2018 17th IEEE International Conference on Machine Learning and Applications

Learning in a Continuous-Valued Attractor Network

Baram Sosis
Dept. of Computer Science
University of Maryland
College Park, Maryland USA
bsosis272@gmail.com

Garrett E. Katz
Dept. of Electrical Engineering and Computer Science
Syracuse University
Syracuse, New York USA
gkatz01@syr.edu

James A. Reggia
Dept. of Computer Science and UMIACS
University of Maryland
College Park, Maryland USA
reggia@cs.umd.edu

***Abstract*—Learning a set of patterns in a content-addressable memory is an important aspect of many neurocomputational systems. Historically, this has often been done using Hebbian learning with attractor neural networks such as the standard discrete-valued Hopfield model. However, such systems are currently severely limited in terms of their memory capacity: as an increasing number of patterns are stored as fixed points, spurious attractors ("false memories") are increasingly created and compromise the network's functionality. Here we adopt a new method for identifying the fixed points (both stored and false memory patterns) learned by attractor networks in general, applying it to the special case of a continuous-valued analogue of the standard Hopfield model. We use computational experiments to show that this continuous-valued model functions as a content-addressable memory, characterizing its ability to learn training examples effectively and to store them at energy minima having substantial basins of attraction. We find that the new fixed point locator method not only identifies learned memories, but also many of the spurious attractors that occur. These results are a step towards systematically characterizing what is learned by attractor networks and may lead to more effective learning by allowing the use of techniques such as selective application of anti-Hebbian unlearning of spurious memories.**

***Index Terms*—Hebbian learning, attractor neural networks, directional fibers**

I. Introduction

Neural networks have become an increasingly prominent part of machine learning over the last decade, led by substantial advances in the successful application of a variety of deep learning methods for pattern classification, natural language processing, game playing, and other tasks [1]. Deep convolutional networks based on feedforward architectures provide a good example of this, as manifested by their major successes in image processing competitions. Deep learning is also interesting in the sense that it discovers useful hierarchical application-specific features/concepts from its training data [2]. However, there are some substantial limitations of contemporary deep learning methods. For example, their most prominent successes have come from supervised learning (gradient descent methods based on error backpropagation) requiring numerous training epochs using extremely large data sets. The need for large amounts of data is problematic if one only has access to very limited data and/or if one wants to learn very rapidly from just a few examples, or even a single example. Such considerations motivate the exploration of alternative neural network approaches to machine learning systems, as we do here.

One potential alternative neurocomputational approach is the use of attractor neural networks that are based on Hebbian learning, not error-driven gradient descent, via a single pass through the data. Attractor networks in general are highly recurrent architectures that can advantageously be viewed as dynamical systems and can thus be characterized by their attractor states. Here we restrict our attention to attractor networks with symmetric weights that can serve as content addressable memories. A simple and well-known example of such models is the discrete-valued Hopfield model that is often used in machine learning textbooks as an illustration, e.g., [3]–[5]. Discrete Hopfield networks have symmetric weight matrices and associated energy functions (Lyapanuv functions, Hamiltonians) that, in the context of asynchronous node updating, ensure that the network evolves to a fixed point activation state when started in any arbitrary state. One-step Hebbian learning (i.e., learning a specific memory pattern after seeing it only once) tends to store given training data as fixed point attractors in such a network, which can then be retrieved by later initializing the network to similar partial or noisy activation states and allowing the network to evolve to the nearest attractor state (presumably a learned memory pattern). Unlike many other recurrent neural network learning methods that are based on gradient descent (e.g., LSTM), learning is extremely fast because it is both simple and involves only a single pass through the data ("one step").

Hopfield networks continue to be used in applications today and many variants/extensions have been and continue to be investigated, e.g., [6], [7]. For example, an extended version of basic Hopfield networks was used recently in a cognitive neural architecture that learned to play a simple card game [8]. The Hopfield networks that were components of this model served both as a working memory that learned environmental states, and as a procedural memory that learned simple action sequences to use during problem solving. However, content addressable memories like this also face significant limitations, one of which is their limited memory capacity. Only a relatively small number of memory patterns (training data) can be learned as fixed points and then retrieved reliably, in large measure due to the occurrence of spurious memories (spurious fixed point attractor states). These "false memo-

978-1-5386-6805-4/18/$31.00 ©2018 IEEE
DOI 10.1109/ICMLA.2018.00048

ries" arise due to interference that occurs between weight changes being made in storing different target patterns in memory during learning. Accordingly, a critical question is whether one can identify and characterize a network's learned memories, both intended and spurious. This would not only allow us to better understand what a network has learned, but even more importantly, to also study methods for potentially selectively eliminating the spurious memories stored during learning (e.g., anti-Hebbian unlearning). Substantial past work has acknowledged the importance of identifying fixed points of neural networks in terms of understanding the mechanisms that underlie their performance (e.g., [9]). Typically, in examining this issue, most past work has located stored attractor states in a trained network by initializing its activity state to a random sample of activity patterns (referred to as "random starts" in the following) and collecting the attractor states that were subsequently reached by the network. It is currently unclear how effective this approach is in general.

Recently a new method was introduced for locating fixed points of recurrent neural networks via the use of directional fibers [10], [11]. This approach systematically follows a path through the activation space of arbitrary recurrent neural networks that passes through many of the network's fixed point states. Here we examine the hypothesis that this method provides an effective alternative to random starts in locating the learned attractor states in Hopfield-like content addressable memories. Our method requires us to convert the original discrete state Hopfield model into a continuous version, and we show how to do that here. We then ask and answer several questions. Does the continuous version of the model function reliably as a content addressable memory? Can searching the trained model's activity space effectively locate its fixed points? If so, how many of these fixed points correspond to learned memories versus spurious memories, and how large are the basins of attraction in either case? How does the network's energy vary as a directional fiber is traversed? The results that we present below begin to answer these questions.

II. Methods

The update equation for the original discrete Hopfield model [3]–[5] is given by:

$$v_i(t+1) = \mathrm{sgn}(A_i(t)) \tag{1}$$

where

$$A_i(t) = \sum_{j=1}^{N} w_{ij} v_j(t), \tag{2}$$

$v_i(t)$ is the i-th element of an N-dimensional state vector at time t, and w_{ij} is the i,j-th element of an N-by-N weight matrix. (We will typically omit the t unless needed for clarification.) It can also be expressed in a non-standard way as follows:

$$\Delta v_i = \mathrm{sgn}(A_i) - v_i \tag{3}$$

The weight matrix is calculated using the following Hebbian learning rule:

$$w_{ij} = \begin{cases} 0 & i = j \\ \frac{1}{N} \sum_{k=1}^{d} x_i^k x_j^k & \text{otherwise} \end{cases} \tag{4}$$

where x_i^k is the i-th entry of the k-th training data vector (out of d training vectors), drawn from $\{-1, 1\}^N$.

We consider here a continuous-valued attractor network as an analogue to the standard Hopfield network. The motivation behind the development of this model is to create a system analogous to Hopfield networks that uses a differentiable update rule so that the directional fiber-based solver and other solvers requiring differentiable update rules can be applied to it. We employed a model consisting of a set of fully connected nodes like the standard Hopfield network. This model uses the discrete-time update equation

$$\Delta v_i = \tanh(A_i) - v_i \tag{5}$$

where A_i is modified to incorporate a gain term γ:

$$A_i = \gamma \sum_{j=1}^{N} w_{ij} v_j \tag{6}$$

We chose this update rule as a direct analogue to (3). The weight matrix for this model is generated in the same way as that of the Hopfield model, using (4). Unlike those of the Hopfield model, the fixed points of the continuous-valued model do not lie at the corners of the unit hypercube due to the contractive effects of the $\tanh$ function. To accommodate this, we take their entry-wise sign when comparing them to fixed points of the standard Hopfield model. In addition, gain values larger than 1 are typically needed to ensure the model has a significant number of nontrivial (i.e. besides the zero vector) fixed points.

The update equation for the continuous-valued model can be stated alternatively as

$$v_i(t+1) = f_i(\mathbf{v}(t)) \tag{7}$$

where

$$f_i(\mathbf{v}(t)) = \tanh(A_i(t)). \tag{8}$$

For convenience we will also occasionally make use of the following function:

$$g_i(\mathbf{u}, \mathbf{v}) = \gamma \sum_{j=1}^{N} w_{ij} u_i v_j \tag{9}$$

The continuous-valued model can be updated synchronously (by applying (5) to all v_i simultaneously) or asynchronously (by applying it to each v_i in turn). Our analysis of the asynchronous update method only dealt with sequential updating in a fixed order (i.e. entries are updated in the fixed order $v_1, v_2, \ldots, v_N$), but in principle asynchronous updating could use a randomized update order.

Both update methods have Lyapunov functions, in the synchronous case given by

$$V(\mathbf{v}(t)) = \sum_{i=1}^{N} [g_i(\mathbf{v}(t+1), \mathbf{v}(t)) - \ln(\cosh(A_i(t))) - \ln(\cosh(A_i(t+1)))] \quad (10)$$

and in the asynchronous case (assuming a fixed update order as described above) given by

$$V(\mathbf{v}) = \sum_{i=1}^{N} [0.5 \ln(1 - v_i^2) + v_i \operatorname{arctanh}(v_i) - 0.5 g_i(\mathbf{v}, \mathbf{v})] \quad (11)$$

which are special cases of the results in [12]. See Appendices A and B for derivations. We experimentally verified that these functions are monotonically decreasing along a random sample of activation trajectories.

The directional fiber-based solver requires both the update equation (7) as well as its Jacobian in order to operate on the continuous-valued model. The Jacobian is an N-by-N matrix where the i, j-th entry is given by $\partial f_i / \partial v_j$; in our case it differs depending on which update method is used. It is given by

$$\frac{\partial f_i}{\partial v_j} = \gamma w_{ij} \operatorname{sech}^2(A_i) \quad (12)$$

in the synchronous case, as can be seen by differentiating (8). To describe the asynchronous case, let $v_j(t+i/N)$ indicate the value of v_j after updating the first i entries of $\mathbf{v}(t)$. Then the fixed-order asynchronous updating procedure can be written as follows:

$$v_j\left(t + \frac{i}{N}\right) = \begin{cases} f_j(\mathbf{v}(t + \frac{i-1}{N})) & \text{if } i = j \\ v_j(t + \frac{i-1}{N}) & \text{otherwise} \end{cases} \quad (13)$$

Partial derivatives with respect to the previous time-step can then be computed using the recursive formula:

$$\frac{\partial v_j(t + \frac{i}{N})}{\partial v_k(t)} = \sum_{l=1}^{N} \frac{\partial v_j(t + \frac{i}{N})}{\partial v_l(t + \frac{i-1}{N})} \frac{\partial v_l(t + \frac{i-1}{N})}{\partial v_k(t)} \quad (14)$$

where

$$\frac{\partial v_j(t + \frac{i}{N})}{\partial v_k(t + \frac{i-1}{N})} = \begin{cases} \gamma w_{jk} \operatorname{sech}^2(A_j(t + \frac{i-1}{N})) & \text{if } i = j \\ \delta_{jk} & \text{otherwise} \end{cases} \quad (15)$$

and δ_{jk} is the Kronecker delta. Since $\partial f_i / \partial v_j$ is equivalent to $\partial v_i(t+1)/\partial v_j(t) = \partial v_i(t+N/N)/\partial v_j(t)$ using the notation for asynchronous updating, the Jacobian is described by:

$$\frac{\partial v_j(t+1)}{\partial v_k(t)} = \sum_{l=1}^{N} \frac{\partial v_j(t + \frac{N}{N})}{\partial v_l(t + \frac{N-1}{N})} \frac{\partial v_l(t + \frac{N-1}{N})}{\partial v_k(t)} \quad (16)$$

which can be computed recursively using (14) and (15). These Jacobians can be used to determine whether a fixed point is stable: a fixed point is stable when the eigenvalues of the Jacobian at that point all have magnitude less than 1.

In the following section we describe the experiments we conducted to investigate the properties of this model.[1] We verified that it functions as a content-addressable memory by perturbing learned memories, running the network to convergence, and measuring the distance between the result and the original learned memories. We then quantified the performance of the directional fiber-based solver on this model as a function of the gain parameter, and compared it to the baseline solver using random starts described in [10]. Fixed points were classified on the basis of stability, whether they corresponded to learned memories, and whether they were fixed points of the original discrete-valued Hopfield network trained on the same data. Finally, we investigated how the energy varies along a directional fiber and between fixed points.

III. Results

A. Overview

The experiments below typically used 100-unit networks with $\gamma = 10$ and 3 training vectors (randomly drawn from $\{-1, 1\}^N$) unless noted otherwise. These experiments all used synchronous updating; for brevity we omit results using asynchronous updating because they do not differ significantly from the results reported here.

B. Content-Addressability

In order to verify that the continuous-valued model can serve as a content-addressable memory, we conducted several experiments measuring the model's ability to retrieve memories stored in it using Hebbian learning.

Fig. 1 shows the results of initializing a network (100 units, $\gamma = 10$, and a variable number of random training vectors) at a learned memory with a variable number of the entries (indicated along the abscissa) randomly negated. We measured the Hamming distance (the number of bits that differ between two binary strings) between the learned memory and the result after running the network to convergence in order to estimate the size of the basins of attraction in the network. Each data point represents an average over 50 different networks, over all memories learned by that network, and over 50 random perturbations of each memory. Networks with 10 or fewer learned memories can function with minimal error even when 20 entries are flipped. The learned memories are thus typically stable fixed points with wide basins of attraction under this model.

Like the original Hopfield model, this model has spurious fixed points. However, fixed points that do not correspond directly to the learned memories have much smaller basins of attraction. In Fig. 2 we characterize the individual fixed points of a particular network (100 units, $\gamma = 10$, and 3 random training vectors; the fixed points were found using the directional fiber-based solver). The network was initialized at

[1] Code for reproducing the experiments described here is freely available at https://github.com/bsosis/cont-hopnet.

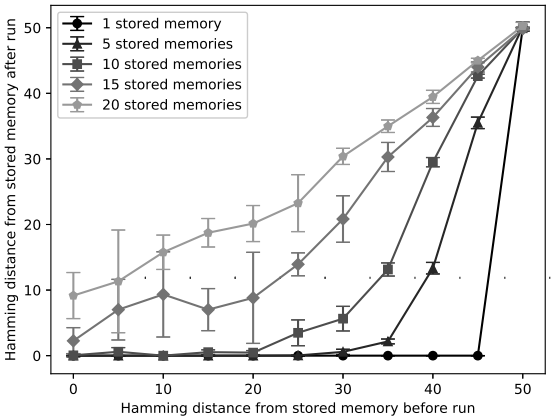


Fig. 1. Average distances from learned memories after networks are perturbed from those memories and run to convergence. Error bars show one standard deviation.

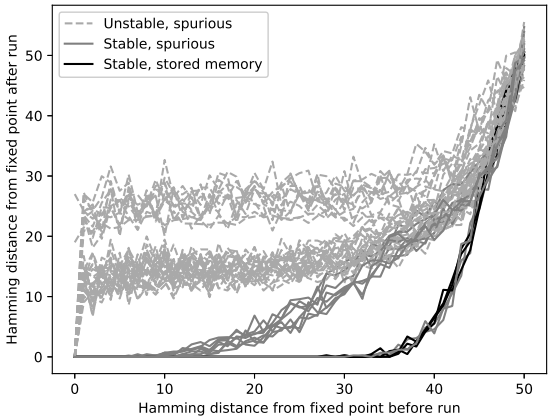


Fig. 2. Distances from fixed points after a particular network is perturbed from those fixed points and run to convergence.

each fixed point with some number of the entries (indicated along the abscissa) randomly negated. We then measured the Hamming distance between the fixed point and the result after running the network to convergence. Each data point represents an average of 100 random perturbations of the fixed point.

We classified the fixed points found as spurious or not by checking whether the entry-wise sign of a given fixed point matched one of the learned training vectors. (Note that like in the original discrete-valued Hopfield model, every fixed point of the continuous-valued model besides the zero vector has a complementary fixed point with every entry negated. The complements of learned training vectors are marked here as spurious.) This network has many unstable spurious fixed points (dashed gray lines), and no unstable fixed points matching any of the learned vectors. It also has several stable spurious fixed points (solid gray lines), but the basins of attraction of these points seem to be relatively small: the network no longer converges back to the fixed point when around 10 entries are flipped. In contrast, stable fixed points corresponding to learned memories (black lines) or the complements of memories (marked as spurious but with curves interwoven with those of the learned memories) can be accurately retrieved even when 30 entries are flipped.

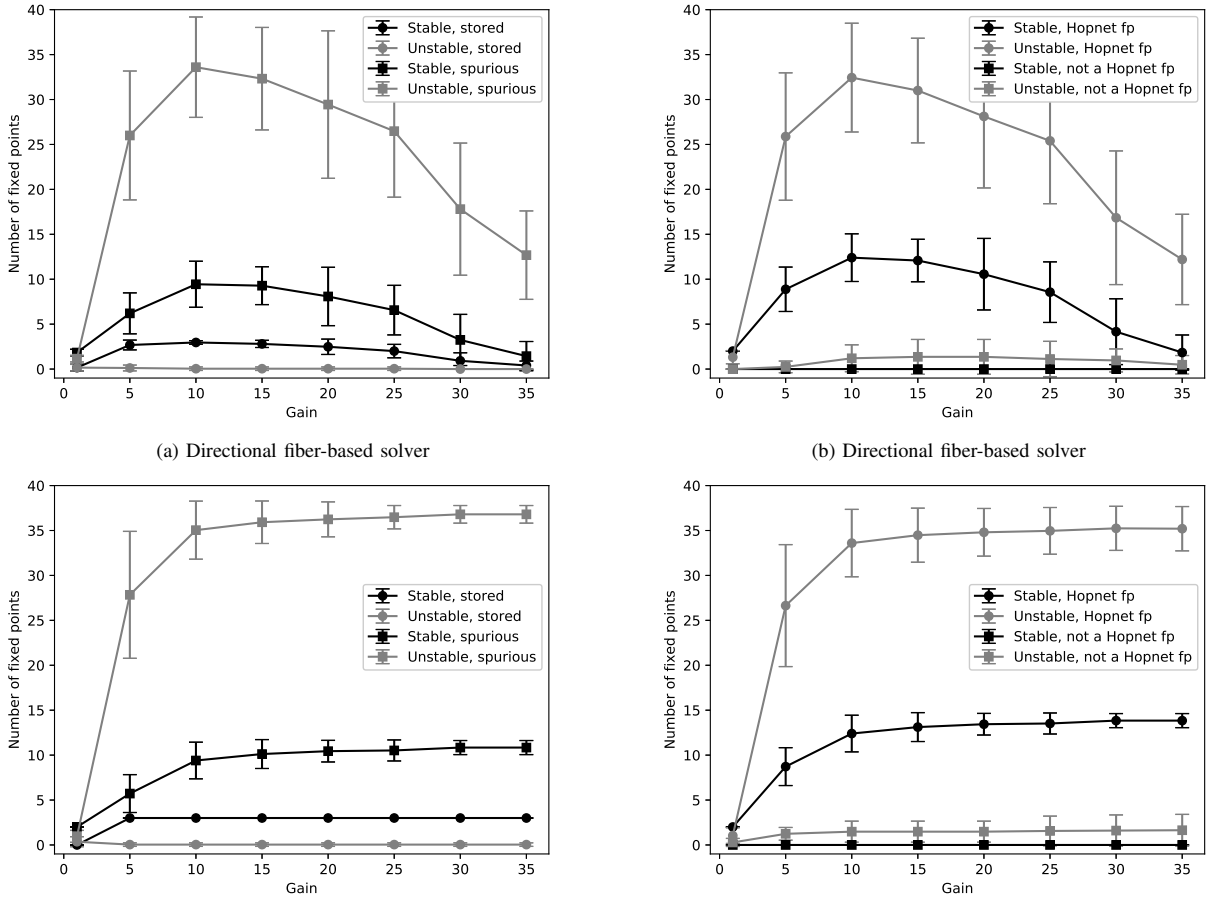
C. Applying the Solvers

After verifying that the continuous-valued model serves as a content-addressable memory, we then investigated how well the directional fiber-based solver and the baseline random-starts solver can find the fixed points of the model. We classified the fixed points found in three ways: whether they are stable or unstable, whether they are spurious or not, and whether they match a fixed point of the equivalent Hopfield model (that is, the discrete-valued Hopfield model using the same weight matrix). Stable and unstable fixed points were identified by examining the Jacobian. Spurious fixed points and points corresponding to learned training vectors were identified by comparing the entry-wise sign of the fixed points to the training vectors. Finally, fixed points were classified as matching a fixed point of the equivalent Hopfield model if their entry-wise sign was a fixed point of the equivalent discrete-valued model.

Results are shown in Fig. 3. Experiments were run using 100-unit networks with a variable gain and 3 training vectors; results were averaged over 25 different networks. Figs. 3a and 3b show the results for the directional fiber-based model, while Figs. 3c and 3d show the results for the baseline solver. Figs. 3a and 3c classify the fixed points on the basis of stability and whether they correspond to learned training vectors or are spurious; Figs. 3b and 3d classify them on the basis of stability and whether they are fixed points under the equivalent discrete-valued Hopfield model. Unlike the directional fiber-based solver, which terminates once it has traversed a fiber long enough that no more fixed points will be found along it [10], the baseline solver can be run indefinitely. We therefore timed the directional fiber-based solver and ran the baseline solver until the same amount of time had elapsed.

We found that the gain parameter has a large impact on the number of fixed points found by the directional fiber-based solver, with a peak at around $\gamma = 10$. When a gain of 10 is used the directional fiber-based solver often finds all three learned training vectors (Fig. 3a). In contrast, the baseline solver is relatively insensitive to the gain, and seems to find all three training vectors more consistently (Fig. 3c). In both cases nearly all fixed points found match fixed points of the equivalent Hopfield model (Figs. 3b and 3d). This lends further justification to the use of the continuous-valued model as an analogue for the Hopfield model.

We also attempted to estimate what fraction of all fixed points were found by the directional fiber-based solver in a particular run. To do this, we repeatedly ran the solver with different directional fibers on the same network (100 units, $\gamma = 10$, and either 3 or 10 random training data vectors); after each run we computed the union of the sets of fixed points



(a) Directional fiber-based solver

(b) Directional fiber-based solver

(c) Baseline solver

(d) Baseline solver

Fig. 3. Number of fixed points found by (a), (b) the directional fiber-based solver and (c), (d) the baseline solver in the continuous-valued model with variable gain, classified on the basis of stability and whether the fixed points (a), (c) match learned training vectors and (b), (d) match fixed points under the equivalent discrete-valued Hopfield model. "Hopfield network fixed point" is abbreviated "Hopnet fp" in the legend. Error bars show one standard deviation.

found up to that point. Fig. 4 shows the results averaged over 10 different networks. While the directional fiber-based solver is able to find nearly all fixed points with only one run when there are relatively few fixed points (as in Fig. 4a), it needs many runs when there are more fixed points, such as when 10 training vectors are used (as in Fig. 4b).

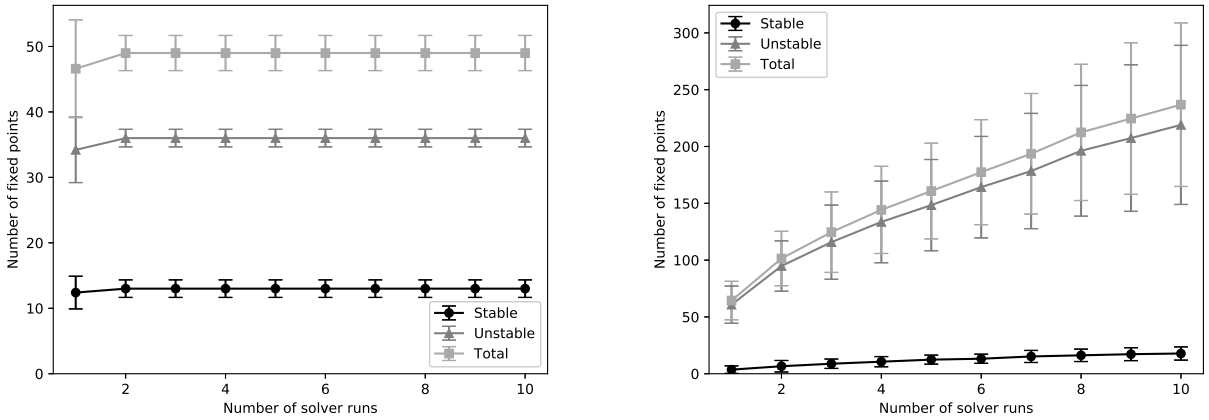
D. Energy

Finally, we ran several experiments to understand how the energy function varies as the directional fiber-based solver traverses a fiber. Fig. 5 shows the energy along a particular fiber traversed by the solver; we used a 100-unit network with $\gamma = 10$ and 3 training vectors. Fixed points are marked with dashed lines; spurious, stable fixed points are marked with squares while non-spurious, stable fixed points are marked with black circles. Since the fiber does not follow the model's update equation, the energy is not necessarily decreasing along it and therefore local minima do not necessarily imply fixed points. However, every fixed point found by the directional fiber-based solver is a critical point of the energy function, with stable fixed points lying in local energy minima. Unstable fixed points appear to lie in energy maxima but further work is needed to rule out the possibility of saddle points.

Fig. 6 shows the fixed points of 25 different networks (100 units, $\gamma = 10$, and 3 learned training vectors; the appearance of fewer than 25 networks is due to overlap) sorted by energy. Stable fixed points typically have a lower energy than unstable fixed points, and stable fixed points corresponding to learned memories (or their complements, since every fixed point besides the zero vector has a complementary fixed point with every entry negated) nearly always have the lowest energy.

IV. Discussion

We have shown that our continuous-valued model can effectively serve as a content-addressable memory, and that the baseline solver and often the directional fiber-based solver can effectively find its fixed points. Moreover, nearly all of these learned attractor states lie in the same orthant as a fixed point under the original discrete-valued Hopfield model, allowing its use as an analogue for the Hopfield model. This



(a) 3 training vectors

(b) 10 training vectors

Fig. 4. Number of unique fixed points found by the directional fiber-based solver when run on the same network with different directional fibers. The networks were trained on either (a) 3 or (b) 10 learned memories. Note the different vertical scales. Error bars show one standard deviation.

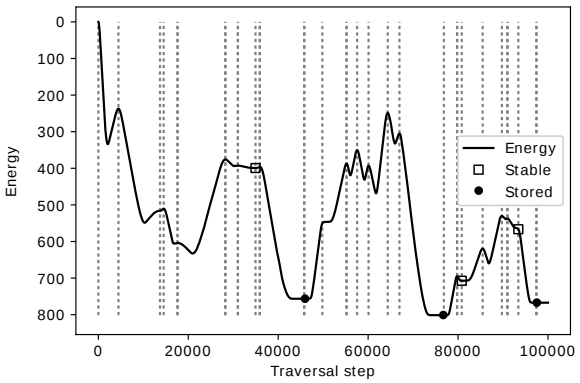


Fig. 5. Plot of the energy function along the fiber followed by a particular run of the directional fiber-based solver. Dashed vertical lines indicate fixed points, white squares indicate spurious stable fixed points, and black circles indicate non-spurious stable fixed points (learned memories).

in turn allows the fixed points of Hopfield networks to be studied by applying these solvers (or other solvers that require a differentiable update rule) to the equivalent continuous-valued model. This gives researchers investigating the learned memories in attractor networks a significant new tool in their toolbox. The energy experiments presented above are one example of the type of studies this model enables.

It appears from our experiments (Fig. 3) that the baseline solver outperforms the directional fiber-based solver on this model when the gain used is high (above 15, gain values that are generally not used in practice). This runs contrary to [10], in which the authors show that the directional fiber-based solver often outperforms the baseline solver. One potential difference that may explain the discrepancy is the learning procedures used. While we used Hebbian learning via (4), in [10] the authors used an asymmetric matrix with random weights. Perhaps the most significant consequence of this is that their procedure results in models with far more fixed points than those considered here. If the number of fixed points has a large impact on the relative performance of the two solvers, then it may be best to view the algorithms as complementary, with each performing best on different problem sizes and types. More research is needed to investigate this possibility, but elucidating the relationship between the solvers may help further our understanding of the memory capacity of attractor networks so they can be used more effectively as learning systems.

In summary, our results reported here are one step towards characterizing what is learned by attractor networks. In terms of future work, we conjecture that with increased memory load (increasing numbers of specific patterns to learn) the fiber-based solver will increasingly find attractors that are missed by the baseline solver, just as was observed with other classes of neural networks. A key next step will be to use such information to improve the learning ability of our continuous-valued model by selectively erasing learned spurious attractor states using methods such as anti-Hebbian unlearning.

APPENDIX A
DERIVATION OF (10)

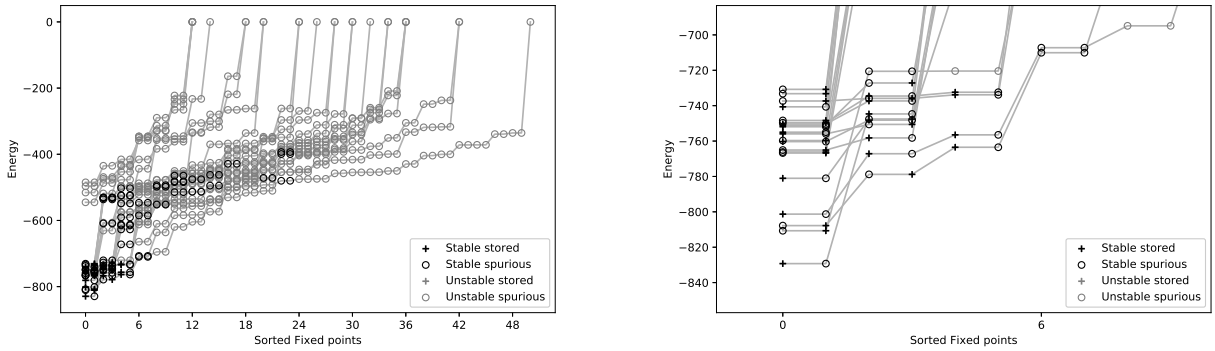
In [12] the authors give the general form of the Lyapunov function obeyed by a continuous-valued, discrete time dynamical system with a synchronous update procedure of the form:

$$v_i(t+1) = h\left(B_i\right) \tag{17}$$

where

$$B_i = \sum_{j=1}^{N} w_{ij} v_j(t) - b_i \tag{18}$$

under the constraint that $w_{ij} = w_{ji}$; in our case $b_i = 0$ for all i. Also note that since we use γw_{ij} in every case where w_{ij} is used, γ can be incorporated into B_i and we can simply use



(a) (b)

Fig. 6. The fixed points found by the directional fiber-based solver on several networks, sorted by energy. Stable fixed points and fixed points corresponding to learned memories are marked on the graph. (b) shows an expanded view of the lower left corner of (a).

A_i as defined in (6) instead. Using our notation, the general form of the Lyapunov function is as follows:

$$V(\mathbf{v}(t)) = \sum_{i=1}^{N} [g_i(\mathbf{v}(t), \mathbf{v}(t-1)) - \int_c^{A_i(t-1)} h(v)\, dv - \int_c^{A_i(t)} h(v)\, dv] \tag{19}$$

where c is an arbitrary constant in an interval over which f is strictly increasing. In our case $h(v) = \tanh(v)$ with an antiderivative $\ln(\cosh(v))$, so by picking $c = 0$ and observing that $\ln(\cosh(0)) = 0$, we have the following result:

$$V(\mathbf{v}(t)) = \sum_{i=1}^{N} [g_i(\mathbf{v}(t), \mathbf{v}(t-1)) - \ln(\cosh(A_i(t-1))) - \ln(\cosh(A_i(t)))] \tag{20}$$

Since $\mathbf{v}(t-1)$ is not accessible on the first iteration of the model, we replaced the $\mathbf{v}(t)$ and $\mathbf{v}(t-1)$ terms in (20) with $\mathbf{v}(t+1)$ and $\mathbf{v}(t)$, respectively, giving (10). This is still a Lyapunov function because $V(\mathbf{v}(t))$ is non-increasing for any t, regardless of how t is shifted.

Appendix B
Derivation of (11)

In [12] the authors give the general form of the Lyapunov function obeyed by a continuous-valued, discrete time dynamical system operating under an asynchronous update procedure with the same constraints as in Appendix A, and the additional constraint that W has a nonnegative diagonal. Using our notation, the general form is as follows:

$$V(\mathbf{v}) = -\frac{1}{2} \sum_{i=1}^{N} [g_i(\mathbf{v}, \mathbf{v}) + b_i v_i + \int_0^{v_i} h^{-1}(v)\, dv] \tag{21}$$

In our case $b_i = 0$ and $h^{-1}(v) = \operatorname{arctanh}(v)$ with an antiderivative $0.5\ln(1-v^2) + v\operatorname{arctanh}(v)$; since this antiderivative is 0 when evaluated at 0, we get the result in (11).

Acknowledgment

This work was supported in part by DARPA Award HR00111890044.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge: MIT press, 2016, vol. 1.

[3] S. S. Haykin, *Neural networks and learning machines*. Upper Saddle River, NJ: Pearson, 2009.

[4] S. Marsland, *Machine learning: an algorithmic perspective*. CRC press, 2015.

[5] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[6] C. Gorman, A. Robins, and A. Knott, "Hopfield networks as a model of prototype-based category learning: A method to distinguish trained, spurious, and prototypical attractors," *Neural Networks*, vol. 91, pp. 76–84, 2017.

[7] D. Nowicki and H. Siegelmann, "Flexible kernel memory," *PLOS ONE*, vol. 5, pp. 1–18, 06 2010. [Online]. Available: https://doi.org/10.1371/journal.pone.0010955

[8] J. Sylvester and J. Reggia, "Engineering neural systems for high-level problem solving," *Neural Networks*, vol. 79, pp. 37–52, 2016.

[9] D. Sussillo and O. Barak, "Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks," *Neural Computation*, vol. 25, no. 3, pp. 626–649, 2013.

[10] G. E. Katz and J. A. Reggia, "Using directional fibers to locate fixed points of recurrent neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 9, pp. 3636–46, 2018. [Online]. Available: https://doi.org/10.1109/TNNLS.2017.2733544

[11] ——, "Applications of directional fibers to fixed point location and non-convex optimization," in *Proc. 16th Annual Conf. on Scientific Computing*. Las Vegas: CSREA Press, August 2018, pp. 140–146.

[12] F. Fogelman Soulié, C. Mejia, E. Goles, and S. Martinez, "Energy functions in neural networks with continuous local functions," *Complex Systems*, vol. 3, pp. 269–293, 1989.