Efficient and optimal Little-Hopfield auto-associative memory storage using minimum probability flow

Christopher Hillar
Redwood Center for Theoretical Neuroscience
University of California, Berkeley
Berkeley, CA 94720
chillar@msri.org

Jascha Sohl-Dickstein
Redwood Center for Theoretical Neuroscience
University of California, Berkeley
Berkeley, CA 94720
jascha.sohldickstein@gmail.com

Kilian Koepsell
Redwood Center for Theoretical Neuroscience
University of California, Berkeley
Berkeley, CA 94720
kilian@berkeley.edu

Abstract

We present an algorithm to store binary memories in a Little-Hopfield neural network using minimum probability flow, a recent technique to fit parameters in energy-based probabilistic models. In the case of memories without noise, our algorithm provably achieves optimal pattern storage (which we show is at least one pattern per neuron) and outperforms classical methods both in speed and memory recovery. Moreover, when trained on noisy or corrupted versions of a fixed set of binary patterns, our algorithm finds networks which correctly store the originals. We also demonstrate this finding visually with the unsupervised storage and clean-up of large binary fingerprint images from significantly corrupted samples.\(^1\)

Introduction. In 1982, motivated by neural modeling work of [1] and the Ising spin glass model from statistical physics [2], Hopfield introduced a method for the storage and retrieval of binary patterns in an auto-associative neural-network [3]. Even today, this model and its various extensions [4, 5] provide a plausible mechanism for memory formation in the brain. However, existing techniques for training Little-Hopfield networks suffer either from limited pattern capacity or excessive training time, and they exhibit poor performance when trained on unlabeled, corrupted memories.

Our main theoretical contributions here are the introduction of a tractable and neurally-plausible algorithm for the optimal storage of patterns in a Little-Hopfield network, a proof that the capacity of such a network is at least one pattern per neuron, and a novel local learning rule for training neural networks. Our approach is inspired by minimum probability flow [6], a recent technique for fitting probabilistic models that avoids computations with a partition function, the usually intractable normalization constant of a parameterized probability distribution.

We also present several experimental results. When compared with standard techniques for Little-Hopfield pattern storage, our method is shown to be superior in efficiency and generalization. Another finding is that our algorithm can store many patterns in a Little-Hopfield network from highly corrupted (unlabeled) samples of them. This discovery is also corroborated visually by the storage of 64 × 64 binary images of human fingerprints from highly corrupted versions, as in Fig. 1.

\(^1\)Supported by an NSF All-Institutes Postdoctoral Fellowship administered by the Mathematical Sciences Research Institute through its core grant DMS-0441170 (CH) and by NSF grant IIS-0917342 (KK). This work appeared at the 2012 Neural Information Processing Systems (NIPS) workshop on Discrete Optimization in Machine Learning (DISCML).
Background. A Little-Hopfield network \( \mathcal{H} = (J, \theta) \) on \( n \) nodes \( \{1, \ldots, n\} \) consists of a symmetric weight matrix \( J \in \mathbb{R}^{n \times n} \) with zero diagonal and a threshold vector \( \theta = (\theta_1, \ldots, \theta_n) \top \in \mathbb{R}^n \). We do not allow any row \( J_i \) of the matrix \( J \) to be zero. The possible states of the network are all length \( n \) binary strings \( \{0, 1\}^n \), which we represent as binary column vectors \( x = (x_1, \ldots, x_n) \top \), each \( x_i \in \{0, 1\} \) indicating the state \( x_i \) of node \( i \). Given any state \( x = (x_1, \ldots, x_n) \top \), an (asynchronous) dynamical update of \( x \) consists of replacing \( x_i \) in \( x \) (in consecutive order starting with \( i = 1 \)) with the value

\[
x_i = H(J, x - \theta_i).
\]

Here, \( J_i \) is the \( i \)th row of \( J \) and \( H \) is the Heaviside function given by \( H(r) = 1 \) if \( r > 0 \) and \( H(r) = 0 \) if \( r \leq 0 \).

The energy \( E_x \) of a binary pattern \( x \) in a Little-Hopfield network is defined to be

\[
E_x(J, \theta) := -\frac{1}{2}x \top Jx + \theta \top x - \sum_{i<j} x_i x_j J_{ij} - \sum_{i=1}^{n} \theta_i x_i,
\]

(2)

identical to the energy function for an Ising spin glass. In fact, the dynamics of a Little-Hopfield network can be seen as 0-temperature Gibbs sampling of this energy function. A fundamental property of Little-Hopfield networks is that asynchronous dynamical updates do not increase the energy (2). Thus, after a finite number of updates, each initial state \( x \) converges to a fixed-point \( x^* = (x_1^*, \ldots, x_n^*) \top \) of the dynamics; that is, \( x_i^* = H(J, x^* - \theta_i) \) for each \( i \).

Given a binary pattern \( x \), the neighborhood \( \mathcal{N}(x) \) of \( x \) consists of those binary vectors which are Hamming distance 1 away from \( x \) (i.e., those with exactly one bit different from \( x \)). We say that \( x \) is a strict local minimum if every \( x' \in \mathcal{N}(x) \) has a strictly larger energy:

\[
0 > E_x - E_{x'} = (J_i x - \theta_i) \delta_i,
\]

(3)

where \( \delta_i = 1 - 2x_i \) and \( x_i \) is the bit that differs between \( x \) and \( x' \). It is straightforward to verify that if \( x \) is a strict local minimum, then it is a fixed-point of the dynamics.

A basic problem is to construct Little-Hopfield networks with a given set \( D \) of binary patterns as fixed-points or strict local minima of the energy function (2). Such networks are useful for memory denoising and retrieval since corrupted versions of patterns in \( D \) will converge through the dynamics to the originals. Traditional approaches to this problem consist of iterating over \( D \) a learning rule [7] that updates a network’s weights and thresholds given a training pattern \( x \in D \). For the purposes of
In other words, for any fixed \( a < 1 \), the fraction of all subsets of \( m = an \) patterns that can be made strict local minima (and thus fixed-points) of a Little-Hopfield network with \( n \) nodes converges to
Figure 2: (Left) Shows fraction of patterns made fixed-points of a Little-Hopfield network using OPR (outer-product rule), MPF (minimum probability flow), and PER (perceptron) as a function of the number of randomly generated training patterns \( m \). Here, \( n = 64 \) binary nodes and we have averaged over \( t = 20 \) trials. The slight difference in performance between MPF and PER is due to the extraordinary number of iterations required for PER to achieve perfect storage of patterns near the critical pattern capacity of the Little-Hopfield network. See also Fig. 2. (Right) Shows time (on a log scale) to train a Little-Hopfield network with \( n = 64 \) neurons to store \( m \) patterns using OPR, PER, and MPF (averaged over \( t = 20 \) trials).

We close this section by defining a new learning rule for a neural network. In words, the minimum probability flow learning rule (MPF) takes an input training pattern \( x \) and moves the parameters \((J, \theta)\) a small amount in the direction of steepest descent of the MPF objective function \( K_D(J, \theta) \) with \( D = \{x\} \). These updates for \( J_{ij} \) and \( \theta_i \) take the form (where again, \( \delta = 1 - 2x \)):

\[
\Delta J_{ij} \propto -\delta_i x_j e^{\frac{1}{2}(J_{ij} x - \theta_i)} \delta_i - \delta_j x_i e^{\frac{1}{2}(J_{ij} x - \theta_j)} \delta_j \tag{5}
\]

\[
\Delta \theta_i \propto \delta_i e^{\frac{1}{2}(J_{ij} x - \theta_i)} \delta_i \tag{6}
\]

It is clear from (5) and (6) that MPF is a local learning rule.

**Experimental results.** We performed several experiments comparing standard techniques for fitting Little-Hopfield networks with minimizing the MPF objective function (4). All computations were performed on standard desktop computers, and we used used the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm [18] to minimize (4).

In our first experiment, we compared MPF to the two methods OPR and PER for finding 64-node Little-Hopfield networks storing a given set of patterns \( D \). For each of 20 trials, we used the three techniques to store a randomly generated set of \( m \) binary patterns, where \( m \) ranged from 1 to 120. The results are displayed in Fig. 2 and support the conclusions of Theorem 1 and Theorem 2.

To evaluate the efficiency of our method relative to standard techniques, we compared training time of a 64-node network as in Fig. 2 with the three techniques OPR, MPF, and PER. The resulting computation times are displayed in Fig. 2 on a logarithmic scale. Notice that computation time for MPF and PER significantly increases near the capacity threshold of the Little-Hopfield network.

For our third experiment, we compared the denoising performance of MPF and PER. For each of four values for \( m \) in a 128-node Little-Hopfield network, we determined weights and thresholds for storing all of a set of \( m \) randomly generated binary patterns using both MPF and PER. We then flipped 0 to 64 of the bits in the stored patterns and let the dynamics (1) converge (with weights and thresholds given by MPF and PER), recording if the converged pattern was identical to the original pattern or not. Our results are shown in Fig 3, and they demonstrate the superior corrupted memory retrieval performance of MPF.
Figure 3: Shows fraction of exact pattern recovery for a perfectly trained $n = 128$ Little-Hopfield network using rules PER (figure on the left) and MPF (figure on the right) as a function of bit corruption at start of recovery dynamics for various numbers $m$ of patterns to store. We remark that this figure and the next do not include OPR as its performance was far worse than MPF or PER.

Figure 4: Shows fraction of patterns (shown in red for MPF and blue for PER) and fraction of bits (shown in dotted red for MPF and dotted blue for PER) recalled of trained networks (with $n = 64$ nodes each) as a function of the number of patterns $m$ to be stored. Training patterns were presented repeatedly with 20 bit corruption (i.e., 31% of the bits flipped). (averaged over $t = 13$ trials.)

We also tested the how the efficiency of our algorithm scales with the number of nodes $n$. For varying $n$, we fit $m = n/4$ (randomly generated) patterns in $n$-node Little-Hopfield networks using MPF over 50 trials and examined the training time. Our experiments show that the average training time to fit Little-Hopfield networks with MPF is well-modeled by a polynomial $O(n^{5/2})$.

A surprising final finding in our investigation was that MPF can store patterns from highly corrupted or noisy versions on its own and without supervision. This result is explained in Fig 4. To illustrate the experiment visually, we stored $m = 80$ binary fingerprints in a 4096-node Little-Hopfield network using a large set of training samples which were corrupted by flipping at random 30% of the original bits; see Fig. 1 for more details.

Discussion. We have presented a novel technique for the storage of patterns in a Little-Hopfield associative memory. The first step of the method is to fit an Ising model using minimum probability flow learning to a discrete distribution supported equally on a set of binary target patterns. Next, we use the learned Ising model parameters to define a Little-Hopfield network. We show that when the set of target patterns is storable, these steps result in a Little-Hopfield network that stores all of the patterns as fixed-points. We have also demonstrated that the resulting (convex) algorithm outperforms current techniques for training Little-Hopfield networks. We have shown improved recovery of memories from noisy patterns and improved training speed as compared to training by PER. We have demonstrated optimal storage capacity in the noiseless case, outperforming OPR. We have also demonstrated the unsupervised storage of memories from heavily corrupted training data.
Furthermore, the learning rule that results from our method is local; that is, updating the weights between two units requires only their states and feedforward input. As MPF allows the fitting of large Little-Hopfield networks quickly, new investigations into the structure of Little-Hopfield networks are possible [19]. It is our hope that the robustness and speed of this learning technique will enable practical use of Little-Hopfield associative memories in computational neuroscience, computer science, and scientific modeling.

References