A SPIKING NEURAL NETWORK WITH LOCAL LEARNING RULES DERIVED FROM NONNEGATIVE SIMILARITY MATCHING

Cengiz Pehlevan

John A. Paulson School of Engineering and Applied Sciences,
Harvard University, Cambridge, MA 02138, USA

ABSTRACT

The design and analysis of spiking neural network algorithms will be accelerated by the advent of new theoretical approaches. In an attempt at such approach, we provide a principled derivation of a spiking algorithm for unsupervised learning, starting from the nonnegative similarity matching cost function. The resulting network consists of integrate-and-fire units and exhibits local learning rules, making it biologically plausible and also suitable for neuromorphic hardware. We show in simulations that the algorithm can perform sparse feature extraction and manifold learning, two tasks which can be formulated as nonnegative similarity matching problems.

Index Terms— nonnegative similarity matching, spiking neural networks, online optimization

1. INTRODUCTION

While our brains serve as an evidence of the capability and the efficiency of spike-based computation, despite the recent progress in neural networks research [1], spiking neural networks (SNNs) [2] still remain largely unexplored and underutilized. This is partly because SNNs pose new challenges in theoretical understanding compared to neural networks with analogue units (AUNNs). Novel analytical methods would advance the design and analysis of SNN algorithms.

In this paper, we present a theoretically principled approach to designing SNN algorithms that learn representations from data in an unsupervised manner. Differing from modern deep learning methods [1], we not only "derive" the learning rules but also the dynamics and the architecture of an SNN from a cost function, whose optimization describes a learning task. This approach results in efficient SNNs specialized to solve the task. Access to a cost function allows prediction of the SNN algorithm's behavior on different datasets.

A challenge to the derivation of SNNs is the desired locality of an SNN's learning rules, i.e. SNN synaptic updates should depend only on the activities of the pre- and post-synaptic units. Local updates are necessary for implementability on neuromorphic hardware [3] and also for biological plausibility. But, how could such uninformed learning be optimal in any sense? Naively, if the synapse had access to the activity of other units, it could make a better update. Indeed, existing derivations of learning SNNs ended up with non-local learning rules, starting from various cost functions and optimizing by gradient methods. To arrive at local learning, authors either resorted to approximations [4, 5, 6], or to a contrastive learning procedure [7].

To solve the locality problem, we will adopt a technology that was recently developed for deriving AUNNs: similarity-based cost functions [8]. This family of costs lead to AUNNs with local learning rules when optimized by gradient methods [8, 9]. We will focus on a particular similarity-based cost function, nonnegative similarity matching (NSM) [10, 11], because of its versatile usability across many learning tasks.

In order to introduce the NSM problem, we assume the input data be a set of vectors, $\mathbf{x}_{t=1,\dots,T} \in \mathbb{R}^n$ and the output be another set $\mathbf{y}_{t=1,\dots,T} \in \mathbb{R}^k$. Taking dot product as a similarity measure, NSM aims to learn a representation where the similarities between output vector pairs match that of the input pairs, subject to nonnegativity constraints and regularization:

$$\min_{\forall \mathbf{y}_t \geq 0} \frac{1}{2T^2} \sum_{t=1}^{T} \sum_{t'=1}^{T} \left(\mathbf{x}_t^\top \mathbf{x}_{t'} - \mathbf{y}_t^\top \mathbf{y}_{t'} - \alpha^2 \right)^2 + \frac{2\lambda_1}{T} \sum_{t=1}^{T} \|\mathbf{y}_t\|_1 + \frac{\lambda_2}{T} \sum_{t=1}^{T} \|\mathbf{y}_t\|_2^2 . \quad (1)$$

Without the regularizers ($\alpha = \lambda_1 = \lambda_2 = 0$), this cost function was used for clustering [10, 11, 12], sparse encoding and feature extraction [11, 13], and blind nonnegative source separation [14]. With α and λ_2 turned on, it was used for manifold learning [15]. We also include an l_1-norm regularization for the capability of increased sparsity in the output [16, 17, 18]. Overall, an SNN that solves the NSM problem will be a versatile tool for learning representations from data.

A second challenge to the derivation of SNNs is the spiking dynamics. Recent work showed how to design and derive spike-based optimization algorithms to solve certain optimization problems [19, 20, 21, 22] in non-learning settings. In order to derive a learning SNN with local learning rules, we will use one of these spike-based algorithms, that of [22], to optimize the NSM cost function (1). We call the resulting

algorithm Spiking NSM.

The rest of the paper is organized as follows. In Section 2, we show how an AUNN with local learning rules can be derived from NSM. Building on these results, in Section 3, we derive the Spiking NSM algorithm. In Section 4, we present our numerical experiments on sparse encoding and feature extraction, and manifold learning. We conclude in Section 5.

2. DERIVATION OF AN AUNN ALGORITHM FROM THE NSM COST FUNCTION

We start by deriving an AUNN from the cost in (1). While our presentation mostly follows previous accounts [11, 9], there are important variations and novelties that enable the derivation of the Spiking NSM algorithm in the next section.

There are immediate problems with deriving an AUNN from (1). It has only inputs and outputs, but not synaptic weights. A neural network operates in an online fashion, producing an output $\mathbf{y}_t$ immediately after seeing an input $\mathbf{x}_t$, but, in (1), pairs of inputs and outputs from different time points interact with each other.

These problems can be solved by following the procedure described in [9]. Starting from the NSM cost (1), we obtain a dual min-max objective by introducing new auxiliary variables $\mathbf{W} \in \mathbb{R}^{k\times n}$, $\mathbf{M} \in \mathbb{R}^{k\times k}$, and $\mathbf{b} \in \mathbb{R}^k$, which will be interpreted as synaptic weights shortly:

$$\min_{\mathbf{W}\in\mathbb{R}^{k\times n}} \max_{\mathbf{M}\in\mathbb{R}^{k\times k}} \max_{\mathbf{b}\in\mathbb{R}^k} \frac{1}{T}\sum_{t=1}^{T} l_t(\mathbf{W},\mathbf{M},\mathbf{b}), \qquad (2)$$

where

$$\begin{aligned} l_t :=& \operatorname{Tr}\mathbf{W}^\top\mathbf{W} - \frac{1}{2}\operatorname{Tr}\mathbf{M}^\top\mathbf{M} - \|\mathbf{b}\|_2^2 \\ &+ \min_{\mathbf{y}_t\geq 0} h_t(\mathbf{W},\mathbf{M},\mathbf{b},\mathbf{y}_t), \\ h_t :=& -2\mathbf{y}_t^\top(\mathbf{W}\mathbf{x}_t - \alpha\mathbf{b}) + \mathbf{y}_t^\top\mathbf{M}\mathbf{y}_t \\ &+ 2\lambda_1\|\mathbf{y}_t\|_1 + \lambda_2\|\mathbf{y}_t\|_2^2 . \end{aligned} \qquad (3)$$

The new objective (2) is equivalent to (1) upto a change in the order of optimization, which can be seen by plugging back the optimal values of $\mathbf{W}^* = \frac{1}{T}\sum_t \mathbf{y}_t\mathbf{x}_t^\top$, $\mathbf{M}^* = \frac{1}{T}\sum_t \mathbf{y}_t\mathbf{y}_t^\top$ and $\mathbf{b}^* = \frac{\alpha}{T}\sum_t \mathbf{y}_t$.

The min-max objective allows an online NSM algorithm because the objective is factorized into a summation of terms, l_t, in a way that pairs of inputs and outputs from different time points are decoupled. For each input $\mathbf{x}_t$, we use a two-step alternating optimization procedure [23, 24] on l_t to produce an output $\mathbf{y}_t$ and update variables $\mathbf{W}$, $\mathbf{M}$, and $\mathbf{b}$. We now discuss these steps, and how they map to the operations of an AUNN with local learning rules.

2.1. Solving for outputs with an AUNN

The first step of the alternating optimization is minimizing h_t (and l_t) with respect to nonnegative $\mathbf{y}_t$, while keeping $\mathbf{W}$, $\mathbf{M}$, and $\mathbf{b}$ fixed[1]. Define $\bar{\mathbf{M}} := \mathbf{M} - \operatorname{diag}(\mathbf{M})$, where diag operator sets off-diagonal elements of a matrix to zero. Then, the following dynamical system minimizes h_t[2]:

$$\begin{aligned} \frac{du_i(\tau)}{d\tau} &= -u_i(\tau) + [\mathbf{W}\mathbf{x}_t]_i - \alpha b_i - \left[\bar{\mathbf{M}}\mathbf{y}_t(\tau)\right]_i , \\ y_{t,i}(\tau) = g_i(u_i(\tau)) &:= \begin{cases} 0, & u_i(\tau)\leq\lambda_1 \\ \frac{u_i(\tau)-\lambda_1}{\lambda_2+M_{ii}}, & u_i(\tau)>\lambda_1 \end{cases} , \\ & \qquad i = 1,\ldots,k. \end{aligned} \qquad (4)$$

This system can be interpreted as the dynamics of the neural network shown in Figure 1. $\mathbf{x}_t$ is the input to the network and $\mathbf{y}_t$ is the output vector of unit activities. $\mathbf{W}$ and $-\bar{\mathbf{M}}$ are feedforward and lateral synaptic weight matrices. $-\alpha b_i$ is the synaptic weight to unit i from an input unit with activity 1. Finally, g_i is a unit-dependent activation function.

We note that previous AUNN derivations from similarity-based cost functions used subgradient descent [18] , projected gradient descent [9] or coordinate descent [11, 8, 26] dynamics for this step of the algorithm. Our dynamics choice here is motivated by its generalization to an SNN, which will be presented in the Section 3.

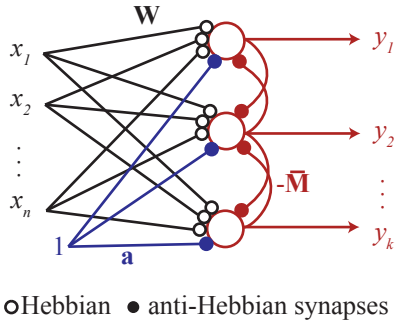


Fig. 1. The network architecture. Some lateral connections are not shown for better visibility. $a_i = -\alpha b_i$ for the AUNN algorithm, and $a_i = -\alpha b_i - \lambda_1$ for the SNN algorithm.

2.2. Updating synaptic weights with local learning rules

The second step of the alternating optimization is to perform gradient updates in $\mathbf{W}$, $\mathbf{b}$, and $\mathbf{M}$ with fixed $\mathbf{y}_t$, which we write in component notation to expose their locality:

$$\begin{aligned} &\Delta W_{ij} = \eta\left(y_{t,i}x_{t,j} - W_{ij}\right), \ \Delta\bar{M}_{ij,i\neq j} = \eta\left(y_{t,i}y_{t,j} - \bar{M}_{ij}\right), \\ &\Delta M_{ii} = \eta\left(y_{t,i}^2 - M_{ii}\right), \qquad \Delta b_i = \eta\left(\alpha y_{t,i} - b_i\right), \end{aligned} \qquad (5)$$

where η is a learning rate. $\mathbf{W}$ update is a Hebbian synaptic plasticity rule. $\bar{\mathbf{M}}$ and $\mathbf{b}$ updates are anti-Hebbian (because of the $-$ signs in corresponding terms in (4)). M_{ii} update changes the gain function of a neuron, and can be interpreted as a homeostatic plasticity rule.

[1] Note that this is a nonnegative elastic net problem [17].

[2] It is easy to show that for this dynamics $\frac{dh_t}{d\tau} \leq 0$. A rigorous discussion of convergence can be found in [25] (see the Conclusion section of [25]).

3. DERIVATION OF THE SPIKING NSM ALGORITHM FROM THE NSM COST FUNCTION

Next, we derive an SNN algorithm with local learning rules, the Spiking NSM algorithm, from the NSM cost function (1). We do this by replacing the optimization algorithm for minimizing h_t with a spike-based one.

3.1. Solving for outputs with an SNN

The SNN of Tang, Lin and Davies [22] minimizes h_t. We first describe the SNN and then cite a theorem for its convergence to the fixed point of the AUNN given in (4). Since the AUNN fixed point is the minimum of h_t, the SNN minimizes h_t.

Consider a network of k integrate-and-fire units with the same architecture as in Figure 1. We denote the i^{th} unit's membrane potential by $V_i(\tau)$, input current by $I_i(\tau)$, and q^{th} spike time by $\tau_{i,q}$. The units are perfect integrators, i.e. their subthreshold membrane potentials are given by:

$$\frac{dV_i(\tau)}{d\tau} = I_i(\tau), \qquad V_i(0) = 0, \qquad i = 1, \ldots, k. \quad (6)$$

When $V_i(\tau)$ reaches a firing threshold $V_i^{\text{th}} := \lambda_2 + M_{ii}$, the unit emits a spike and the membrane potential is set to 0. Synaptic input is defined as

$$\frac{dI_i(\tau)}{d\tau} = -I_i(\tau) + \left[\mathbf{W}\mathbf{x}_t\right]_i - \alpha b_i - \lambda_1 - \left[\bar{\mathbf{M}}\boldsymbol{\sigma}(\tau)\right]_i,$$
$$I_i(0) = \left[\mathbf{W}\mathbf{x}_t\right]_i - \lambda_1 - \alpha b_i, \qquad i = 1, \ldots, k, \quad (7)$$

where $\mathbf{x}_t$ is the input to the network, $\sigma_i(\tau) := \sum_q \delta(\tau - \tau_{i,q})$ is the spike train of the i^{th} unit, $\mathbf{W}$ and $-\bar{\mathbf{M}}$ are feedforward and lateral synaptic weight matrices, and $-\alpha b_i - \lambda_1$ is the synaptic weight to unit i from an input unit with activity 1.

Under mild assumptions, it can be shown that time-averaged spike trains converge to the fixed point of the AUNN defined in (4) and therefore minimize h_t. More precisely, let's define

$$\tilde{y}_i(\tau) := \frac{1}{\tau}\int_0^\tau d\tau'\, \sigma_i(\tau'), \qquad i = 1, \ldots, k. \quad (8)$$

Theorem (Tang, Lin, Davies [22])**.** *(Informal) Assume that the duration between a unit's consecutive spikes is not arbitrarily long but upper bounded, with the exception of the unit stopping spiking altogether after some time. Then, as $\tau \to \infty$, $\tilde{y}_i(\tau)$ converges to the value of $y_{t,i}$ at the fixed point of the dynamical system* (4)*.*

Proof. The results of [22] can be easily extended to prove this result. See especially the Discussion section of [22]. □

3.2. Updating synaptic weights with local learning rules

After the spiking dynamics converges, we update $\mathbf{W}$, $\mathbf{b}$ and $\mathbf{M}$ as in (5), using the spiking estimate for $\mathbf{y}_t$ from (8). Note that M_{ii} updates are still interpreted as homeostatic plasticity, but this time they change the firing thresholds of units.

The final Spiking NSM algorithm is summarized below.

Algorithm Spiking NSM

Input: Parameters α, λ_1 and λ_2. Initial weights $\mathbf{M} \in \mathbb{R}^{k\times k}$, $\mathbf{W} \in \mathbb{R}^{k\times n}$ and $\mathbf{b} \in \mathbb{R}^k$.
for $t = 1, 2, 3, \ldots$ **do**
`// Spiking neural dynamics`
Taking $\mathbf{x}_t$ as input, run the SNN defined by equations (6), (7), (8) until convergence.
`// Synaptic and homeostatic plasticity`
Update $\mathbf{W}$, $\mathbf{b}$ and $\mathbf{M}$ as in (5), using the spiking estimate for $\mathbf{y}_t$ from (8).
end for

4. EXPERIMENTAL RESULTS

In this section, we apply the Spiking NSM algorithm to various datasets for sparse encoding and feature extraction [11], and manifold learning [15]. Our purpose here is not to compare the performance of NSM with other unsupervised learning methods, this was done in [12, 13, 14]. We wish to demonstrate that Spiking NSM actually performs online NSM.

4.1. Solving for outputs with spike-based dynamics

Before attempting a learning task, we first checked whether the SNN defined by equations (6), (7), (8) indeed minimizes h_t, by simulating it with randomly chosen values for lateral connectivity and inputs. More precisely, we set $\alpha = 0.3$, $\lambda_1 = 0.3$, $\lambda_2 = 0.1$, drew b_i from a uniform distribution in $[0, 1]$ and $(\mathbf{W}\mathbf{x})_i$ in $[0, 5]$, and set $\mathbf{M} = \mathbf{V}\mathbf{V}^\top$, where V_{ij} were drawn from a uniform distribution in $[0, 1/\sqrt{k}]$. For each $k \in \{2, 4, 8, 16, 32, 64, 128, 256\}$, we repeated this procedure until we obtained 100 accepted parameter sets. A parameter set was accepted if the norm of the minimum of h_t, found by MATLAB's fmincon function, had an l_2-norm greater than 0.01. For each parameter choice, we simulated the corresponding SNN until $\tau = 500$, using a first-order Euler method with a step-size $d\tau = 0.01$.

Figure 2 shows that the SNN achieves the minimum of h_t within a few percent. The error for a parameter set was measured by $\|\mathbf{y} - \hat{\mathbf{y}}\|_2 \,/\, \|\hat{\mathbf{y}}\|_2$, where $\mathbf{y}$ is the result from the SNN and $\hat{\mathbf{y}}$ from fmincon. We observed that longer simulation times and finer step-sizes led to better performance. In the rest of this section, we used the simulation time and step size configuration used here.

4.2. Sparse encoding and feature extraction

Previously, NSM was shown to extract sparse features from data [11, 14]. We trained Spiking NSM networks on two

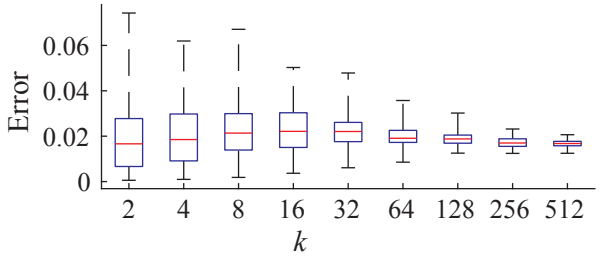
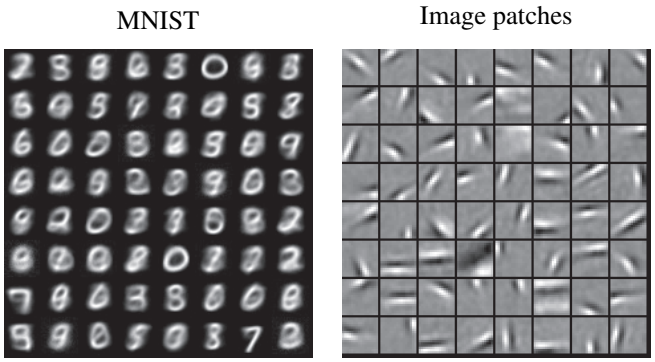
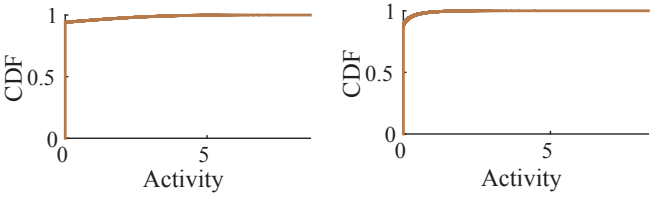


Fig. 2. Spike-based minimization of h_t. Red lines show median, box edges show 25th and 75th percentiles and whiskers show maxima and minima.

datasets to test this function: 1) the MNIST dataset of 6×10^4 images of hand-written digits [27], and 2) a dataset of 4×10^5 16-by-16 image patches sampled randomly from a set of whitened natural scenes [23]. For both simulations, $\alpha = \lambda_2 = 0$, initial $\mathbf{M}$ was set to identity matrix and $\mathbf{b}$ to zero. Learning rates were 10^{-3} for the first 10^4 steps, 10^{-5} for the next 9×10^4 steps, and 0.5×10^{-5} later. For MNIST, $\lambda_1 = 0.5$, $k = 196$ and initial W_{ij} were drawn uniformly from $[0, 1/14]$. For image patches, $\lambda_1 = 0$, $k = 256$ and initial W_{ij} were drawn from $\mathcal{N}(0, 1/196)$. At each iteration, a randomly chosen datum was shown to the network.



(a) Examples of learned features



(b) CDFs of output unit activity

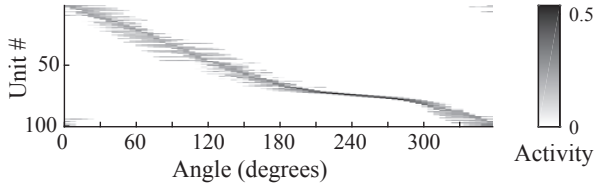
Fig. 3. Spiking NSM extracts sparse features.

Figure 3 shows the results of our simulations after 10^6 iterations. Panel (a) displays 64 examples of learned features for each dataset, extracted from the rows of $\mathbf{W}$ as in [11]. Image patch features are oriented edges, as in sparse coding [23]. Panel (b) displays cumulative distribution functions (CDFs) of the networks' time-averaged spiking output activities (Eq. (8)) calculated over the whole datasets. Activities are highly sparse.

4.3. Manifold learning

NSM learns a data manifold by learning features that tile the manifold [15]. To test this function, we used a one-dimensional data manifold in a high dimensional space, composed of 71 576-by-768 images of a shoe rotated by 5°increments [28] (examples shown in Figure 4(a)). After normalizing each image to unit norm, we trained a Spiking NSM with $k = 100$ output units and $\lambda_1 = \lambda_2 = 0$, $\alpha = 0.8$. The α parameter sets the scale of local similarity neighborhoods [15]. The rest of the simulation parameters matched the MNIST case.

(a) Examples of input images (b) Examples of learned features



(c) Ordered unit responses as a function of the rotation angle

Fig. 4. Spiking NSM tiles data manifolds

Figure 4 shows our results after 1.8×10^4 iterations. Panel (b) displays 6 example learned features, which are localized to the vicinity of particular shoe rotation angles. Panel (c) shows the time-averaged spiking activity of output units (Eq. (8)) as a function of the shoe's rotation angle. As promised, the units tile the data manifold.

5. CONCLUSION

We presented a principled derivation of the Spiking NSM algorithm, which exhibits local learning rules, from the NSM cost function (1). We applied the algorithm to various datasets and interpreted its action as sparse feature extraction and encoding, or manifold learning, based on analytical analyses of the NSM cost function [14, 15].

With the advent of new spiking neuromorphic hardware [3], the need for new SNN algorithms with local learning rules is increasing. We expect the principled approach presented in this paper to be useful in this quest.

6. REFERENCES

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

[2] C. D. Schuman et al., "A survey of neuromorphic computing and neural networks in hardware," *arXiv:1705.06963*, 2017.

[3] M. Davies et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, January 2018.

[4] P. Vertechi, W. Brendel, and C.K. Machens, "Unsupervised learning of an efficient short-term memory network," in *NeurIPS*, 2014, pp. 3653–3661.

[5] A. Gilra and W. Gerstner, "Predicting non-linear dynamics by stable local learning in a recurrent spiking neural network," *Elife*, vol. 6, pp. e28295, 2017.

[6] A. Alemi et al., "Learning nonlinear dynamics in efficient, balanced spiking networks using local plasticity rules.," in *AAAI*, 2018.

[7] T.-H. Lin and P.T.P. Tang, "Dictionary learning by dynamical neural networks," *arXiv:1805.08952*, 2018.

[8] C. Pehlevan, T. Hu, and D.B. Chklovskii, "A hebbian/anti-hebbian neural network for linear subspace learning: A derivation from multidimensional scaling of streaming data," *Neural Comput.*, vol. 27, pp. 1461–1495, 2015.

[9] C. Pehlevan, A.M. Sengupta, and D.B. Chklovskii, "Why do similarity matching objectives lead to hebbian/anti-hebbian networks?," *Neural Comput.*, vol. 30, no. 1, pp. 84–124, 2018.

[10] D. Kuang, H. Park, and C.H.Q. Ding, "Symmetric nonnegative matrix factorization for graph clustering.," in *SDM*. SIAM, 2012, vol. 12, pp. 106–117.

[11] C. Pehlevan and D.B. Chklovskii, "A hebbian/anti-hebbian network derived from online non-negative matrix factorization can cluster and discover sparse features," in *ACSSC*. IEEE, 2014, pp. 769–775.

[12] Y. Bahroun, E. Hunsicker, and A. Soltoggio, "Neural networks for efficient nonlinear online clustering," in *ICONIP*. Springer, 2017, pp. 316–324.

[13] Y. Bahroun and A. Soltoggio, "Online representation learning with single and multi-layer hebbian networks for image classification," in *ICANN*. Springer, 2017, pp. 354–363.

[14] C. Pehlevan, S. Mohan, and D.B. Chklovskii, "Blind nonnegative source separation using biological neural networks," *Neural Comput.*, vol. 29, no. 11, pp. 2925–2954, 2017.

[15] A. Sengupta et al., "Manifold-tiling localized receptive fields are optimal in similarity-preserving neural networks," in *NeurIPS*, 2018.

[16] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. R. Stat. Soc. B.*, pp. 267–288, 1996.

[17] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J. R. Stat. Soc. B.*, vol. 67, no. 2, pp. 301–320, 2005.

[18] T. Hu, C. Pehlevan, and D.B. Chklovskii, "A hebbian/anti-hebbian network for online sparse dictionary learning derived from symmetric matrix factorization," in *ACSSC*. IEEE, 2014, pp. 613–619.

[19] T. Hu, A. Genkin, and D.B. Chklovskii, "A network of spiking neurons for computing sparse representations in an energy-efficient way," *Neural Comput.*, vol. 24, no. 11, pp. 2852–2872, 2012.

[20] S. Shapero et al., "Optimal sparse approximation with integrate and fire neurons," *Int. J. Neural. Syst.*, vol. 24, no. 05, pp. 1440001, 2014.

[21] M. Boerlin and S. Denève, "Spike-based population coding and working memory," *PLoS Comput. Biol.*, vol. 7, no. 2, pp. e1001080, 2011.

[22] P.T.P. Tang, T.-H. Lin, and M. Davies, "Sparse coding by spiking neural networks: Convergence theory and computational results," *arXiv:1705.05475*, 2017.

[23] B.A. Olshausen and D.J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607, 1996.

[24] S. Arora et al., "Simple, efficient, and neural algorithms for sparse coding," *Proc. Mach. Learn Res.*, 2015.

[25] P.T.P. Tang, "Convergence of lca flows to (c) lasso solutions," *arXiv:1603.01644*, 2016.

[26] H.S. Seung and J. Zung, "A correlation game for unsupervised learning yields computational interpretations of hebbian excitation, anti-hebbian inhibition, and synapse elimination," *arXiv:1704.00646*, 2017.

[27] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[28] J.-M. Geusebroek, G. J. Burghouts, and A.W.M. Smeulders, "The amsterdam library of object images," *Int. J. Comput. Vis.*, vol. 61, no. 1, pp. 103–112, 2005.