"Lateral Inhibition" in a Fully Distributed Connectionist Architecture

Simon D. Levy (levys@wlu.edu)
Department of Computer Science, Washington and Lee University
Lexington, VA 24450 USA

Ross W. Gayler (r.gayler@gmail.com)
Department of Psychology, La Trobe University
Victoria 3086 Australia

Abstract

We present a fully distributed connectionist architecture supporting lateral inhibition / winner-takes all competition. All items (individuals, relations, and structures) are represented by high-dimensional distributed vectors, and (multi)sets of items as the sum of such vectors. The architecture uses a neurally plausible permutation circuit to support a multiset intersection operation without decomposing the summed vector into its constituent items or requiring more hardware for more complex representations. Iterating this operation produces a vector in which an initially slightly favored item comes to dominate the others. This result (1) challenges the view that lateral inhibition calls for localist representation; and (2) points toward a neural implementation where more complex representations do not require more complex hardware.

Keywords: Lateral inhibition; winner-takes-all; connectionism; distributed representation; Vector Symbolic Architecture

Introduction

Connectionist representations are typically classified as localist, distributed, or some combination of both. In a localist representation each node corresponds to a single item or concept. In a distributed representation each node participates in the representation of every concept, and each concept is "spread out" (distributed) among every node. Proponents of localist representation cite simplicity and transparency as benefits of localist coding. Proponents of distributed representations argue that the robustness of such representations in the presence of noise makes them more plausible and appealing, and cite related impressive work on modeling neuropsychological disorders using distributed connectionist representations. For a review see Olson & Humphreys (1997). A comprehensive argument for distributed representations is of course beyond the scope of this article. We will focus here instead on a particular capability that appears to be exclusive to localist representations, and will provide an alternative analysis using a distributed representation.

In a 2000 target article in *Behavioral and Brain Sciences*, Page (2000) argues for a "generalized localist model" with a localist representation on one layer and general (distributed) o representations on the others. Each node in the localist layer is associated with a category, and a lateral inhibition (winner-takes-all competition) function is used, allowing the localist layer to act as a classifier for (distributed) patterns on an incoming layer. Indeed, the ability of localist representations to support competitive classification seems to be the main appeal of localism, as suggested by the remarks of the commentators who supported Page's position (*e.g.* Phaf & Wolters, 2000).

In this article we will argue that localist representations are not necessary to support winner-takes-all competition or lateral inhibition in general. We will present a fully distributed connectionist architecture supporting lateral inhibition / winner-takes all behavior, in which all items (individuals, relations, and structures) are represented by high-dimensional distributed vectors, and (multi)sets of items as the sum of such vectors. Unlike a localist representation, such representations are based on a fixed neural architecture that does not need to grow as new representational categories are added.

Problems with Localism

The greatest challenge to connectionist accounts of cognition continues to be the problem of compostionality, that is, the problem of how to put simpler items like words and concepts together to make more complex structures like sentences and propositions (Fodor & Pylyshyn, 1988; Jackendoff, 2002). Localist connectionism addresses this challenge by assigning one neuron or pool of neurons to each item, and employing additional (pools of) neurons as higher-order elements for organizing the simpler items via physical connections or temporal synchrony. For example, the Neural Blackboard Architecture of van der Velde (2006) builds sentences out of words via "structure assemblies" corresponding to traditional syntactic categories like Noun Phrase and Verb Phrase. Hummel and Holyoak's LISA model of analogical mapping (Hummel & Holyoak, 1997) uses higher-order assemblies to represent the bindings of individuals to semantic roles like agent and patient.

In a forthcoming article, Stewart and Eliasmith (Stewart & Eliasmith, forthcoming) provide a detailed analysis of the computational complexity entailed by localist accounts of composition. This analysis suggests that the need to have physical connections between all pairs of items causes localist representations lead to a combinatorial explosion when applied to realistically-sized item inventories, such as the vocabularies of natural languages. An alternative approach, which dates back to the work of Pollack (1990) and others, attempts to encode structures of arbitrary complexity on a fixed-size connectionist architecture.[1] Commenting on Pol-

[1] A serious limitation of Pollack's Recursive Auto-Associative (RAAM) network was the need to *learn* representations (via backpropagation). The work presented here avoids the need for learning,

lack's results, Hammerton (1998) notes that it is important to consider whether the representations produced by such architectures can be manipulated holistically, or whether they require "functional localism", such as serial extraction of components, in order to support useful computations. Even if it is neurally plausible, which seems unlikely, functional localism strikes us as essentially an implementation of classical symbol processing (cf. Fodor & Pylyshyn, 1988), foregoing much of the appeal of connectionism.

Another problem with localist implementation of lateral inhibition is that the system can only implement winner-takes-all; that is, the result is the choice of 1 out of k alternatives. For some problems, however, it would be more appropriate to have a *set* of answers returned. It is not clear how this would be achievable with localist winner-take-all implementation. What is needed is a new type of network that exhibits the attractor dynamics of localist winner-takes-all networks, but which can converge simultaneously to a *set* of items, rather than a single item.

The work presented here addresses these issues, providing a holistic implementation of an operation previously thought to require localist coding.

Vector Symbolic Architectures

Vector Symbolic Architecture is a name that we coined to describe a class of connectionist models that use high-dimensional vectors (with as few as 1000 dimensions, but more typically around 10,000) of low-precision numbers to encode structured information as distributed representations. That is, VSAs can represent complex entities such as trees and graphs; and every such entity, no matter how simple or complex, is represented by a pattern of activation distributed over all the elements of the vector. This general class of architectures traces its origins to the tensor product work of Smolensky (1990), but avoids the exponential growth in dimensionality of tensor products. The currently available VSAs employ three types of operation on vectors: a multiplication-like operator, an addition-like operator, and a permutation-like operator. The multiplication-like operation is used to associate or bind vectors. The addition-like operation is used to superpose vectors or add them to a set. The permutation-like operation is used to quote or protect vectors from the other operations.

The use of hyperdimensional vectors to represent symbols and their combinations provides a number of mathematically desirable and biologically realistic features. A hyperdimensional vector space can contain as many mutually orthogonal vectors as there are dimensions, and exponentially many almost-orthogonal vectors (Hecht-Nielsen, 1994), thereby supporting the representation of astronomically large numbers of distinct items. Such representations are also highly robust to noise: a significant fraction of the values in a vector can be randomly changed before it becomes more similar to another vector than to its original form. To cite a result from a forthcoming paper by Kanerva (in press): *When meaningful entities are represented by 10,000-[element] vectors, many of the bits can be changed more than a third by natural variation in stimulus and by random errors and noise, and the resulting vector can still be identified with the correct one, in that it is closer to the original "error-free" vector than to any unrelated vector chosen so far, with near certainty.* It is also possible to implement such vectors in a spiking neuron model (Eliasmith, 2005), lending them a further degree of biological plausibility.

The main difference among types of VSAs is in the kind of numbers used as vector elements and the related choice of multiplication-like operation. Holographic Reduced Representations (Plate, 2003) use real numbers and circular convolution. Binary Spatter Codes (Kanerva, 1994) use binary (Boolean) values and elementwise exclusive-or. MAP (Multiply, Add, Permute) coding (Gayler, 1998) uses bipolar (-1/+1) values and elementwise multiplication. A useful feature of BSC and MAP is that every vector is its own multiplicative inverse: multiplying a vector by itself elementwise yields the multiplicative identity vector ($A * A = \mathbf{1} = B * B$, where $\mathbf{1}$ is the identity vector, but $A + A = 2A$). As in ordinary algebra, multiplication and addition are associative and commutative, and multiplication distributes over addition.

We used MAP in the work described here. In MAP, properties are *accumulated* through vector addition; hence, it is trivial to have multiple, self-reinforcing copies of the same property (vector) in a single representation. For example, given a vector representation A of the property *affluent* and a vector representation B of the property *brave*, the representation $A + A + B = 2A + B$ could represent being very affluent and somewhat brave. Second, the *association* of two representations through elementwise multiplication produces a third representation that is completely dissimilar from both elements. If C represents an individual, say, Charlie, the proposition that Charlie is brave could be represented as $B * C$, whose similarity (vector cosine) with both B and C is close to zero. Together, these facts mean that a given entity can be associated with a large number of properties (and vice versa): $C * (2A + B)$, etc.

Without an additional mechanism, self-cancellation *would* pose a challenge when copies of structures are embedded in themselves recursively. For example, if D_1 and D_2 represented the semantic roles *doubter* and *(thing) doubted*, then one possible way to represent the proposition *Bill doubted that Charlie doubted that Ed is affluent* as

$$D_1 * B + D_2 * (D_1 * C + D_2 * A * E)$$

Without further modification, the two copies of D_2 would have the undesired affect of canceling each other out. As mentioned above and discussed at length in (Levy, to appear), the permutation operator of the MAP architecture provides a neurally plausible mechanism for quoting or protecting vectors in these situations.

by relying on fixed, constant-time mechanisms for associating and composing vector representations.

As an example of holistic computation in MAP, consider the common task of retrieving a set of items associated with a given property. We imagine three individuals: A and B having property P and C having property Q. In a MAP encoding, each individual and property would be encoded in a hyper-dimensional vector, and the association of properties with individuals would be the vector sum of the elementwise products between each individual and its property:

$$V = A*P + B*P + C*Q$$

To retrieve the set of individuals having property P, we multiply the "knowledge-base" vector V by P. The self-inverse property of MAP produces a representation of the individuals A and B, as well as a "noise" component not corresponding to any individual or property "known" to the system:

$$\begin{gathered} P*V = \\ P*(A*P+B*P+C*Q) = \\ A*P*P \ + \ B*P*P \ + \ C*Q*P = \\ A \ + \ B \ + \ C*Q*P = \\ A \ + \ B \ + \ noise \end{gathered}$$

Comparing this resulting vector to the vectors for each of the individuals will yield a high similarity (dot product, cosine) between the result vector and both A and B, but not C. In other words, a single holistic computation on two vectors (P and V) has retrieved structurally sensitive information about distinct individuals, without (1) the need for explicit physical connections among the individuals (and the concomitant additional representational hardware) or (2) a functionally localist decomposition. This power comes at the cost of of noise in the retrieved representation, which is not a deal-breaker for this example. If noise becomes a problem (as it can in recurrent circuits like the analogy-mapping circuit described below, where noise accumulates), the noise can be removed from the result vector by passing the vector through a "cleanup memory" that stores only the meaningful items, or vector directions: here, A, B, and C.

The issue of noise in VSA is rather subtle. In a localist representation there are distinguished directions in the vector space that correspond to the individual units, because individual units represent individual concepts. In VSA there are no inherently distinguished directions. For example, the vector X might represent the concept A, but it could just as well represent $A+B$ or $C*D+E$, etc. The functional equivalent of distinguished directions is provided by the contents of the cleanup memory, which are initialized for a particular problem. Noise is then any pattern which is not stored in cleanup memory. Unlike localist representations, which require reconfiguring the "hardware" for each new problem, VSA reuses the same fixed cleanup hardware (e.g. an autoassociative Hofpield network) for every problem.

Lateral Inhibition as Self-Intersection

Consider a situation in which three categories A and B, and C are competing with one another on a given neural layer L_2, to classify input patterns on an incoming layer L_1. An example localist implementation is shown in Figure 1. Each node in L_2 has an inhibitory connection to every other node in that layer. The connections from L_1 to L_2 can be interpreted as setting the state of L_2 to reflect the initial evidence for each of the categories. The inhibitory connections within L_2 implement a recurrent process that increases the differences between the most supported category and the other categories.

We can interpret the state of L_2 as a multiset - a set of weighted elements. Each category X_i (here, A, B, or C) is weighted by a non-negative real-valued coefficient k_i that reflects the importance of X_i in the multiset, with $0 \leq k_i \leq 1$. Given this interpretation of the L_2 state as a multiset we need a multiset operation that increases the differences between the most supported category and the other categories. We do this with multiset intersection (multiplication of the corresponding category weights) and normalization (constraining the sum of the category weights to be constant).

To see what we mean by multiset intersection, consider multisets $X = \{k_1A, k_2B, k_3C\}, Y = \{k_4A, k_5B, k_6C\}$. Intersecting X and Y would would yield a multiset $\{k_1k_4A, k_2k_5B, k_3k_6C\}$. Intersecting X with itself would yield $(k_1)^2A \ + \ (k_2)^2B \ + \ (k_3)^2C$, magnifying the differences between the k_i. Normalization of the result forces the smaller k_i towards zero. The repeated application of self-intersection with normalization yields a similar dynamic to lateral inhibition thereby implementing winner-takes-all competition.

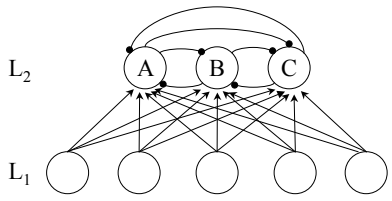

Figure 1: Lateral inhibition in a localist network

In a localist network like the one in Figure 1, the multiset coefficients k_i correspond to the activations of the nodes in the L_2 layer. In VSA, a multiset is represented as a single vector, for example, $k_1A \ + \ k_2B \ + \ k_3C$ where A, B, and C are hyperdimensional vectors and k_1, k_2, and k_3 are non-negative scalars. Note that this network can *only* represent a choice between A, B, and C. No other category can be considered without modifying the physical structure of the network.

How are we to perform the multiset intersection of two such vectors? Because of the self-cancellation property of the MAP architecture, simple elementwise multiplication (the standard MAP product operator) of the two vectors will not implement this operation. We could extract the k_i by iterating through each of the vectors A, B, and C and dividing x and y elementwise by each mapping, but this is the very kind of functionally localist approach that we are trying to avoid.

To implement this intersection operator in a holistic, distributed manner we exploit the third component of the MAP architecture: permutation. For explanatory purposes we can conceive of our solution as a simple register-based machine, where (as in a traditional von Neumann architecture), each register holds a temporary stage of the computation. (In our version, of course, the register contents are hyperdimensional vectors.). As depicted in Figure 2, our solution works as follows: 1: and 2: are registers loaded with the vectors representing the multisets to be intersected. $\mathrm{P}_1()$ computes some fixed permutation of the vector in 1:, and $\mathrm{P}_2()$ computes a different fixed permutation of the vector in 2: (randomly chosen permutations are sufficient). Register 3: contains the product (via elementwise multiplication) of these permuted vectors. Register 4: is another variety of "cleanup" memory (a constant vector value) pre-loaded with each of the principal vectors transformed by multiplying it with permutations of itself; *i.e.*, $4 := \Sigma_{i=1}^{n} X_i * \mathrm{P}_1(X_i) * \mathrm{P}_2(X_i)$. In other words, register 4: indicates the items of interest to the system and is functionally analogous to the L_2 units in the localist network; however, the contents of the register can be changed at any time without modifying the underlying hardware. Note so that each of these registers contains a high-dimensional vector representing an arbitrarily complex multiset, and each arrow in Figure 2 represents the transfer of a high-dimensional vector.

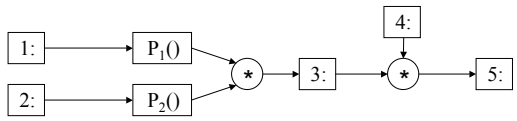

Figure 2: A neural circuit for vector intersection.

In brief, the circuit in Figure 2 works by guaranteeing that the permutations will cancel for only the subset of X_i present in both input registers, with the other X_i being rendered as random noise. In order to improve noise-reduction it is necessary to take the sum over several such intersection circuits, each based on different permutations. This sum over permutations has a natural interpretation in the synaptic connections between neural layers of sigma-pi units. Each unit (neuron) in one layer calculates the sum over many products of a few inputs from units in the prior layer. The apparent complexity of Figure 2 is a consequence of drawing it for explanatory clarity rather than computational complexity. The intersection network could be implemented in a single layer of sigma-pi units.

To see how this circuit implements intersection, consider again the simple case of a system with three meaningful vectors A, B, and C where we want to compute the intersection of $x = k_1A + k_2B + k_3C$ with $y = k_4A + k_5B + k_6C$. The vector x is loaded into register 1:, y is loaded into 2:, and the sum

$$A * \mathrm{P}_1(A) * \mathrm{P}_2(A) + B * \mathrm{P}_1(B) * \mathrm{P}_2(B) + C * \mathrm{P}_1(C) * \mathrm{P}_2(C)$$

is loaded into 4:. After passing the register contents through their respective permutations and multiplying the results, register 3: will contain

$$\mathrm{P}_1(k_1A + k_2B + k_3C) * \mathrm{P}_2(k_4A + k_5B + k_6C) =$$
$$(k_1\mathrm{P}_1(A) + k_2\mathrm{P}_1(B) + k_3\mathrm{P}_1(C)) * (k_4\mathrm{P}_2(A) + k_5\mathrm{P}_2(B) + k_6\mathrm{P}_2(C)) =$$
$$k_1k_4\mathrm{P}_1(A) * \mathrm{P}_2(A) + k_2k_5\mathrm{P}_1(B) * \mathrm{P}_2(B) + k_3k_6\mathrm{P}_1(C) * \mathrm{P}_2(C) + \textit{noise}$$

where *noise* represents terms not corresponding to a meaningful component of the intersection. Multiplying this sum in register 3: by the contents of register 4: will then result in the desired intersection (plus additional noise), via the self-cancellation property:

$$[k_1k_4\mathrm{P}_1(A) * \mathrm{P}_2(A) + k_2k_5\mathrm{P}_1(B) * \mathrm{P}_2(B) + k_3k_6\mathrm{P}_1(C) * \mathrm{P}_2(C)] *$$
$$[A * \mathrm{P}_1(A) * \mathrm{P}_2(A) + B * \mathrm{P}_1(B) * \mathrm{P}_2(B) + C * \mathrm{P}_1(C) * \mathrm{P}_2(C)] =$$
$$k_1k_4\mathrm{P}_1(A) * \mathrm{P}_2(A) * A * \mathrm{P}_1(A) * \mathrm{P}_2(A) +$$
$$k_1k_4\mathrm{P}_1(A) * \mathrm{P}_2(A) * B * \mathrm{P}_1(B) * \mathrm{P}_2(B) +$$
$$k_1k_4\mathrm{P}_1(A) * \mathrm{P}_2(A) * C * \mathrm{P}_1(C) * \mathrm{P}_2(C) +$$
$$k_2k_5\mathrm{P}_1(B) * \mathrm{P}_2(B) * A * \mathrm{P}_1(A) * \mathrm{P}_2(A) +$$
$$k_2k_5\mathrm{P}_1(B) * \mathrm{P}_2(B) * B * \mathrm{P}_1(B) * \mathrm{P}_2(B) +$$
$$k_2k_5\mathrm{P}_1(B) * \mathrm{P}_2(B) * C * \mathrm{P}_1(C) * \mathrm{P}_2(C) +$$
$$k_3k_6\mathrm{P}_1(C) * \mathrm{P}_2(C) * A * \mathrm{P}_1(A) * \mathrm{P}_2(A) +$$
$$k_3k_6\mathrm{P}_1(C) * \mathrm{P}_2(C) * B * \mathrm{P}_1(B) * \mathrm{P}_2(B) +$$
$$k_3k_6\mathrm{P}_1(C) * \mathrm{P}_2(C) * C * \mathrm{P}_1(C) * \mathrm{P}_2(C) =$$
$$k_1k_4A +$$
$$k_1k_4\mathrm{P}_1(A) * \mathrm{P}_2(A) * B * \mathrm{P}_1(B) * \mathrm{P}_2(B) +$$
$$k_1k_4\mathrm{P}_1(A) * \mathrm{P}_2(A) * C * \mathrm{P}_1(C) * \mathrm{P}_2(C) +$$
$$k_2k_5\mathrm{P}_1(B) * \mathrm{P}_2(B) * A * \mathrm{P}_1(A) * \mathrm{P}_2(A) +$$
$$k_2k_5B +$$
$$k_2k_5\mathrm{P}_1(B) * \mathrm{P}_2(B) * C * \mathrm{P}_1(C) * \mathrm{P}_2(C) +$$
$$k_3k_6\mathrm{P}_1(C) * \mathrm{P}_2(C) * A * \mathrm{P}_1(A) * \mathrm{P}_2(A) +$$
$$k_3k_6\mathrm{P}_1(C) * \mathrm{P}_2(C) * B * \mathrm{P}_1(B) * \mathrm{P}_2(B) +$$
$$k_3k_6C =$$
$$k_1k_4A + k_2k_5B + k_3k_6C + \textit{noise}$$

Note that this apparently complex calculation is actually a single elementwise vector product operation. The circuit does not "see" the complexity of the vectors it operates on. The same holds true for the normalizing operator mentioned above: normalization is implemented as a scalar multiplier applied to the entire vector to keep the sum of the element activations approximately constant.

Experimental Results

As a proof-of-concept for our distributed lateral inhibition architecture, we ran several experimental trials using the circuit from Figure 2. We started with an initial vector $x_0 = 1/N\sum_{i=1}^{N} k_iX_i$, with $k_j = 1.02$ for one arbitrarily chosen j and $k_i = 1$ for $i \neq j$. We then iterated the operation $x_{t+1} = \textit{normalize}(x_t \wedge x_t)$, where $\wedge$ is the intersection operator in Figure 2, and $\textit{normalize}(x) = x/\max_l(|x_l|)$. (The initial conditions thus represent a temporary violation of the constraints given above for k_i that are immediately rectified by the normalizing operation.) We stopped iterating when the Euclidean distance between x_t and x_{t-1} fell below 0.01.

Figure 3 shows a typical result, for $N = 3$ a vector x of 2000 dimensions, and 100 permutations. The system quickly converges to an x in which a single X_i dominates. We have reproduced these results for larger values of N, using vectors

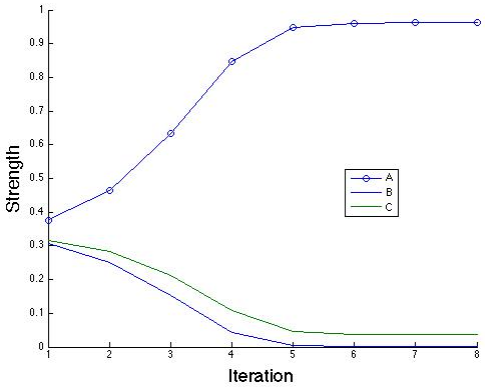

Figure 3: Winner-takes all in VSA implementation

with more realistically large dimensions and more permutations (connectivity). It is important to emphasize that *all representations and operations in this and the next experiment are fully distributed*. The figure was produced by serial extraction of the strengths of the three principal vectors of this system, but this was done only for purposes of illustration. There is nothing in the system that requires the intervention of a localist "homunculus" at any stage.

Application to Analogical Mapping

Analogical mapping has long been a focus of efforts in cognitive modeling. There are several successful connectionist cognitive models of analogy (Holyoak & Thagard, 1989; Hummel & Holyoak, 1997; Eliasmith & Thagard, 2001). These models vary in their theoretical emphases and the details of their connectionist implementations. However, they all share a problem in the scalability of the amount of computational resources or effort required to construct the connectionist mapping network. We contend that this is a consequence of using localist connectionist representations or using distributed representations in a localist manner.

To address this issue, we have recently developed a model that treats analogical mapping as a special case of graph isomorphism; that is, the solution of finding an optimal mapping between two structures (graphs) consisting of individuals (vertices) and their relations (edges). For example, in the simple graphs in Figure 4, the maximal isomorphism is {A=P, B=Q, C=R, D=S} or {A=P, B=Q, C=S, D=R}. Our model builds on the work of Pelillo (1999), who uses replicator dynamics (originally developed in evolutionary game theory) to solve the problem with a localist representation. In Pelillo's solution, iterated multiplication of a localist edge-consistency matrix w by a localist vertex-mapping vector x produces a localist "payoff" vector π expressing the quality of the solution. Elementwise multiplication of x with π produces an updated x representing an improved set of vertex mappings. This elementwise multiplication can be construed as a multiset intersection.

In our VSA implementation of this model, all entities (vertices, edges, and w, x, and π) are represented as high-dimensional MAP vectors. Vertex mappings in x are represented as the sums of the corresponding pairwise edge-mapping products (A*P + A*Q + ... + C*S), and the winner-takes-all intersection circuit of Figure 2 supports competition among mutually inconsistent mappings (C=R, D=S vs. C=S, D=R), without decomposing x into its constituent edge mappings. As shown in Figure 5, the VSA implementation can exhibit dynamic convergence to a solution in a way that is qualitatively similar to the localist implementation. Here, each curve corresponds to the level of support for a specific node mapping; *e.g.*, AP represents the support for the correspondence between nodes A and P. Notice that the components corresponding to the correct node mappings compete with and suppress the components corresponding to incorrect node mappings. As in the previous experiment, the convergence takes place without decomposition into localist components, the figure being a localist presentation for illustration only.

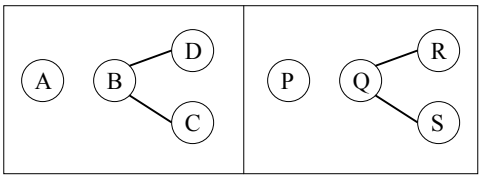

Figure 4: A simple graph isomorphism problem

Conclusion

We have presented a fully distributed connectionist architecture supporting lateral inhibition / winner-takes all competition. The architecture uses a neurally plausible permutation circuit to support a multiset intersection operation without decomposing the summed vector into its constituent items. This approach compares favorably with a localist approach when applied to the task of analogical mapping. Our results thus challenge the commonly-accepted view that lateral inhibition calls for localist representation. More profoundly, our model points toward a neural implementation where more complex representations do not require more complex or dynamically rewired hardware, a long-standing goal of connectionist cognitive modeling.

Acknowledgments

The authors thank two anonymous reviewers for the most helpful comments.

Software Download

Matlab code implementing the experiments described in this paper can be downloaded from **`tinyurl.com/lidemo`**.

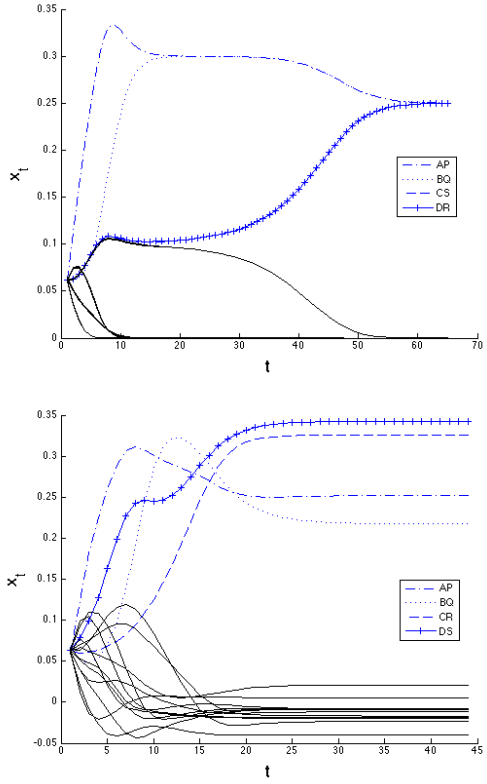

Figure 5: Convergence of localist (top) and VSA (bottom) implementations of graph isomorphism.

References

Eliasmith, C. (2005). Cognition with neurons: A large-scale, biologically realistic model of the Wason task. In G. Bara, L. Barsalou, & M. Bucciarelli (Eds.), *Proceedings of the 27 th annual meeting of the cognitive science society.*

Eliasmith, C., & Thagard, P. (2001). Integrating structure and meaning: A distributed model of analogical mapping. *Cognitive Science*, *25*, 245-286.

Fodor, J., & Pylyshyn, Z. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, *28*, 3-71.

Gayler, R. (1998). Multiplicative binding, representation operators, and analogy,. In K. Holyoak, D. Gentner, & B. Kokinov (Eds.), *Advances in analogy research: Integration of theory and data from the cognitive, computational, and neural sciences* (p. 405). Sofia, Bulgaria: New Bulgarian University.

Hammerton, J. A. (1998). Holistic computation: Reconstructing a muddled concept. *Connection Science*, *10*, 10-1.

Hecht-Nielsen, R. (1994). Context vectors: general purpose approximate meaning representations self-organized from raw data. In J. Zurada, R. M. II, & C. Robinson (Eds.), *Computational intelligence: Imitating life* (p. 4356). IEEE Press.

Holyoak, J., & Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science*, *13*, 295-355.

Hummel, J., & Holyoak, K. (1997). Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review*, *104*, 427-466.

Jackendoff, R. (2002). *Foundations of language: Brain, meaning, grammar, evolution*. Oxford University Press.

Kanerva, P. (1994). The binary spatter code for encoding concepts at many levels. In M. Marinaro & P. Morasso (Eds.), *Icann '94: Proceedings of international conference on artificial neural networks* (Vol. 1, p. 226-229). London: Springer-Verlag.

Kanerva, P. (in press). Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*.

Levy, S. D. (to appear). Becoming recursive: Toward a computational neuroscience account of recursion in language and thought. In H. van der Hulst (Ed.), *Recursion in human language.* The Hague: Mouton de Gruyter.

Olson, A. C., & Humphreys, G. W. (1997). Connectionist models of neuropsychological disorders. *Trends in Cognitive Sciences*, *1*(6), 222 - 228.

Page, M. (2000). Connectionist modelling in psychology: A localist manifesto. *Behavioral and Brain Sciences*, *23*, 443512.

Pelillo, M. (1999). Replicator equations, maximal cliques, and graph isomorphism. *Neural Computation*, *11*, 1933-1955.

Phaf, R. H., & Wolters, G. (2000). A competitive manifesto. *Behavioral and Brain Sciences*, *23*, 487-488.

Plate, T. A. (2003). *Holographic reduced representation: Distributed representation for cognitive science*. CSLI Publications.

Pollack, J. (1990). Recursive distributed representations. *Artifical Intelligence*, *36*, 77-105.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, *46*, 159-216.

Stewart, T., & Eliasmith, C. (forthcoming). Oxford handbook of compositionality. In W. Hinzen, E. Machery, & M. Werning (Eds.), *Compositionality and biologically plausible models.* Oxford: Oxford University Press.

van der Velde, F. (2006). Neural blackboard architectures of combinatorial structures in cognition. *Behavioral and Brain Sciences*, *29*, 1-72.