., Rizzuto, D. S., & Schneider, A. R. (2005). Theoretical correlations and measured correlations: Relating recognition and recall in four distributed memory models. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *31*, 933–953. http://dx.doi.org/10.1037/0278-7393.31.5.933.

Kanerva, P. (1996). Binary spatter-coding of ordered k-tuples. In C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, B. Sendhoff (Eds.), *Artificial Neural Networks – International Conference on Artificial Neural Networks 96 Proceedings* (pp. 869–873). Bochum, Germany.

Kelly, M. A. (2010). *Advancing the theory and utility of holographic reduced representations* (Master's thesis). Canada: Queen's University, ProQuest Dissertations and Theses. http://search.proquest.com/docview/853293422.

Kelly, M. A. (2016). *The memory tesseract: Developing a unified framework for modelling memory and cognition* (Doctoral dissertation). Canada: Carleton University, CURVE Theses and Dissertations Collection.

Kelly, M. A., Blostein, D., & Mewhort, D. J. K. (2013). Encoding structure in holographic reduced representations. *Canadian Journal of Experimental Psychology*, *67*, 79–93. http://dx.doi.org/10.1037/a0030301.

Kelly, M. A., Kwok, K., & West, R. L. (2015). Holographic declarative memory and the fan effect: A test case for a new memory model for ACT-R. In N. A. Taatgen, M. K. van Vugt, J. P. Borst, & K. Mehlhorn (Eds.), *Proceedings of the 13th international conference on cognitive modeling* (pp. 148–153). Groningen, the Netherlands: University of Groningen. http://www.iccm2015.org/proceedings/papers/0036/.

Knight, J., Voelker, A. R., Mundy, A., Eliasmith, C., & Furber, S. (2016). Efficient SpiNNaker simulation of a heteroassociative memory using the Neural Engineering Framework. In *The 2016 International Joint Conference on Neural Networks (IJCNN)* (pp. 5210–5217). Vancouver, BC, Canada: IEEE. http://dx.doi.org/10.1109/IJCNN.2016.7727888.

Kwantes, P. J. (2005). Using context to build semantics. *Psychonomic Bulletin & Review*, *12*, 703–710. http://dx.doi.org/10.3758/BF03196761.

Kwantes, P. J., & Mewhort, D. J. K. (1999). Modeling lexical decision and word naming as a retrieval process. *Canadian Journal of Experimental Psychology*, *53*, 306–315. http://dx.doi.org/10.1037/h0087318.

Levy, S.D., & Gayler, R.W. (2009). "Lateral inhibition" in a fully distributed connectionist architecture. In A. Howes, D. Peebles, R. Cooper (Eds.), *Proceedings of the 9th International Conference on Cognitive Modeling - ICCM 2009* (pp. 318–323). Manchester, UK.

Lewandowsky, S., & Farrell, S. (2008). Short-term memory: New data and a model. In B. H. Ross (Ed.), *The psychology of learning and motivation*: *Vol. 49* (pp. 1–48). London, UK: Elsevier. http://dx.doi.org/10.1016/S0079-7421(08)00001-7.

Mewhort, D. J. K., & Johns, E. E. (2005). Sharpening the echo: An iterative-resonance model for short-term recognition memory. *Memory*, *13*, 300–307. http://dx.doi.org/10.1080/09658210344000242.

Murdock, B. B. (1989). Learning in a distributed memory model. In C. Izawa (Ed.), *Current issues in cognitive processes: The Tulane Flowerree symposium on cognition*. Hillsdale, NJ: Erlbaum.

Murdock, B. B. (1993). TODAM2: a model for the storage and retrieval of item, associative and serial-order information. *Psychological Review*, *100*, 183–203. http://dx.doi.org/10.1037/0033-295X.100.2.183.

Nosofsky, R. M. (1986). Attention, similarity and the identification-categorization relationship. *Journal of Experimental psychology: General*, *115*, 39–57. http://dx.doi.org/10.1037/0096-3445.115.1.39.

Nosofsky, R. M. (1991). Tests of an exemplar model for relating perceptual classification and recognition memory. *Journal of Experimental Psychology: Human Perception and Performance*, *17*, 3–27. http://dx.doi.org/10.1037/0096-1523.17.1.3.

Osth, A. F., & Dennis, S. (2015). Sources of interference in item and associative recognition memory. *Psychological Review*, *122*, 260–311. http://dx.doi.org/10.1037/a0038692.

Plate, T. A. (1994). *Distributed representations and nested compositional structure* (Doctoral dissertation). Canada: University of Toronto, ProQuest Dissertations and Theses. http://search.proquest.com/docview/304158858.

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

Plate, T. A. (2000). Randomly connected sigma-pi neurons can form associative memories. *Network: Computation in Neural Systems*, *11*, 321–332. http://dx.doi.org/10.1088/0954-898X_11_4_305.

Rasmussen, D., & Eliasmith, C. (2011). A neural model of rule generation in inductive reasoning. *Topics in Cognitive Science*, *3*, 140–153. http://dx.doi.org/10.1111/j.1756-8765.2010.01127.x.

Rutledge-Taylor, M. F., Kelly, M. A., West, R. L., & Pyke, A. A. (2014). Dynamically structured holographic memory. *Biologically Inspired Cognitive Architectures*, *9*, 9–32. http://dx.doi.org/10.1016/j.bica.2014.06.001.

Shiffrin, R. M., Ratcliff, R., & Clark, S. E. (1990). List-strength effect: II. Theoretical mechanisms. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *16*, 179–195. http://dx.doi.org/10.1037/0278-7393.16.2.179.

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

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, *46*, 159–216. http://dx.doi.org/10.1016/0004-3702(90)90007-M.

Thagard, P., & Stewart, T. C. (2011). The Aha! experience: Creativity through emergent binding in neural networks. *Cognitive Science*, *35*, 1–33. http://dx.doi.org/10.1111/j.1551-6709.2010.01142.x.

Thomas, R. P., Dougherty, M. R., Sprenger, A. M., & Harbison, J. I. (2008). Diagnostic hypothesis generation and human judgment. *Psychological Review*, *115*, 155–185. http://dx.doi.org/10.1037/0033-295X.115.1.155.

Tulving, E., & Watkins, M. J. (1973). Continuity between recall and recognition. *American Journal of Psychology*, *86*, 739–748. http://dx.doi.org/10.2307/1422081.