Analogical mapping and inference with binary spatter codes and sparse distributed memory

Blerim Emruli, Ross W. Gayler and Fredrik Sandin

Abstract— **Analogy-making is a key function of human cognition. Therefore, the development of computational models of analogy that automatically learn from examples can lead to significant advances in cognitive systems. Analogies require complex, relational representations of learned structures, which is challenging for both symbolic and neurally inspired models. Vector symbolic architectures (VSAs) are a class of connectionist models for the representation and manipulation of compositional structures, which can be used to model analogy. We study a novel VSA network for the analogical mapping of compositional structures, which integrates an associative memory known as sparse distributed memory (SDM). The SDM enables non-commutative binding of compositional structures, which makes it possible to predict novel patterns in sequences. To demonstrate this property we apply the network to a commonly used intelligence test called Raven's Progressive Matrices. We present results of simulation experiments for the Raven's task and calculate the probability of prediction error at 95% confidence level. We find that non-commutative binding requires sparse activation of the SDM and that 10–20% concept-specific activation of neurons is optimal. The optimal dimensionality of the binary distributed representations of the VSA is of the order 10^4, which is comparable with former results and the average synapse count of neurons in the cerebral cortex.**

I. Introduction

Analogy-making is the process of identifying and generalizing relational patterns. Unlike typical statistical generalization methods, analogy differs in that it can generalize from very few instances (provided they have sufficiently constraining internal structure). Intentional, conscious analogy-making plays a significant role in language, art, music, engineering and science. More importantly, unconscious, automatic analogy-making is a candidate for the core function of human cognition [1]. Consequently, the development of computational models of analogy-making that can be implemented as neuromorphic [2] or digital connectionist systems may lead to significant advances in cognitive systems.

The ability of a neural system or model to make analogies is intimately connected to the representational abilities of that system. Analogies require complex, relational representations of structure, such as graphs of mult-argument predicates where the arguments may be either atomic symbols or instantiated predicates [3]. Symbolic models of analogy have generally used hand-coded representations of relational structures. Neurally inspired models have typically found the representation of relational structures to be challenging. Some connectionist models assign meanings to individual neurons as localist representations [3]. In this respect they are very similar to traditional symbolic models. One major difficulty with this approach is the implied need to create neurons and connections on the fly to represent all the possible novel relations. Rewiring the brain on the timescale of cognition is physiologically implausible and the number of neurons and synapses required for localist representation vastly exceeds the size of the brain [4]. Neural models that represent concepts as patterns of activity distributed over many neurons can be constructed to represent structured relations. However, the distributed representations have to be hand-coded (similar to symbolic models). Consequently, the distributed representations generated in this way are prone to the criticism that are similar to the localist representations generated by symbolic models. Nevertheless, most of today's computational models of analogy (both connectionist and symbolic) use hand-coded representations. The problem how to encode and ground entities (whether atomic or composite) automatically needs further work. See last paragraph of discussion section about this limitation and suggestions on how to overcome this problem.

Vector Symbolic Architectures (VSAs) are a class of connectionist, distributed representational schemes that are able to easily represent and manipulate relational structures [5]. Representations of composite entities are constructed directly from representations of the components as a consequence of the network architecture without requiring any learning. This avoids the slow learning problems that bedevil the use of distributed representations of composite structures in typical neural networks. Because VSA representations are distributed it is possible to represent multiple entities simultaneously on the same set of neurons by superposing their representations. It is also possible to represent transformations of representations as activity patterns in their own right. These properties make it possible to add novel capabilities to VSA networks on the fly by creating new patterns of activation rather than having to allocate new units and connections as would be required by localist neural networks. Thus, even though VSA networks are constructed from standard connectionist components their functional capabilities differ markedly from typical connectionist networks as a consequence of their architecure, and they are not widely understood. VSAs may be treated as mathematical abstractions and implemented directly as connectionist models with limited biological realism [6], or implemented more realistically with networks of spiking neurons [7].

Given that the representation and manipulation of structure

Blerim Emruli and Fredrik Sandin are with EISLAB at the Luleå University of Technology, 97187 Luleå, Sweden. E-mail: blerim.emruli@ltu.se, fredrik.sandin@ltu.se. Ross W. Gayler is an Honorary Associate of La Trobe University, Bundoora VIC 3086, Australia. E-mail: r.gayler@gmail.com.

is central to analogy and that VSAs can easily represent and manipulate structure there have been multiple applications of VSAs to different aspects of analogy [8]–[11]. We have developed the Analogical Mapping Unit (AMU), a VSA network that combines known principles in a novel way so that analogical mappings of multi-place predicates that generalize to new examples can be learnt [12]. The focus of this paper is to study and demonstrate a non-commutative binding property of the AMU, which enables analogical mapping of ordered sequences of patterns. This is an essential feature for problem solving that is not shared with other binary VSA models of analogy, which generates mapping vectors by superposition and commutative binding of source and target patterns. To demonstrate the learning and inference of sequences in the context of relational mapping we apply our network to a commonly used intelligence test called Raven's Progressive Matrices.

II. Background

In order to set the scene for the empirical demonstration at the core of this paper we need to provide sufficient background on computational modeling of analogy, VSAs and their use for the representation and manipulation of structures, our VSA-based AMU [12], and the Raven's Progressive Matrices task used for our demonstration.

A. Computational models of analogy

Analogy is defined as the process of mapping relations and objects from one particular structure (a source), A, to another particular structure (a target), B; $M : A \rightarrow B$ [10]. The source is familiar or known, whereas the target is a novel composition of relations or objects, relatively unknown and not among the learned examples. Present theories of analogy tend to divide this process in three to five subprocesses, the key ones being: *retrieval*, *mapping* and *inference* [1], [10]. Retrieval concerns finding one out of many potential targets that is likely to be structurally consistent with the source. Mapping concerns finding mappings from the components of the source to components of the target such that the mapped image of the source is maximally structurally consistent with the target. Inference concerns identifying elements of the mapped image of the source which are absent from the target and treating them as predictions of unobserved elements of the target.

Most studies of computational modeling of analogy focus on the mapping stage [10], [13], which can be conceptualized as finding maximal subgraph isomorphisms. The source and target are each presented by a graph with vertices representing components and edges representing relationships between the components. If the vertices and edges of the source can be relabeled (mapped) to exactly replicate the target then the source and target graphs are structurally identical (isomorphic). If some subset of the source can be relabeled to be identical to a subset of the target there is a subgraph isomorphism. When the subset that can be made identical is as large as possible it is a maximal subgraph isomorphism. The central point of graph isomorphism is that mappings must be systematic and that relational structure imposes strong constraints. That is, when a vertex is mapped it must be mapped the same way in all the relationships (edges) that it participates in. Consequently, any computational model of analogical mapping must provide mechanisms to generate and apply mappings that are systematic.

Computational models of analogical mapping usually are categorized as symbolic, connectionist, or symbolic-connectionist hybrids [13], [14]. The mapping stage of the analogy-making process has been studied for quite some time and it is possible to model this stage purely symbolically, for example based on graph matching methods and algorithms. Because of our concern for plausible neural implementation we limit our attention to distributed connectionist models of analogical mapping. Some models such as Analogical Constraint Mapping Engine (ACME) [3] and Learning and Inference with Schemas and Analogies (LISA) [15] have localist or semi-distributed representations of the components, which is outside our field of interest. Distributed Representation Analogy MApper (Drama) [10] is very close to being a distributed re-implementation of ACME. However, although the source and target are represented with VSAs, the network which discovers the mapping between the source and target is localist and has to be constructed on the fly for each problem. These features take Drama outside the scope of our interest. For the interested reader, a comparison between ACME, LISA and Drama is presented in [10].

The connectionist models of analogy-making most similar to our approach are [8], [9], [11]. Each of these is based on VSA networks, and consequently uses distributed representations. Gayler and Levy's model [11] is concerned with discovering the mapping between the source and target representations. It implements a mapping network similar to ACME and Drama except that the implementation is fully distributed and does not require on the fly construction of the network to solve different analogy problems. Plate [9] was concerned with the calculation of the similarity between the VSA representations of the source and target structures. He found that the patterns of similarity between the VSA representations of stimuli were congruent with those observed in human experiments on analogy. Kanerva's approach to analogical mapping [8] is explained in more detail in [16].

B. Vector Symbolic Architectures

Vector symbolic architectures (VSAs) are a class of connectionist representational schemes able to represent and manipulate nested relational structures, or, in other words, *compositional structures* such as trees and graphs [5], [6], [17]. In a VSA all representations are high-dimensonal vectors of the same, fixed dimensionality. The vectors of two arbitrary entities can be combined by a VSA network to yield a new vector representing the composition of the two component entities. The vector representing the composite entity is the same dimensionality as each of the vectors representing the component entites. This enables the representations to be

recursively combined to approximate representations of complex structures such as trees. There are standard operations in VSA networks that allow representations to be composed, decomposed, probed for similarity and transformed in various ways. These operations do not have to be learned as they are automatic consequences of the mathematical structure of VSA networks. VSAs support *holistic transformations* where one compositional structure is mapped onto another compositional structure without having to first decompose the source representation into its components [18], [19]. These mappings can be learned from examples [16, p. 251]. and generalize to map structures composed of novel elements [6], [20] and to structures of higher complexity than those in the training set [18]. These observed properties are interesting, because they suggest that VSA networks naturally provide the mechanisms needed to perform analogical mapping to a level that is psychologically plausible [10], [21] and computationally feasible [13].

Well-known examples of VSAs are the Binary Spatter Code (BSC) [17] and the Holographic Reduced Representation (HRR) [6]. All VSAs provide a product-like operator (binding) and a sum-like operator (bundling) and, optionally, some other operators such as permutation. The domains of the vector elements and the definitions of the operators vary between members of the VSA family. For example, for HRRs the vector elements of HRR are real or complex numbers, binding is implemented as circular convolution, and bundling is implemented by element-wise addition, whereas for BSCs the vector elements are binary, binding is element-wise XOR and bundling is element-wise majority [22]. Despite the differences of implementation between classes of VSA, they appear to be functionally equivalent. This suggests that the functional properties are robust consequences of equipping a high-dimensional vector space with product-like and sum-like operators. Plate made essentially the same point in his discussion of disordered compression of an outer product in the context of neural implementation [23, p. 154].

1) Binary Spatter Codes: Our AMU network is implemented in BSC. Every entity (whether atomic or composite) is represented by a binary vector of dimensionality $\mathcal{D}$, $x_k = (x_{k,1}, x_{k,2}, x_{k,3}, \ldots, x_{k,\mathcal{D}})$. The binding operator, $\otimes$, is defined as the element-wise binary XOR operation. Bundling of multiple vectors x_k is defined as an element-wise binary average $\langle \sum_{k=1}^{n} x_{k,i} \rangle = \Theta\left(1/n \sum_{k=1}^{n} x_{k,i}\right)$ where $\Theta(x) = \{1 \text{ for } x > 0.5,\ 0 \text{ for } x < 0.5,\ \text{random otherwise}\}$ is an element-wise majority rule. Each of the VSA operators (binding, bundling and any other operators defined) can be implemented as a simple connectionist network fragment. The binding and bundling operators take two input vectors and produce a single output vector. At the level of the VSA mathematical abstraction the computation can be construed as a composition of the operators.

2) BSC representation of compositional structures: Atomic entities are usually represented by vectors whose elements are populated randomly with an equal probability of zeros and ones. This ensures that the expected similarity between representations (defined as the correlation of vector elements or cosine of the angle between the vectors) is approximately zero with a very high probability [24]. Entities represented this way behave like classical symbols, they are either identical or completely different, which is the required behavior for the work here, as analogical mapping is generally construed as a symbolic problem. In principle it is possible to construct the atomic representations from the activity of sensors, and it is also possible to have graded similarity structures.

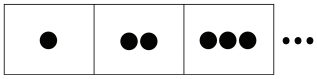

Fig. 1. A sequence of basic symbols.

The network designer chooses the framework used to construct or learn the compositional structures. This is analogous to the way a programmer would choose the data structures required to implement an algorithm. There are many ways to represent a sequence with a VSA, just as a sequence could be implemented by an array, a list, a tree, or some other data structure in a programming language. For example, the representation of the first element of the sequence in Figure 1 might be encoded first by binding pairs of representations and then bundling them together, such as $x = \langle shape \otimes \bullet + position \otimes 1 + number \otimes 1 \rangle$. This is a role-filler binding, where, for example the role, $shape$, is bound to the filler, $\bullet$ [25]. For further examples, see Table II. This is a "true" variable binding, because the elements themselves do not change as a result of the binding, that is, the binding is a completely separate representation from the representations of the components. Moreover, although the representation of the binding is dissimilar to the representations of the role and filler, either component can be recovered from the binding given the other component as a query key (see below).

3) Cleanup memory: In models with localist representations individual neurons correspond to represented entities. Viewing the set of neuron activations as defining a vector space this implies that the axes of that space are distinguished directions. VSA representations are patterns of activation across the neurons. That is, the representations may be arbitrary directions in the vector space. In the absence of any other information it is not possible to tell whether some arbitrary vector represents a composite entity, an atomic entity, or nothing at all. Because of this, VSA systems require a mechanism to establish distinguished directions corresponding to represented entities. This is typically done with a cleanup memory. Items that are to be established as distinguished directions in the vector space can be established on the fly by loading them into cleanup memory during the operation of the network. The archetypal cleanup memory retrieves the stored vector that is most similar to the the input vector.

Cleanup memory has been implemented in a variety of ways including neurally realistic fashion [26]. In recurrent

neural circuits it is sufficient for the retrieved vector to be closer to a distinguished direction than the input vector is. This establishes an attractor in the vector space (the archetypal cleanup memory maps to a point attractor in a single step). Viewing storing items in cleanup memory as establishing attractors opens up possibilities because there are multiple type of attractors (e.g. point, line, plane, cyclic). These different kinds of attractors can be used to perform different computational tasks. Thus, although cleanup memory was introduced in a rather narrow sense it is better viewed as a broader class of computational components for the identification of distinguished directions in a vector space. Our use of Sparse Distributed Memory in the AMU resembles a clean-up memory, which enables generalization by learning of distinguished directions in the vector space of representations.

C. Sparse Distributed Memory

Sparse distributed memory (SDM) is a model of associative and episodic memory, which is thoroughly described by Kanerva [27]. He developed SDM to model some characteristics of human long-term memory. We have included a detailed description of the SDM in the first publication of the AMU [12]. Here we only summarize the essential SDM concepts to clarify the terminology we use. The SDM consists of two parts: a binary address matrix, A, and an integer content matrix, C. These two matrices are initialized as follows; The matrix A is populated randomly with zeros and ones with equal probability, and the matrix C is initialized with zeros. The rows of the matrix A are referred to as address vectors, and the rows of the matrix C are counter vectors. There is a one-to-one link between address vectors and counter vectors, so that an activated address vector corresponds to one specific activated counter vector. The address and counter vectors have dimensionality $\mathcal{D}$. The number of address vectors and counter vectors defines the size of the memory, S.

Each address vector corresponds to an address (determined by its pattern of zeros and ones) in the space of all possible addresses. The number of address vectors, S, is much less than the number of possible addresses, $2^{\mathcal{D}}$, hence they are a sparse sample of the address space. When writing a memory an address vector and data vector are presented to the SDM. It is very unlikely that the input address vector is identical to any of the address vectors in A. Instead, the set of address vectors in A that are sufficiently similar to the input address vector (as defined by Hamming distance) are activated. All the counter vectors corresponding to the activated address vectors are updated with the input data vector. That is, the activated counter vectors are updated but the activated address vectors are static. It is typically sufficient to have six or seven bits of precision in the counter elements [27]. The random nature of the representations makes saturation unlikely with that precision. Reading from the SDM is very similar. The input consists of an address only. All the address vectors sufficiently similar to the input address are activated. The activated counter vectors are summed to an average vector and converted to binary values to yield the retrieved data vector.

SDM can operate as an auto-associative or an hetero-associative memory, simply by ensuring that the input address and data vector are identical or different (respectively). In the AMU network the SDM is used as a hetero-associative memory, it uses one input vector (e.g. X) as the address for storage and retrieval, and the other input vector (e.g. Y) as the data to be stored and retrieved. Storing data Y at address X is functionally equivalent to binding them and retrieving Y given the address X is functionally equivalent to unbinding. Bindings in VSA are represented in the same vector space as the entities to be bound, whereas in SDM the bindings are represented in a different vector space.

In principle, a simple feedforward neural network can implement an auto-associative SDM (Figure 2). In general, the activation and update of counter vectors in an SDM resembles synaptic plasticity in a soft winner take-all population of neurons.

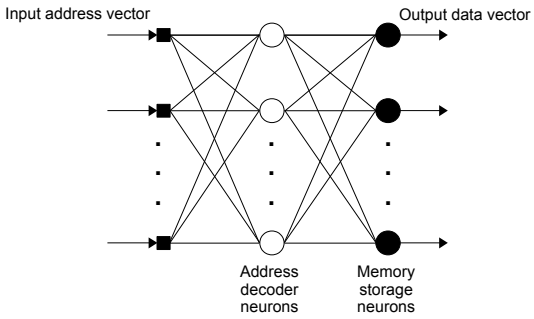

Fig. 2. Sparse distributed memory (SDM) interpreted as a feedforward artificial neural network with sparse activations. An SDM also has a learning circuit with additional inputs, which are not displayed in this figure.

1) BSC representation of analogical mappings: The starting point for the development of the AMU [12] is the idea of holistic mapping vectors [18], [20], [23]. A mapping vector, M, can be constructed that maps one compositional structure to another compositional structure. In traditional connectionist models transformations are implemented by learned patterns of synaptic weights and do not exist in the same representational space as the entity representations related by the transformation. However, VSA mapping vectors are implemented as patterns of activation and exist in the same representational space as the entity representations related by the transformation. Consequently, the representations of transformations may be created and transformed on the fly. This imbues VSA networks with great flexibility relative to traditional connectionist networks.

Kanerva [20] considered mappings of the type "X is the mother of Y" $\rightarrow$ "X is the parent of Y". For our demonstration we use the same mathematical form of the compositional structures to map one circle to two circles, two circles to three circles, and so on, as would be required when interpreting Figure 1 as a sequence with a "successor" relationship between consecutive elements. If we define an

explicit mapping vector in this way $M = \bullet \otimes \bullet\bullet$, it follows that $M \otimes \bullet = \bullet\bullet$ and $M \otimes (\bullet \otimes X) = (\bullet\bullet \otimes X)$ where X is an arbitrary representation, because the XOR-based binding operator commutes and is its own inverse, $\bullet \otimes \bullet = \mathbb{I}$. That is, M is able to transform $\bullet$ into $\bullet\bullet$ in the context of being components bound into a larger structure. When bundling several mapping examples,

$$M_\Sigma = \langle \bullet \otimes \bullet\bullet + \bullet\bullet \otimes \bullet\bullet\bullet + \blacksquare \otimes \blacksquare\blacksquare + \ldots \rangle, \quad (1)$$

the compositional structure of the role–filler relations is integrated in the mapping vector so that it *generalizes* correctly to novel compositional structures $M_\Sigma \otimes \blacktriangle \approx \blacktriangle\blacktriangle$. The symbol $\blacktriangle$ has not been involved in the construction of M_Σ but the mapping anyway results in an analogically correct compositional structure because the mapping vector essentially operates on the number role, transforming the corresponding filler to a pattern that approximately represents the next higher integer. For further examples of analogy making with mapping vectors, see [12], [18], [20], [23].

2) Bidirectionality of explicit mapping vectors: In [12] we argue that robust analogical mapping requires that the mapping direction of VSAs, in particular BSCs, must be controlled. This is necessary to enable learning and inference of novel patterns in ordered sequences. The commutative and involutary (self-inverse) property of the binding operator implies that $M_\Sigma \otimes \bullet\bullet \approx \bullet$, which means that the mapping vector Eq. (1) is *bidirectional*. In this example that poses a problem because we want to encode a mapping so that one circle is mapped to two circles, but not the other way around. Note that the bidirectionality of mappings is problematic also in [20] because "parent" is mapped to "mother", which is not necessarily a good thing since a "father" is also a "parent". Therefore, managing the directionality of the mapping is important to enable learning and prediction of sequences. More generally, compositional structures can be encoded in terms of multi-argument predicates (e.g. $gives(giver, recipient, gift)$) and the arguments cannot be symmetrically substituted for each other.

One way to make the binding operator in BSC and HRR noncommutative is by differentially permuting the elements of the vector arguments of the operator [24], [23, p. 84]. The AMU network incorporates a Sparse Distributed Memory (SDM), which treats its two arguments differently (Section II-C), making the AMU non-commutative. This approach is appealing because it does not require specific permutation and inverse permutation schemes, and the associative memory also solves the problem of organizing multiple mapping vectors (see Section II-D).

D. Analogical Mapping Unit

The VSA models of analogical mapping that are introduced above are limited to explicit, isolated mapping vectors. In practical applications it is necessary to have a framework for the learning and application of many different mappings. This is the purpose of the analogical mapping unit (AMU) that we proposed in [12]. Our AMU is a VSA network for the mapping of distributed representations of compositional structures. Here we briefly summarize the AMU network, which is the basis of the discussion and simulation results that are presented in this paper.

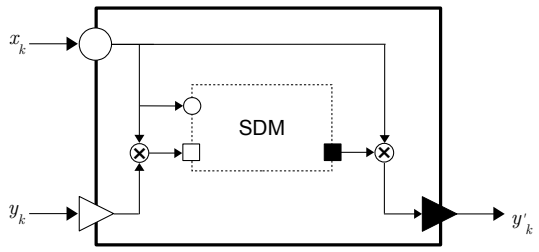

Fig. 3. The analogical mapping unit (AMU) which learns mappings of the type $x_k \rightarrow y_k$ from examples and uses bundled mapping vectors stored in the SDM to calculate the output vector y'_k.

The AMU incorporates an SDM for the storage and retrieval of analogical mapping vectors, see Figure 3. Like the SDM, the AMU takes two binary input vectors and provides one binary output vector, but with a different result and interpretation. The source pattern, x_k, of one particular mapping example is used to activate locations within the SDM where the mapping vector $x_k \otimes y_k$ is stored. This mechanism has two important consequences: 1) Mapping vectors of unrelated sources, x_k, are encoded in different storage locations, which makes it possible to automatically learn multiple unrelated mapping vectors. For example, the two mappings "the circle is above the square" $\rightarrow$ "the square is below the circle" and "A is the mother of B" $\rightarrow$ "A is the parent of B" are unrelated mappings that should be bundled to different storage locations. Similar mapping examples are automatically bundled in nearby storage locations, which is necessary for generalization and analogy making. 2) The SDM activation mechanism breaks the commutative property of the mapping vectors, $\langle \sum_k x_k \otimes y_k \rangle$, because activated locations have addresses that are similar to x_k. The reversed mapping $y_k \rightarrow x_k$ is unlikely as long as a limited subset of the SDM locations are activated in a query because y_k is nearly orthogonal to x_k by chance. This point is demonstrated with simulation experiments in the next section.

The AMU network automates the process of creating, storing and retrieving multiple mappings. Given two input vectors (e.g. the source $x_k = \bullet$ and the target $y_k = \bullet\bullet$ the AMU stores the corresponding mapping vector in the activated SDM locations. This process resembles Hebbian learning. When the source of a learned example is presented, an approximation of the learned target is calculated by the AMU. If a novel source (e.g. $x'_k = \blacktriangle$) that is structurally similar to learned examples is presented, the AMU network calculates an approximate analogically mapped image of the input (e.g. $y'_k = \blacktriangle\blacktriangle$). The AMU has three exogenous parameters: the size S, probability for activation χ of the memory, and the dimensionality $\mathcal{D}$ of the VSA, see Table I. See [12] for further technical details of the AMU.

TABLE I

SUMMARY OF EXOGENOUS PARAMETERS OF THE AMU.

Expression	Description
$\mathcal{D}$	Dimensionality of the binary representations.
S	Size, number of address decoder and data storage neurons.
χ	Average probability for activating an address decoder neuron.

E. Raven's Progressive Matrices

The Raven's Progressive Matrices (RPM) task is a widely used test of fluid intelligence (the ability to solve problems in novel situations). Tasks inspired by RPM are commonly used in computational neuroscience and artificial intelligence as examples of high-level cognitive tasks.

In the RPM test subjects are presented with a 3×3 matrix, in which only the last cell in the bottom right corner is empty, and asked to pick one of the alternative solutions that will correctly complete the matrix, see Figure 4. In this work, a Raven's matrix similar to the one presented in [10] is used, which is in the spirit of the Advanced Progressive Matrices - a more difficult test used to differentiate average from above-average adults [28].

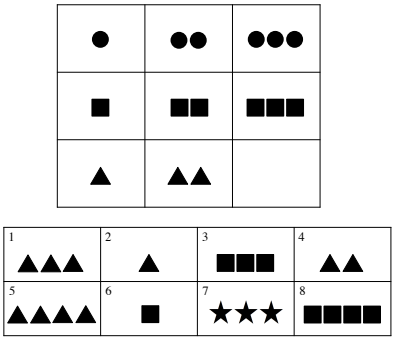

Fig. 4. A basic Raven's Progessive Matrix task. The AMU can predict the missing pattern in the third row, which is a novel pattern that is not explicitly learned. See Figure 5 for a comparison between the eight alternatives displayed here and the pattern inferred by the AMU.

Some of the studies that tackle this task computationally include the well-known model presented in [29], and more recently [10], [30], [31]. Notice that there are differences in the computational architecture and in problem solving focus in all of these models. Our aim here is not to compare the performance of the AMU network with the previous studies but rather to explore if the network is able to encode sequences, and to investigate to which extent the network is successful in solving a particular Raven's matrix.

III. APPLYING THE AMU TO THE RPM TASK

Here we explain step by step how the AMU network learns and correctly solves the Raven's matrix presented in Figure 4. As described in Section II-B.2, compositional representations of symbols are encoded using randomly generated high-dimensional binary vectors. Such vectors are generated for the roles: $shape, position, number$ and the fillers: $1, 2, 3, \bullet, \blacksquare, \blacktriangle$. The roles and fillers are then combined with the binding and bundling operators, for examples see Table II [1].

TABLE II

COMPOSITIONAL STRUCTURES USED IN THE SIMULATION.

Structure	Representation
One circle	$\bullet = \langle shape \otimes \bullet \ + \ position \otimes 1 \ + \ number \otimes 1 \rangle$
Two circles	$\bullet\bullet = \langle shape \otimes \bullet \ + \ position \otimes 2 \ + \ number \otimes 2 \rangle$
Three circles	$\bullet\bullet\bullet = \langle shape \otimes \bullet \ + \ position \otimes 3 \ + \ number \otimes 3 \rangle$
Alternative 1	$\blacktriangle\blacktriangle\blacktriangle = \langle shape \otimes \blacktriangle \ + \ position \otimes 3 \ + \ number \otimes 3 \rangle$
Alternative 2	$\blacktriangle = \langle shape \otimes \blacktriangle \ + \ position \otimes 3 \ + \ number \otimes 1 \rangle$
Alternative 3	$\blacksquare\blacksquare\blacksquare = \langle shape \otimes \blacksquare \ + \ position \otimes 3 \ + \ number \otimes 3 \rangle$

Eq. (1) defines a mapping vector M_Σ that can map one circle to two circles, two circles to three circles and so forth, which is analogous to the rule "increase the number of shapes by one". However, because of the commutative property of the binding operator the mapping vector is bidirectional so that two circles can be mapped to one circle, three circles to two circles and so on. In Section II-D we presented the AMU network, which solves this problem by breaking the commutative property of the mapping vectors with an associative memory.

To construct a generic mapping vector that will correctly solve the RPM task we present mapping examples to the AMU for each sequence of symbols of the RPM task in a top-down, row-wise way. Eye-tracking studies show that humans use a top-down and row-wise scanning strategy when solving this kind of task [29]. When the mapping examples are presented to the AMU the corresponding mapping vectors are bundled in the SDM of the AMU, which enables generalization by learning the mappings "after one X follows two X" and "after two X follows three X".

After learning the first five mapping examples the AMU network correctly predicts the last cell in Figure 4 given the input vector $x_k = \blacktriangle\blacktriangle$. The corresponding output vector, y'_k, is compared with eight alternative solutions in terms of the vector element correlation coefficient, ρ. The different alternative solutions are encoded in the same way as the Raven's matrix symbols, for examples see the last three rows of Table II. The result of this simulation is presented in Figure 5. The alternative with the highest correlation is selected as the correct answer. The parameters of the AMU are set to $S = 1000$, $\mathcal{D} = 8192$ and $\chi = 0.1$. This implies that the AMU has 1000 locations out of which 100 are activated in each storage and retrieval operation. We estimate the confidence interval for the probability of error with the Agresti-Coull interval [32], which is similar to the pragmatic rule "add 2 successes and 2 failures". The simulations are repeated until the 95% margin of error is one tenth of the probability of error (thousands of repetitions).

[1]The astute reader will have noticed that we have restricted ourselves to only one, simple RPM task and encoded precisely the attributes required to solve that task. We did this because we are using this task as a proof of principle demonstration of problem solving by systematic substitution rather than a realistic general purpose problem solver (which would require a much more capable system to learn the concepts needed to recognise a problem and solve it).

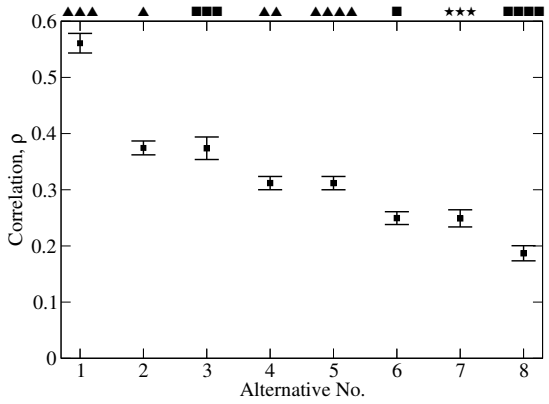

Fig. 5. The correlation, ρ, between the prediction of the AMU and eight different alternative solutions to the Raven's Progressive Matrix task. Error bars denote standard deviations. The parameters of the AMU are $S = 1000$, $\mathcal{D} = 8192$ and $\chi = 0.1$. Alternative number one has the highest correlation, $\rho = 0.56 \pm 0.017$. The probability that the AMU predicts an incorrect alternative is $0.004 \pm 8 \cdot 10^{-4}$ at 95% confidence level.

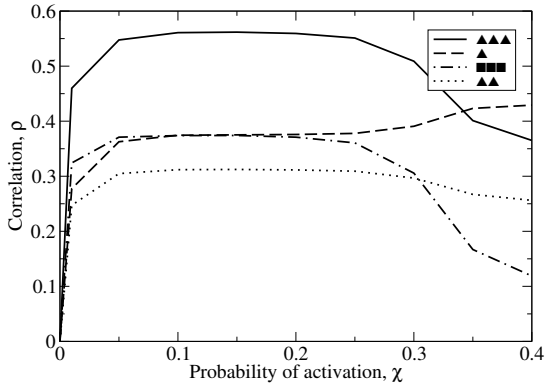

Fig. 6. Correlations between different alternative solutions and the prediction of the AMU versus the probability of activating a storage location, χ. Other parameters of the AMU are $S = 1000$ and $\mathcal{D} = 8192$. The correlation between the first and second alternative are similar for $\chi \gtrsim 0.35$.

Another interesting result appears when we vary the average probability of activating an SDM location, χ, see Figure 6. The activation probability affects the performance of the AMU significantly. For example, for $\chi > 0.3$ the AMU predicts that one triangle and three triangles are equally correct answers, which is not the case. At high values of χ the commutative property of the mapping is restored because the SDM association mechanism can no longer distinguish between forward and backward mappings. Figure 6 suggests that the optimal sparseness of the AMU for solving the RPM task is between $0.1 < \chi < 0.2$. A sparse representation of the mappings is required for prediction of patterns in sequences.

Up to this point the simulations are based on $\mathcal{D} = 8192$. We illustrate the effect of varying the dimensionality of the VSA in Figure 7. This result suggest that the optimal choice for the dimensionality of the BSC representations is of the order 10^4, which is consistent with the results for other tasks [12], [20], [24]. A slightly higher dimensionality than 10^4 may be motivated, but 10^5 is too much because it does not improve the probability of error and requires more storage space and computation. Kanerva derived this result also from basic statistical properties of hyperdimensional binary spaces [27]. This is an interesting result in a neural network context because that number matches the average number of excitatory synapses on pyramidal cells in the cerebral cortex. Of course, this may be entirely coincidental, but it suggests lines of enquiry attempting to explain facets of neurophysiology in terms of the the impact on computational processes.

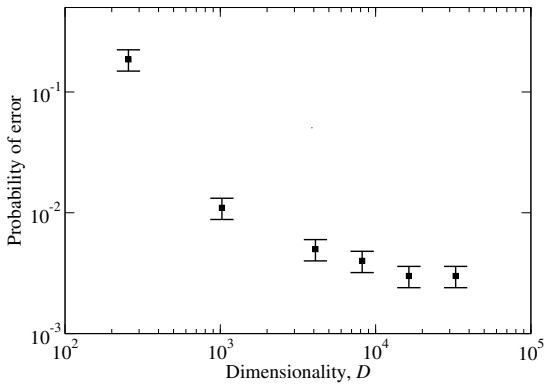

Fig. 7. The probability that the AMU predicts an incorrect alternative versus the dimensionality of the VSA representations, $\mathcal{D}$. Other parameters of the AMU are $S = 1000$ and $\chi = 0.1$. The optimal dimensionality is of the order $\mathcal{D} = 10^4$. Error bars indicate 95% confidence intervals.

IV. Discussion

Robust analogical mapping requires that the direction of mappings is controlled, so that when a network learns $x_k \rightarrow y_k$ the reversed mapping $y_k \rightarrow x_k$ is not implicitly learned. The commutative property of the BCS binding operator makes isolated mapping vectors bidirectional, which is problematic for inference. Permutations can be used to encode ordered sequences, for example by permuting the elements of one vector before the binding takes place [24]. In this paper we present an alternative approach for non-commutative binding, which results from a combination of the ordinary XOR binding operator of BSC with an associative memory in the AMU network.

We study the non-commutative binding property of the AMU network by testing it on an RPM task which is similar to that considered in [33]. The AMU network predicts the missing symbol in a 3×3 RPM with high probability after learning the five known mapping examples, provided that the activation of neurons in the AMU is sparse (on the order of 10–20%). We find that the AMU is unable to distinguish between forward and reversed mappings if more than about 35% of the neurons are activated. In [33] the RPM symbols are encoded as superpositions of HRR role-filler bindings, and the commutative property of the binding operator is broken by transforming one of the factors to its inverse. This enables the HRR network to learn sequences through inductive reasoning in the form of a general rule; "increase the number of shapes by one". The problem of

how to achieve this kind of reasoning with a BSC is an open question.

Although simple in its design, the AMU network has a few remarkable properties: 1) The optimal dimensionality of the BSC is of the order 10^4, which is comparable to the average synapse count on principal cells in the cerebral cortex. This interesting coincidence of the dimensionality $\mathcal{D}$ with neural measurements complements some previous studies [20], [27], as well as our own previous work [12] where the AMU network was introduced. 2) Sparse activation, the network performs well when 10-20% of the neurons are activated. 3) Fast learning, the network can generalize correctly after learning three examples and improves with further training [12]. In addition, note that the AMU relies on a combination of fully distributed representation (the BSC) and concept-specific activation of neurons in the SDM, which resembles concept cells in neurobiology.

An important next step is to encode compositional structures directly from sensory data, which is widely referred to as the "encoding problem" [34]. Previous studies have also acknowledged the limitation of hand-coded representations [29], [33], [35] and some partially handled it [30]. It is possible to address this problem by incorporating an encoding network, for example in the form of receptive fields and deep learning networks.

ACKNOWLEDGEMENTS

B.E. thanks Monash University for hosting him as a visiting researcher and the Australian National University and NICTA for the scholarship granted to attend the Machine Learning Summer School 2010, which stimulated the collaboration with R.G. This work is partially supported by the Swedish Foundation for International Cooperation in Research and Higher Education (STINT) under the grant number IG2011-2025.

REFERENCES

[1] K. J. Holyoak, D. Gentner, and B. N. Kokinov, "Introduction: The place of analogy in cognition," in *The Analogical Mind: Perspectives from Cognitive Science*, pp. 1–19, MIT Press, 2001.

[2] G. Indiveri and T. K. Horiuchi, "Frontiers in neuromorphic engineering," *Frontiers in Neuroscience*, vol. 5, no. 118, 2011.

[3] K. J. Holyoak and P. Thagard, "Analogical mapping by constraint satisfaction," *Cognitive Science*, vol. 13, no. 3, pp. 295–355, 1989.

[4] T. Stewart and C. Eliasmith, "Compositionality and biologically plausible models," in *The Oxford Handbook of Compositionality* (M. Werning, W. Hinzen, and E. Machery, eds.), Oxford University Press, 2012.

[5] R. W. Gayler, "Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience," in *Proceedings of the Joint International Conference on Cognitive Science*, pp. 133–138, 2003.

[6] T. A. Plate, "Holographic reduced representations," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 623–641, 1995.

[7] C. Eliasmith and C. H. Anderson, *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. A Bradford Book, Aug. 2004.

[8] P. Kanerva, "Dual role of analogy in the design of a cognitive computer," in *Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational, and Neural Sciences* (D. Gentner, K. J. Holyoak, and B. N. Kokinov, eds.), New Bulgarian University, 1998.

[9] T. A. Plate, "Analogy retrieval and processing with distributed vector representations," *Expert Systems: The International Journal of Knowledge Engineering and Neural Networks, Special Issue on Connectionist Symbol Processing*, vol. 17, no. 1, pp. 29–40, 2000.

[10] C. Eliasmith and P. Thagard, "Integrating structure and meaning: a distributed model of analogical mapping," *Cognitive Science*, vol. 25, no. 2, pp. 245–286, 2001.

[11] R. W. Gayler and S. Levy, "A distributed basis for analogical mapping," in *Proceedings of the 2nd International Conference on Analogy*, (New Bulgarian University, Sofia), 2009.

[12] B. Emruli and F. Sandin, "Analogical mapping with sparse distributed memory: a simple model that learns to generalize from examples," *Cognitive Computation*, vol. E-pub ahead of print, pp. 1–15, 2013.

[13] D. Gentner and K. D. Forbus, "Computational models of analogy," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 2, no. 3, pp. 266–276, 2011.

[14] R. M. French, "The computational modeling of analogy-making," *Trends in Cognitive Sciences*, vol. 6, no. 5, pp. 200–205, 2002.

[15] J. E. Hummel and K. J. Holyoak, "Distributed representations of structure: A theory of analogical access and mapping," *Psychological Review*, vol. 104, no. 427–466, 1997.

[16] P. Kanerva, "Computing with large random patterns," in *The Foundations of Real-World Intelligence* (Y. Uesaka, P. Kanerva, and H. Asoh, eds.), no. 251-269, CSLI Publications, 2001.

[17] P. Kanerva, "Binary spatter-coding of ordered k-tuples," in *Proceedings of the International Conference on Artificial Neural Networks*, pp. 869–873, 1996.

[18] J. Neumann, "Learning the systematic transformation of holographic reduced representations," *Cognitive Systems Research*, vol. 3, no. 2, pp. 227–235, 2002.

[19] J. Hammerton, "Holistic computation: Reconstructing a muddled concept," *Connection Science*, vol. 10, no. 1, pp. 3–19, 1998.

[20] P. Kanerva, "Large patterns make great symbols: An example of learning from example," in *Hybrid Neural Systems* (S. Wermter and R. Sun, eds.), vol. 1778, pp. 194–203, Springer, 2000.

[21] D. Gentner and A. B. Markman, "Defining structural similarity," *The Journal of Cognitive Science*, vol. 6, pp. 1–20, 2006.

[22] R. W. Gayler, "Multiplicative binding, representation operators & analogy (workshop poster)," in *Advances in analogy research: Integration of theory and data from the cognitive, computational, and neural sciences* (D. Gentner, K. J. Holyoak, and B. N. Kokinov, eds.), p. 405, Sofia, Bulgaria: New Bulgarian University, 1998.

[23] T. A. Plate, *Distributed Representations and Nested Compositional Structure*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Canada, 1994.

[24] P. Kanerva, "Hyperdimensional computing: an introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, 2009.

[25] P. Smolensky, "Tensor product variable binding and the representation of symbolic structures in connectionist systems," *Artificial Intelligence*, vol. 46, no. 1-2, pp. 159–216, 1990.

[26] T. C. Stewart, Y. Tang, and C. Eliasmith, "A biologically realistic cleanup memory: Autoassociation in spiking neurons," *Cognitive Systems Research*, vol. 12, no. 2, pp. 84–92, 2011.

[27] P. Kanerva, *Sparse Distributed Memory*. The MIT Press, 1988.

[28] J. Raven, J. C. Raven, and J. H. Court, *Raven's Progressive Matrices and Vocabulary Scales*. Oxford Psychologists Press, 1998.

[29] P. A. Carpenter, M. A. Just, and P. Shell, "What one intelligence test measures: a theoretical account of the processing in the Raven Progressive Matrices test," *Psychological review*, vol. 97, pp. 404–431, July 1990.

[30] A. Lovett, K. D. Forbus, and J. Usher, "A structure-mapping model of Raven's Progressive Matrices," in *Proceedings of the 32nd annual conference of the cognitive science society*, 2010.

[31] M. Kunda, K. McGreggor, and A. K. Goel, "A computational model for solving problems from the Raven's Progressive Matrices intelligence test using iconic visual representations," *Cognitive Systems Research*, vol. 22–23, pp. 47–66, 2013.

[32] A. Agresti and B. A. Coull, "Approximate is better than "Exact" for interval estimation of binomial proportions," *The American Statistician*, vol. 52, no. 2, pp. 119–126, 1998.

[33] D. Rasmussen and C. Eliasmith, "A neural model of rule generation in inductive reasoning," *Topics in Cognitive Science*, vol. 3, no. 1, pp. 140–153, 2011.

[34] S. Harnad, "The symbol grounding problem," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1-3, pp. 335–346, 1990.

[35] D. R. Little, S. Lewandowsky, and G. L. Thomas, "A bayesian model of rule induction in Raven's Progressive Matrices," in *Proceedings of the 34th annual conference of the cognitive science society*, 2012.