Sparse coding via thresholding and local competition in neural circuits

Christopher J. Rozell, Don H. Johnson,
Richard G. Baraniuk, Bruno A. Olshausen

Abstract

While evidence indicates that neural systems may be employing sparse approximations to represent sensed stimuli, the mechanisms underlying this ability are not understood. We describe a *locally competitive algorithm* (LCA) that solves a collection of sparse coding principles minimizing a weighted combination of mean-squared error (MSE) and a coefficient cost function. LCAs are designed to be implemented in a dynamical system composed of many neuron-like elements operating in parallel. These algorithms use thresholding functions to induce local (usually one-way) inhibitory competitions between nodes to produce sparse representations. LCAs produce coefficients with sparsity levels comparable to the most popular centralized sparse coding algorithms while being readily suited for neural implementation. Additionally, LCA coefficients for video sequences demonstrate inertial properties that are both qualitatively and quantitatively more regular (i.e., smoother and more predictable) than the coefficients produced by greedy algorithms.

1 Introduction

Natural images can be well-approximated by a small subset of elements from an overcomplete dictionary (Field, 1994; Olshausen, 2003; Candès and Donoho, 2004). The process of choosing a good subset of dictionary elements along with the corresponding coefficients to represent a signal is known as *sparse approximation*. Recent theoretical and experimental evidence indicates that many sensory neural systems appear to employ similar sparse representations with their population codes (Vinje and Gallant, 2002; Lewicki, 2002; Olshausen and Field, 2004, 1996; Delgutte et al., 1998), encoding a stimulus in the activity of just a few neurons. While sparse coding in neural systems is an intriguing hypothesis, the challenge of collecting simultaneous data from large neural populations makes it difficult to evaluate its credibility without testing predictions from a specific proposed coding mechanism.

Unfortunately, we currently lack a proposed sparse coding mechanism that is realistically implementable in the parallel architectures of neural systems *and* produces sparse coefficients that efficiently represent the time-varying stimuli important to biological systems. Sparse approximation is a difficult non-convex optimization problem that is at the center of much research in mathematics and signal processing. Existing sparse approximation algorithms suffer from one or more of the following drawbacks: 1) they are not implementable in the parallel analog architectures used by neural systems; 2) they have difficulty producing exactly zero-valued coefficients in finite time; 3) they produce coefficients for time-varying stimuli that contain inefficient fluctuations, making the stimulus content more difficult to interpret; or 4) they only use a heuristic approximation to minimizing a desired objective function.

We introduce and study a new neurally plausible algorithm based on the principles of *thresholding* and *local competition* that solves a family of sparse approximation problems corresponding to various sparsity metrics. In our Locally Competitive Algorithms (LCAs), neurons in a population continually compete with neighboring units using (usually one-way) lateral inhibition to calculate coefficients representing an input signal using an overcomplete dictionary. Our continuous-time LCA is described by the dynamics of a system of nonlinear ordinary differential equations (ODEs) that govern the internal state (membrane potential) and external communication (short-term firing rate) of units in a neural population. These systems use computational primitives that correspond to simple analog elements (e.g., resistors, capacitors, amplifiers), making them realistic for parallel implementations. We show that each LCA corresponds to an optimal sparse approximation problem that minimizes an energy function combining reconstruction mean-squared error (MSE) and a sparsity-inducing cost function.

This paper develops a neural architecture for LCAs, shows their correspondence to a broad class of sparse approximation problems, and demonstrates that LCAs possess three properties critical for a neurally plausible sparse coding algorithm. First, we show that the LCA dynamical system is stable, guaranteeing that a physical implementation is well-behaved. Next, we show that LCAs perform their primary task well, finding codes for fixed images that have sparsity comparable to the most popular centralized algorithms. Finally, we demonstrate that LCAs display *inertia*, coding video sequences with a coefficient time series that is significantly smoother in time than the coefficients produced by other algorithms. This increased coefficient regularity better reflects the smooth nature of natural input signals, making the coefficients much more predictable and making it easier for higher-level structures to identify and understand the changing content in the time-varying stimulus.

Although still lacking in biophysical realism, the LCA methods presented here represent a first step toward a testable neurobiological model of sparse coding. As an added benefit, the parallel analog architecture described by our LCAs could greatly benefit the many modern signal processing applications that rely on sparse approximation. While the principles we describe apply to many signal modalities, we will focus on the visual system and the representation of video sequences.

2 Background and related work

2.1 Sparse approximation

Given an N-dimensional stimulus $\boldsymbol{s} \in \mathbb{R}^N$ (e.g., an N-pixel image), we seek a representation in terms of a dictionary $\mathcal{D}$ composed of M vectors $\{\boldsymbol{\phi}_m\}$ that span the space $\mathbb{R}^N$. Define the ℓ^p norm of the vector $\boldsymbol{x}$ to be $||\boldsymbol{x}||_p = \left(\sum_m |x_m|^p\right)^{1/p}$ and the inner product between $\boldsymbol{x}$ and $\boldsymbol{y}$ to be $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_m x_m y_m$. Without loss of generality, assume the dictionary vectors are unit-norm, $||\boldsymbol{\phi}_m||_2 = 1$. When the dictionary is overcomplete $(M > N)$, there are an infinite number of ways to choose coefficients $\{a_m\}$ such that $\boldsymbol{s} = \sum_{m=1}^M a_m \boldsymbol{\phi}_m$. In optimal sparse approximation, we seek the coefficients having the fewest number of non-zero entries by solving the minimization problem

$$\min_{\boldsymbol{a}} ||\boldsymbol{a}||_0 \quad \text{subject to} \quad \boldsymbol{s} = \sum_{m=1}^{M} a_m \boldsymbol{\phi}_m, \tag{1}$$

where the ℓ^0 "norm"[1] denotes the number of non-zero elements of $\boldsymbol{a} = [a_1, a_2, \ldots, a_M]$. Unfortunately, this combinatorial optimization problem is NP-hard (Natarajan, 1995).

In the signal processing community, two approaches are typically used to find acceptable suboptimal solutions to this intractable problem. The first general approach substitutes an alternate sparsity measure to convexify the ℓ^0 norm. One well-known example is Basis Pursuit (BP) (Chen et al., 2001), which replaces the ℓ^0 norm with the ℓ^1 norm

$$\min_{\boldsymbol{a}} ||\boldsymbol{a}||_1 \quad \text{subject to} \quad \boldsymbol{s} = \sum_{m=1}^{M} a_m \boldsymbol{\phi}_m. \tag{2}$$

Despite this substitution, BP has the same solution as the optimal sparse approximation problem (Donoho and Elad, 2003) if the signal is sparse compared to the most similar pair of dictionary elements (e.g., $||\boldsymbol{a}||_0 < \min_{m \neq n} \frac{1}{2}\left[1 + 1/\langle \boldsymbol{\phi}_m, \boldsymbol{\phi}_n \rangle\right]$). In practice, the presence of signal noise often leads to using a modified approach called Basis Pursuit De-Noising (BPDN) (Chen et al., 2001) that makes a tradeoff between

[1] While clearly not a norm in the mathematical sense, we will use this terminology prevalent in the literature.

reconstruction mean-squared error (MSE) and sparsity in an unconstrained optimization problem:

$$\min_{\boldsymbol{a}} \left(\left|\left| \boldsymbol{s} - \sum_{m=1}^{M} a_m \boldsymbol{\phi}_m \right|\right|_2^2 + \lambda \left|\left|\boldsymbol{a}\right|\right|_1 \right), \tag{3}$$

where λ is a tradeoff parameter. BPDN provides the ℓ^1-sparsest approximation for a given reconstruction quality. There are many algorithms that can be used to solve the BPDN optimization problem, with interior point-type methods being the most common choice.

The second general approach employed by signal processing researchers uses iterative greedy algorithms to constructively build up a signal representation (Tropp, 2004). The canonical example of a greedy algorithm[2] is known in the signal processing community as Matching Pursuit (MP) (Mallat and Zhang, 1993). The MP algorithm is initialized with a residual $r_0 = \boldsymbol{s}$. At the k^{th} iteration, MP finds the index of the single dictionary element best approximating the current residual signal, $\theta_k = \arg\max_m |\langle r_{k-1}, \boldsymbol{\phi}_m \rangle|$. The coefficient $d_k = \langle r_{k-1}, \boldsymbol{\phi}_{\theta_k} \rangle$ and index θ_k are recorded as part of the reconstruction, and the residual is updated, $r_k = r_{k-1} - \boldsymbol{\phi}_{\theta_k} d_k$. After K iterations, the signal approximation using MP is given by $\widehat{\boldsymbol{s}} = \sum_{k=1}^{K} \boldsymbol{\phi}_{\theta_k} d_k$. Though they may not be optimal in general, greedy algorithms often efficiently find good sparse signal representations in practice.

2.2 Sparse coding in neural systems

Recent research in neuroscience suggests that V1 population responses to natural stimuli may be the result of a sparse approximation of images. For example, it has been shown that both the spatial and temporal properties of V1 receptive fields may be accounted for in terms of a dictionary that has been optimized for sparseness in response to natural images (Olshausen, 2003). Additionally, V1 recordings in response to natural scene stimuli show activity levels (corresponding to the coefficients $\{a_m\}$) becoming more sparse as neighboring units are also stimulated (Vinje and Gallant, 2002). These populations are typically very overcomplete (Olshausen and Field, 2004), allowing great flexibility in the representation of a stimulus. Using this flexibility to achieve sparse codes might offer many advantages to sensory neural systems, including enhancing the performance of subsequent processing stages, increasing the storage capacity in associative memories, and increasing the energy efficiency of the system (Olshausen and Field, 2004).

However, existing sparse approximation algorithms do not have implementations that correspond both naturally and efficiently to plausible neural architectures. For convex relaxation approaches, a network implementation of BPDN can be constructed (Olshausen and Field, 1996), following the common practice of using dynamical systems to implement direct gradient descent optimization (Cichocki and Unbehauen, 1993). Unfortunately, this implementation has two major drawbacks. First, it lacks a natural mathematical mechanism to make small coefficients identically zero. While the true BPDN solution would have many coefficients that are exactly zero, direct gradient methods to find an approximate solution in finite time produce coefficients that merely have small magnitudes. Ad hoc thresholding can be used on the results to produce zero-valued coefficients, but such methods lack theoretical justification and can be difficult to use without oracle knowledge of the best threshold value. Second, this implementation requires persistent (two-way) signaling between all units with overlapping receptive fields (e.g., even a node with a nearly zero value would have to continue sending inhibition signals to all similar nodes). In greedy algorithm approaches, spiking neural circuits can be constructed to implement MP (Perrinet, 2005). Unfortunately, this type of circuit implementation relies on a temporal code that requires tightly coupled and precise elements to both encode and decode.

Beyond implementation considerations, existing sparse approximation algorithms also do not consider the time-varying stimuli faced by neural systems. A time-varying input signal $\boldsymbol{s}(t)$ is represented with a set of time-varying coefficients $\{a_m(t)\}$. While temporal coefficient changes are necessary to encode stimulus

[2] Many types of algorithms for convex and non-convex optimization could be considered "greedy" in some sense (including systems based on descending the gradient of an instantaneous energy function). Our use of this terminology will apply to the family of iterative greedy algorithms such as MP to remain consistent with the majority of the signal processing literature.

changes, the most useful encoding would use coefficient changes that reflect the character of the stimulus. In particular, sparse coefficients should have smooth temporal variations in response to smooth changes in the image. However, most sparse approximation schemes have a single goal: select the smallest number of coefficients to represent a fixed signal. This single-minded approach can produce coefficient sequences for time-varying stimuli that are erratic, with drastic changes not only in the values of the coefficients but also in the selection of which coefficients are used. These erratic temporal codes are inefficient because they introduce uncertainty about which coefficients are coding the most significant stimulus changes, thereby complicating the process of understanding the changing stimulus content.

In Section 3 we develop our LCAs, in which dictionary elements continually fight for the right to represent the stimulus. These LCAs adapt their coefficients continually over time as the input changes without having to build a new representation from scratch at each time step. This evolution induces inertia in the coefficients, regularizing the temporal variations for smoothly varying input signals. In contrast to the problems seen with current algorithms, our LCAs are easily implemented in analog circuits composed neuron-like elements, and they encourage both sparsity and smooth temporal variations in the coefficients as the stimulus changes.

2.3 Other related work

There are several sparse approximation methods that do not fit into the two primary approaches of pure greedy algorithms or convex relaxation. Methods such as Sparse Bayesian Learning (Wipf and Rao, 2004; Tipping, 2001), FOCUSS (Rao and Kreutz-Delgado, 1999), modifications of greedy algorithms that select multiple coefficients on each iteration (Donoho et al., 2006; Pece and Petkov, 2000; Feichtinger et al., 1994) and MP extensions that perform an orthogonalization at each step (Davis et al., 1994; Rebollo-Neira and Lowe, 2002) involve computations that would be very difficult to implement in a parallel distributed architecture. While FOCUSS can implement both ℓ^1 global optimization and ℓ^0 local optimization (Rao and Kreutz-Delgado, 1999) in a dynamical system (Kreutz-Delgado et al., 2003) that uses parallel computation to implement a competition strategy among the nodes (strong nodes are encouraged to grow while weak nodes are penalized), it does not lend itself to forming smooth time-varying representations because coefficients cannot be reactivated once they go to zero.

There are also several sparse approximation methods built on a parallel computational framework that are related to our LCAs (Fischer et al., 2004; Kingsbury and Reeves, 2003; Herrity et al., 2006; Rehn and Sommer, 2007; Hale et al., 2007; Figueiredo and Nowak, 2003; Daubechies et al., 2004; Blumensath and Davies, 2008). These algorithms typically start the first iteration with many super-threshold coefficients and iteratively try to prune the representation through a thresholding procedure, rather than charging up from zero as in our LCAs. Appendix A contains a detailed comparison.

3 Locally competitive algorithms for sparse coding

3.1 Architecture of locally competitive algorithms

Our LCAs associate each element of the dictionary $\boldsymbol{\phi}_m \in \mathcal{D}$ with a separate computing node, or "neuron". When the system is presented with an input image $\boldsymbol{s}(t)$, the population of neurons evolves according to fixed dynamics (described below) and settle on a collective output $\{a_m(t)\}$, corresponding to the short-term average firing rate of the neurons.[3] The goal is to define the LCA system dynamics so that few coefficients have non-zero values while approximately reconstructing the input, $\widehat{\boldsymbol{s}}(t) = \sum_m a_m(t)\, \boldsymbol{\phi}_m \approx \boldsymbol{s}(t)$.

The LCA dynamics are inspired by several properties observed in neural systems: inputs cause the membrane potential to "charge up" like a leaky integrator; membrane potentials exceeding a threshold produce "action potentials" for extracellular signaling; and these super-threshold responses inhibit neighboring units through horizontal connections. We represent each unit's sub-threshold value by a time-varying internal state $u_m(t)$. The unit's excitatory input current is proportional to how well the image matches with the

[3] Note that for analytical simplicity we allow positive and negative coefficients, but rectified systems could use two physical units to implement one LCA node.

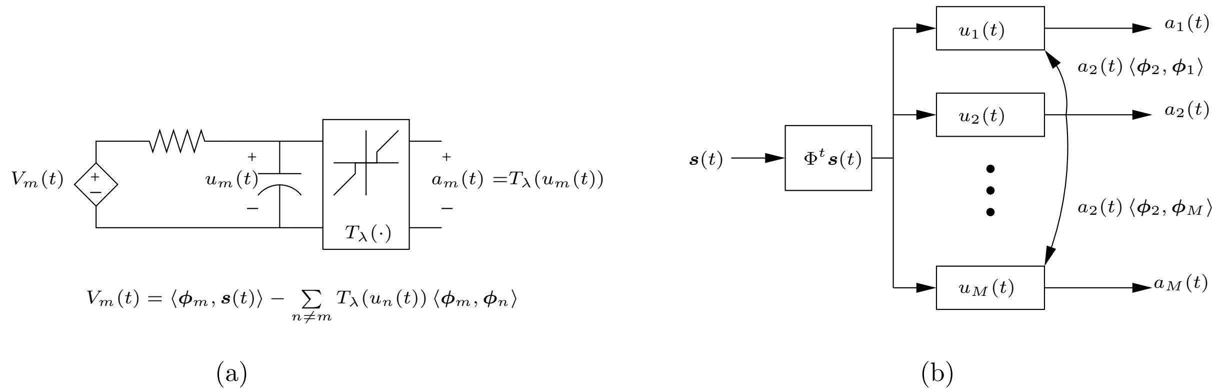

$V_m(t)$
$u_m(t)$
$T_\lambda(\cdot)$
$a_m(t) = T_\lambda(u_m(t))$
$V_m(t) = \langle \boldsymbol{\phi}_m, \boldsymbol{s}(t)\rangle - \sum_{n\neq m} T_\lambda(u_n(t)) \langle \boldsymbol{\phi}_m, \boldsymbol{\phi}_n\rangle$
(a)
$\boldsymbol{s}(t)$
$\Phi^t \boldsymbol{s}(t)$
$u_1(t)$
$u_2(t)$
$u_M(t)$
$a_1(t)$
$a_2(t)$
$a_M(t)$
$a_2(t)\langle \boldsymbol{\phi}_2, \boldsymbol{\phi}_1\rangle$
$a_2(t)\langle \boldsymbol{\phi}_2, \boldsymbol{\phi}_M\rangle$
(b)

Figure 1: (a) LCA nodes behave as a leaky integrators, charging with a speed that depends on how well the input matches the associated dictionary element and the inhibition received from other nodes. (b) A system diagram shows the inhibition signals being sent between nodes. In this case, only node 2 is shown as being active (i.e., having a coefficient above threshold) and inhibiting its neighbors. Since the neighbors are inactive then the inhibition is one-way.

node's receptive field, $b_m(t) = \langle \boldsymbol{\phi}_m, \boldsymbol{s}(t)\rangle$. When the internal state u_m of a node becomes significantly large, the node becomes "active" and produces an output signal a_m used to represent the stimulus and inhibit other nodes. This output coefficient is the result of an activation function applied to the membrane potential, $a_m = T_\lambda(u_m)$, parameterized by the system threshold λ. Though similar activation functions have traditionally taken a sigmoidal form, we consider activation functions that operate as thresholding devices (e.g., essentially zero for values below λ and essentially linear for values above λ).

The nodes best matching the stimulus will have internal state variables that charge at the fastest rates and become active soonest. To induce the competition that allows these nodes to suppress weaker nodes, we have active nodes inhibit other nodes with an inhibition signal proportional to both their activity level and the similarity of the nodes' receptive fields. Specifically, the inhibition signal from the active node m to any other node n is proportional to $a_m G_{m,n}$, where $G_{m,n} = \langle \boldsymbol{\phi}_m, \boldsymbol{\phi}_n\rangle$. The possibility of unidirectional inhibition gives strong nodes a chance to prevent weaker nodes from becoming active and initiating counter-inhibition, thus making the search for a sparse solution more energy efficient. Note that unlike the direct gradient descent methods described in Section 2 that require two-way inhibition signals from all nodes that overlap (i.e., have $G_{m,n} \neq 0$), LCAs only require one-way inhibition from a small selection of nodes (i.e., only the active nodes).

Putting all of the above components together, LCA node dynamics are expressed by the nonlinear ordinary differential equation (ODE)

$$\dot{u}_m(t) = \frac{1}{\tau}\left[b_m(t) - u_m(t) - \sum_{n\neq m} G_{m,n} a_n(t)\right]. \qquad (4)$$

This ODE is essentially the same form as the well-known continuous Hopfield network (Hopfield, 1984). Figure 1 shows a LCA node circuit schematic and a system diagram illustrating the lateral inhibition. To express the system of coupled nonlinear ODEs that govern the whole dynamic system, we represent the internal state variables in the vector $\boldsymbol{u}(t) = [u_1(t), \ldots, u_M(t)]^t$, the active coefficients in the vector $\boldsymbol{a}(t) = [a_1(t), \ldots, a_M(t)]^t = T_\lambda(\boldsymbol{u}(t))$, the dictionary elements in the columns of the $(N \times M)$ matrix $\Phi = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_M]$ and the driving inputs in the vector $\boldsymbol{b}(t) = [b_1(t), \ldots, b_M(t)]^t = \Phi^t \boldsymbol{s}(t)$. The function $T_\lambda(\cdot)$ performs element-by-element thresholding on vector inputs. The stimulus approximation is $\widehat{\boldsymbol{s}}(t) = \Phi \boldsymbol{a}(t)$, and the full dynamic system equation is

$$\dot{\boldsymbol{u}}(t) = f(\boldsymbol{u}(t)) = \frac{1}{\tau}\left[\boldsymbol{b}(t) - \boldsymbol{u}(t) - \left(\Phi^t\Phi - I\right)\boldsymbol{a}(t)\right], \qquad (5)$$
$$\boldsymbol{a}(t) = T_\lambda(\boldsymbol{u}(t)).$$

The LCA architecture described by (5) solves a family of sparse approximation problems with different sparsity measures. Specifically, LCAs descend an energy function combining the reconstruction MSE and a sparsity-inducing cost penalty $C(\cdot)$,

$$E(t) = \frac{1}{2}\left|\left|\boldsymbol{s}(t) - \widehat{\boldsymbol{s}}(t)\right|\right|^2 + \lambda\sum_m C(a_m(t)). \qquad (6)$$

The specific form of the cost function $C(\cdot)$ is determined by the form of the thresholding activation function $T_\lambda(\cdot)$. For a given threshold function, the cost function is specified (up to a constant) by

$$\lambda\frac{dC(a_m)}{da_m} = u_m - a_m = u_m - T_\lambda(u_m). \qquad (7)$$

This correspondence between the thresholding function and the cost function can be seen by computing the derivative of E with respect to the active coefficients, $\{a_m\}$ (see Appendix B). Using the relationship in (7) and letting the internal states $\{u_m\}$ evolve according to $\dot{u}_m \propto -\frac{\partial E}{\partial a_m}$ yields the equation for the internal state dynamics above in (4). Note that although the dynamics are specified through a gradient approach, the system is not performing direct gradient descent (e.g., $\dot{u}_m \neq -\frac{\partial E}{\partial u_m}$). As long as a_m and u_m are related by a monotonically increasing function, the $\{a_m\}$ will also descend the energy function E. This method for showing the correspondence between a dynamic system and an energy function is essentially the same procedure used by Hopfield (Hopfield, 1984) to establish network dynamics in associative memory systems.

We focus specifically on the cost functions associated with thresholding activation functions. Thresholding functions limit the lateral inhibition by allowing only "strong" units to suppress other units and forcing most coefficients to be identically zero. For our purposes, thresholding functions $T_\lambda(\cdot)$ have two distinct behaviors over their range: they are essentially linear with unit slope above threshold λ, and essentially zero below threshold. Among many reasonable choices for thresholding functions, we start with a smooth sigmoidal function

$$T_{(\alpha,\gamma,\lambda)}(u_m) = \frac{u_m - \alpha\lambda}{1 + e^{-\gamma(u_m-\lambda)}}, \qquad (8)$$

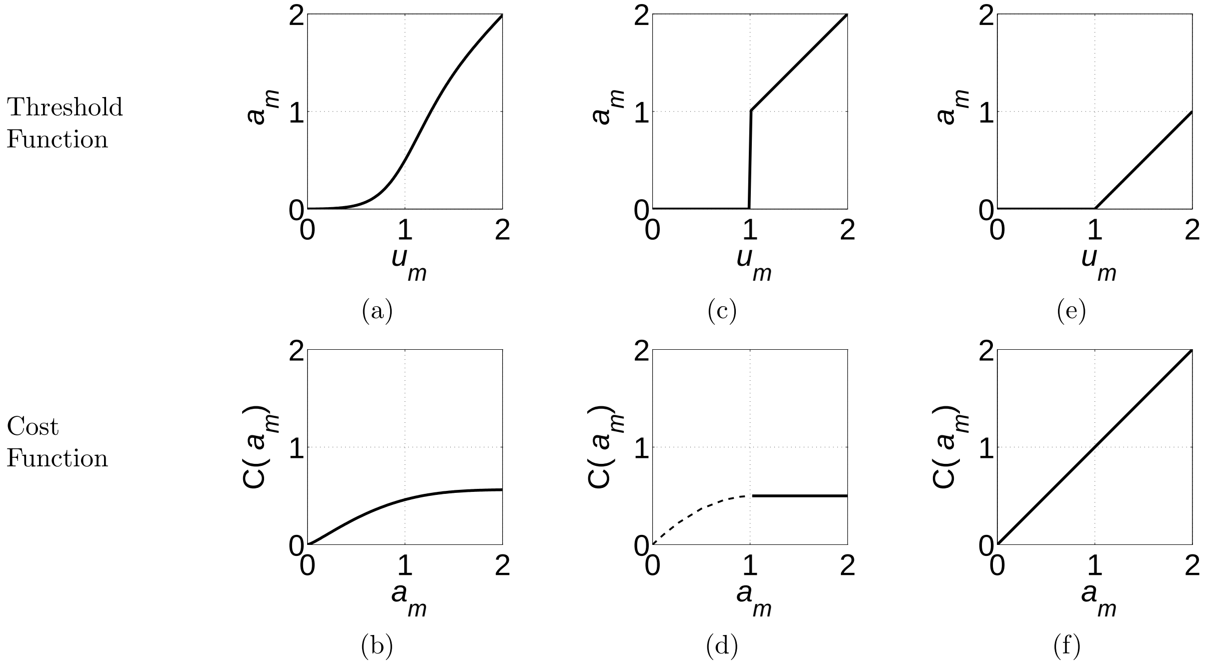
where γ is a parameter controlling the speed of the threshold transition and $\alpha \in [0,1]$ indicates what fraction of an additive adjustment is made for values above threshold. An example sigmoidal thresholding function is shown in Figure 2a. We are particularly interested in the limit of this thresholding function as $\gamma \to \infty$, a piecewise linear function we denote as the *ideal thresholding function*. In the signal processing literature, $T_{(0,\infty,\lambda)}(\cdot) = \lim_{\gamma\to\infty} T_{(0,\gamma,\lambda)}(\cdot)$ is known as a "hard" thresholding function and $T_{(1,\infty,\lambda)}(\cdot) = \lim_{\gamma\to\infty} T_{(1,\gamma,\lambda)}(\cdot)$ is known as a "soft" thresholding function (Donoho, 1995).

3.2 Sparse approximation by locally competitive algorithms

Combining (7) and (8), we can integrate numerically to determine the cost function corresponding to $T_{(\alpha,\gamma,\lambda)}(\cdot)$, shown in Figure 2b. For the ideal threshold functions we derive a corresponding *ideal cost function*,

$$C_{(\alpha,\infty,\lambda)}(a_m) = \frac{(1-\alpha)^2\lambda}{2} + \alpha\left|a_m\right|. \qquad (9)$$

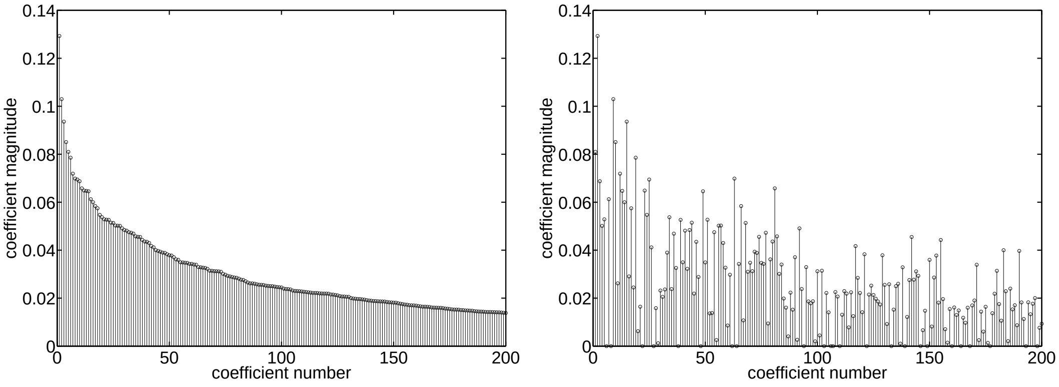
Details of this derivation are in Appendix C. Note that unless $\alpha = 1$ the ideal cost function has a gap because active coefficients cannot take all possible values, $|a_m| \notin [0, (1-\alpha)\lambda]$ (i.e., the ideal thresholding function is not technically invertible).



(a) (c) (e)

(b) (d) (f)

Figure 2: Relationship between the threshold function $T_{(\alpha,\gamma,\lambda)}(\cdot)$ and the sparsity cost function $C(\cdot)$. Only the positive half of the symmetric threshold and cost functions are plotted. (a) Sigmoidal threshold function and (b) cost function for $\gamma = 5$, $\alpha = 0$ and $\lambda = 1$. (c) The ideal hard thresholding function ($\gamma = \infty$, $\alpha = 0$, $\lambda = 1$) and the (d) corresponding cost function. The dashed line shows the limit, but coefficients produced by the ideal thresholding function cannot take values in this range (e) The ideal soft thresholding function ($\gamma = \infty$, $\alpha = 1$, $\lambda = 1$) and the (f) corresponding cost function.



(a) (b)

Figure 3: (a) The top 200 coefficients from a BPDN solver sorted by magnitude. (b) The same coefficients, sorted according to the magnitude ordering of the SLCA coefficients. While there is a gross decreasing trend noticeable, the largest SLCA coefficients are not in the same locations as the largest BPDN coefficients. While the solutions have equivalent energy functions, the two sets of coefficients differ significantly.

3.3 Special case: Soft-thresholding locally competitive algorithm (SLCA)

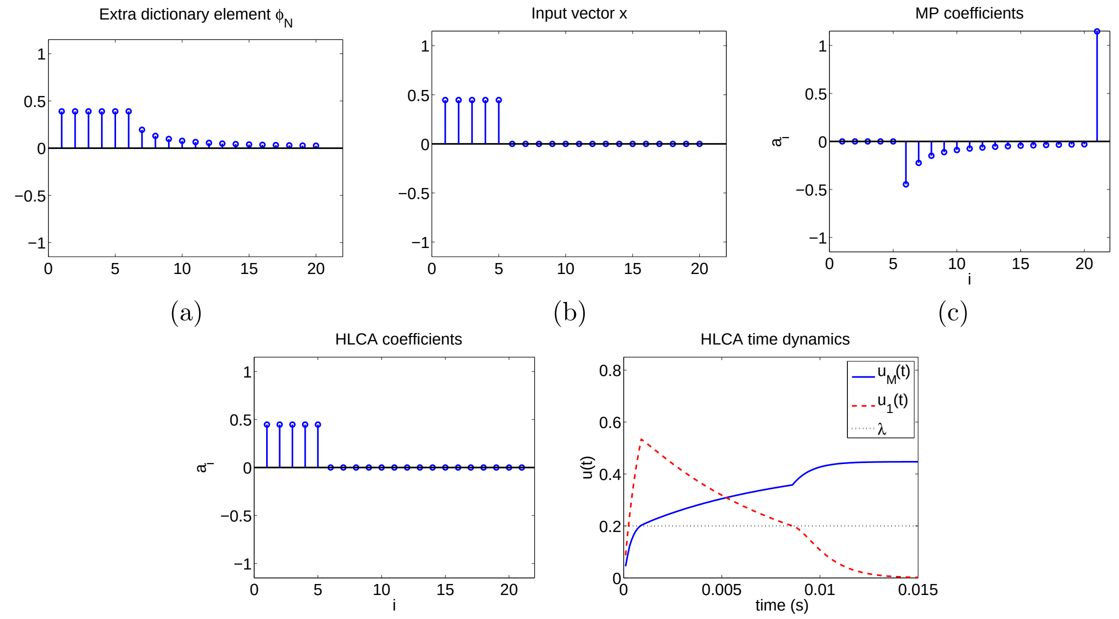
As we see in Section 3.2, LCAs can optimize a variety of different sparsity measures depending on the choice of thresholding function. One special case is the soft thresholding function, corresponding to $\alpha = 1$ and shown graphically in Figures 2e and 2f. The soft-thresholding locally competitive algorithm (SLCA) applies the ℓ^1 norm as a cost function on the active coefficients,

$$C_{(1,\infty,\lambda)}(a_m) = |a_m|.$$

Thus, the SLCA is simply another solution method for the general BPDN problem described in Section 2, and simulations confirm that the SLCA does indeed find solutions with values of $E(t)$ equivalent to the solutions produced by standard interior-point based methods. Despite minimizing the same convex energy function, SLCA and BPDN solvers may find different sets of coefficients, as illustrated in Figure 3. This may be due either to the non-uniqueness of a BPDN solution,[4] or because the numerical approximations have not quite converged to a global optimum in the alloted computation time. The connection between soft-thresholding and BPDN is well-known in the case of orthonormal dictionaries (Chen et al., 2001), and recent results (Elad, 2006) have given some justification for using soft-thresholding in overcomplete dictionaries. The SLCA provides another formal connection between the soft-thresholding function and the ℓ^1 cost function.

Though BPDN uses the ℓ^1-norm as its sparsity penalty, we often expect many of the resulting coefficients to be identically zero (especially when $M \gg N$). However, most numerical methods (including direct gradient descent and interior point solvers) will drive coefficients *toward* zero but will never make them identically zero. While an ad hoc threshold could be applied to the results of a BPDN solver, the SLCA has the advantage of incorporating a natural thresholding function that keeps coefficients identically zero during the computation unless they become active. In other words, while BPDN solvers often start with many non-zero coefficients and try to force coefficients down, the SLCA starts with all coefficients equal to zero and only lets a few grow up. This advantage is especially important for neural systems that must expend energy for non-zero values throughout the entire computation.

[4] With an overcomplete dictionary, it is technically possible to have a BPDN energy function that is convex (guaranteeing all local minima are global minima) but not strictly convex (guaranteeing that a global minimum is unique).



(a) (b) (c)

(d) (e)

Figure 4: (a) The dictionary in this example has one "extra" element that consists of decaying combinations of all other dictionary elements. (b) The input vector has a sparse representation in just a few dictionary elements. (c) MP initially chooses the "extra" dictionary element, preventing it from finding the optimally sparse representation (coefficients shown after 100 iterations). (d) In contrast, the HLCA system finds the optimally sparse coefficients. (e) The time-dynamics of the HLCA system illustrate its advantage. The "extra" dictionary element is the first node to activate, followed shortly by the nodes corresponding to the optimal coefficients. The collective inhibition of the optimal nodes causes the "extra" node to die away.

3.4 Special case: Hard-thresholding locally competitive algorithm (HLCA)

Another important special case is the hard thresholding function, corresponding to $\alpha = 0$ and shown graphically in Figures 2c and 2d. Using the relationship in (7), we see that this hard-thresholding locally competitive algorithm (HLCA) applies an ℓ^0-like cost function by using a constant penalty regardless of magnitude,

$$C_{(0,\infty,\lambda)}(a_m) = \frac{\lambda}{2}\, I(|a_m| > \lambda)\,,$$

where $I(\cdot)$ is the indicator function evaluating to 1 if the argument is true and 0 if the argument is false. Unlike the SLCA, the HLCA energy function is not convex and the system will only find a local minimum of the energy function $E(t)$. As with the SLCA, the HLCA also has connections to known sparse approximation principles. If node m is fully charged, the inhibition signal it sends to other nodes would be exactly the same as the update step when the m^{th} node is chosen in the MP algorithm. However, due to the continuous competition between nodes before they are fully charged, the HLCA is not equivalent to MP in general.

As a demonstration of the power of competitive algorithms over greedy algorithms such as MP, consider a canonical example used to illustrate the shortcomings of iterative greedy algorithms (Chen et al., 2001; DeVore and Temlyakov, 1996). For this example, specify a positive integer $K < N$ and construct a dictionary $\mathcal{D}$ with $M = N + 1$ elements to have the following form:

$$\boldsymbol{\phi}_m = \begin{cases} \boldsymbol{e}_m & \text{if } m \le N \\ \sum_{n=1}^{K} \kappa \boldsymbol{e}_n + \sum_{n=K+1}^{N} \left(\kappa/(n-K)\right) \boldsymbol{e}_n & \text{if } m = N+1, \end{cases}$$

where $\boldsymbol{e}_m$ is the canonical basis element (i.e., it contains a single 1 in the m^{th} location) and κ is a constant to make the vectors have unit norm. In words, the dictionary includes the canonical basis along with one "extra" element that is a decaying combination of all other elements (illustrated in Figure 4, with $N = 20$ and $K = 5$). The input signal is sparsely represented in the first K dictionary elements, $\boldsymbol{s} = \sum_{m=1}^{K} \frac{1}{\sqrt{K}} \boldsymbol{e}_m$. The first MP iteration chooses $\boldsymbol{\phi}_M$, introducing a residual with decaying terms. Even though $\boldsymbol{s}$ has an exact representation in K elements, MP iterates forever trying to atone for this bad initial choice. In contrast, the HLCA initially activates the M^{th} node but uses the collective inhibition from nodes $1, \ldots, K$ to suppress this node and calculate the optimal set of coefficients. While this pathological example is unlikely to exactly occur in natural signals, it is often used as a criticism of greedy methods to demonstrate their shortsightedness. We mention it here to demonstrate the flexibility of LCAs and their differences from pure greedy algorithms.

4 LCA system properties

To be a physically viable sparse coding mechanism, LCAs must exhibit several critical properties: the system must remain stable under normal operating conditions, the system must produce sparse coefficients that represent the stimulus with low error, and coefficient sequences must exhibit regularity in response to time-varying inputs. In this section we show that LCAs exhibit good characteristics in each of these three areas. We focus our analysis on the HLCA both because it yields the most interesting results and because it is notationally the cleanest to discuss. In general, the analysis principles we use will apply to all LCAs through straightforward (through perhaps laborious) extensions.

4.1 Stability

Any proposed neural system must remain well-behaved under normal conditions. While linear systems theory has an intuitive notion of stability that is easily testable (Franklin et al., 1986), no such unifying concept of stability exists for nonlinear systems (Khalil, 2002). Instead, nonlinear systems are characterized in a variety of ways, including their behavior near an equilibrium point $\boldsymbol{u}^*$ where $f(\boldsymbol{u}^*) = 0$ and their input-output relationship.

The various stability analyses of Sections 4.1.1 and 4.1.2 depend on a common criterion. Define $\mathcal{M}_{\boldsymbol{u}(t)} \subseteq [1, \ldots, M]$ as the set of nodes that are above threshold in the internal state vector $\boldsymbol{u}(t)$, $\mathcal{M}_{\boldsymbol{u}(t)} = \{m : |u_m(t)| \geq \lambda\}$. We say that the LCA meets the *stability criterion* if t the set of active vectors $\{\boldsymbol{\phi}_m\}_{m \in \mathcal{M}_{\boldsymbol{u}(t)}}$ is linearly independent. It makes some intuitive sense that this condition is important: if a collection of linearly dependent nodes are active simultaneously, the nodes could have growing coefficients but no net effect on the reconstruction.

The system is likely to satisfy the stability criterion eventually under normal operating conditions for two reasons. First, small subsets of dictionary elements are unlikely to be linearly dependent unless the dictionary is designed with this property (e.g., (Tropp, 2006)). Second, sparse coding systems are actively trying to select dictionary subsets so that they can use many *fewer* coefficients than the dimension of the signal space, $\left|\mathcal{M}_{\boldsymbol{u}(t)}\right| \ll N \ll M$. While the LCA lateral inhibition signals discourage linearly dependent sets from activating, the stability criterion can be violated when a collection of nodes becomes active too quickly, before inhibition can take effect. In simulation, we have observed this situation when the threshold is too low compared to the system time constant. As discussed below, the stability criterion amounts to a sufficient (but not necessary) condition for good behavior, and we have never empirically observed the simulated system becoming unstable even when this condition is transiently violated.

4.1.1 Equilibrium points

In a LCA presented with a static input, we look to the steady-state response (where $\dot{\boldsymbol{u}}(t) = 0$) to determine the coefficients. The character of the equilibrium points $\boldsymbol{u}^*$ ($f(\boldsymbol{u}^*) = 0$) and the system's behavior in a neighborhood around an equilibrium point provides one way to ensure that a system is well-behaved. Consider the ball around an equilibrium point $B_\epsilon(\boldsymbol{u}^*) = \{\boldsymbol{u} : ||\boldsymbol{u} - \boldsymbol{u}^*|| < \epsilon\}$. Nonlinear system analysis typically asks an intuitive question: if the system is perturbed within this ball, does it then run away, stay where it is, or get attracted back? Specifically, a system is said to be *locally asymptotically stable* (Bacciotti and Rosier, 2001) at an equilibrium point $\boldsymbol{u}^*$ if one can specify $\epsilon > 0$ such that

$$\boldsymbol{u}(0) \in B_\epsilon(\boldsymbol{u}^*) \implies \lim_{t\to\infty} \boldsymbol{u}(t) = \boldsymbol{u}^*.$$

Previous research (Cohen and Grossberg, 1983; Yang and Dillon, 1994; Li et al., 1988) has used the tools of Lyapunov functions (Khalil, 2002) to study a Hopfield network (Hopfield, 1984) similar to the LCA architecture. However, all of these analyses make assumptions that do not encompass the ideal thresholding functions used in the LCAs (e.g., they are continuously differentiable and/or monotone increasing). In Appendix D.1 we show that as long as the stability criterion is met, the HLCA:

- has a finite number of equilibrium points;
- has equilibrium points that are almost certainly isolated (no two equilibrium points are arbitrarily close together); and
- is almost certainly locally asymptotically stable for every equilibrium point.

The conditions that hold "almost certainly" are true as long as none of the equilibria have components identically equal to the threshold, ($u_m^* \neq \lambda, \forall m$), which holds with overwhelming probability. With a finite number of isolated equilibria, we can be confident that the HLCA steady-state response is a distinct set of coefficients representing the stimulus. Asymptotic stability also implies a notion of robustness, guaranteeing that the system will remain well-behaved even under perturbations (Theorems 2.8 and 2.9 in (Bacciotti and Rosier, 2001)).

4.1.2 Input-output stability

In physical systems it is important that the energy of both internal and external signals remain bounded for bounded inputs. One intuitive approach to ensuring output stability is to examine the energy function E. We show in Appendix D.2 that for non-decreasing threshold functions, the energy function is non-increasing ($\frac{d}{dt}E(t) \leq 0$) for fixed inputs. While this is encouraging, it does not guarantee input-output stability. To appreciate this effect, note that the HLCA cost function is constant for nodes above threshold — nothing explicitly keeps a node from growing without bound once it is active.

While there is no universal input-output stability test for general nonlinear systems, we observe that the LCA system equation is linear and fixed until a unit crosses threshold. A branch of control theory specifically addresses these *switched systems* (Decarlo et al., 2000). Results from this field indicate that input-output stability can be guaranteed if the individual linear subsystems are stable, and the system does not switch "too fast" between these subsystems (Hespanha and Morse, 1999). In Appendix D.2 we give a precise mathematical statement of this input-output stability and show that the HLCA linear subsystems are individually stable if and only if the stability criterion is met. Therefore, the HLCA is input-output stable as long as nodes are limited in how fast they can change states. We expect that an infinitely fast switching condition is avoided in practice either by the physical principles of the system implementation or through an explicit hysteresis in the thresholding function.

4.2 Sparsity and representation error

Viewing the sparse approximation problem through the lens of rate-distortion theory (Berger, 1971), the most powerful algorithm produces the lowest reconstruction MSE for a given sparsity. When the sparsity measure is the ℓ^1 norm, the problem is convex and the SLCA produces solutions with equivalent ℓ^1-sparsity to interior point BPDN solvers. Despite the analytic appeal of the ℓ^1 norm as a sparsity measure, many systems concerned with energy minimization (including neural systems) likely have an interest in minimizing the ℓ^0 norm of the coefficients. The HLCA is appealing because of its ℓ^0-like sparsity penalty, but this objective function is not convex and the HLCA may find a local minimum. We will show that while HLCA cannot guarantee the ℓ^0 sparsest solution, it produces coefficients that demonstrate comparable sparsity to MP for natural images.

Insight about the HLCA reconstruction fidelity comes from rewriting the LCA system equation

$$\dot{\boldsymbol{u}}(t) = \frac{1}{\tau}\left[\Phi^t\left(\boldsymbol{s}(t) - \widehat{\boldsymbol{s}}(t)\right) - \boldsymbol{u}(t) + T_{(\alpha,\infty,\lambda)}(\boldsymbol{u}(t))\right]. \tag{10}$$

For a constant input, HLCA equilibrium points ($\dot{\boldsymbol{u}}(t) = 0$) occur when the residual error is orthogonal to active nodes and balanced with the internal state variables of inactive nodes.

$$\langle \boldsymbol{\phi}_m, \boldsymbol{s}(t) - \widehat{\boldsymbol{s}}(t)\rangle = \begin{cases} u_m(t) & \text{if } |u_m| \leq \lambda \\ 0 & \text{if } |u_m| > \lambda \end{cases}.$$

Therefore, when HLCA converges the coefficients will perfectly reconstruct the component of the input signal that projects onto the subspace spanned by the final set of active nodes. Using standard results from frame theory (Christensen, 2002), we can bound the HLCA reconstruction MSE in terms of the set of inactive nodes

$$||\boldsymbol{s}(t) - \widehat{\boldsymbol{s}}(t)||^2 \leq \frac{1}{\eta_{\min}} \sum_{m \notin \mathcal{M}_{\boldsymbol{u}(t)}} |\langle \boldsymbol{\phi}_m, \boldsymbol{s}(t) - \widehat{\boldsymbol{s}}(t)\rangle|^2 \leq \frac{\left(M - |\mathcal{M}_{\boldsymbol{u}(t)}|\right)\lambda^2}{\eta_{\min}},$$

where $\eta_{\min}$ is the minimum eigenvalue of $(\Phi\Phi^t)$.

Though the HLCA is not guaranteed to find the globally optimal ℓ^0 sparsest solution we must ensure that it does not produce unreasonably non-sparse solutions. While the system nonlinearity makes it impossible to analytically determine the LCA steady-state coefficients, it is possible to rule out some coefficients as *not* being possible. For example, let $\mathcal{M} \subseteq [1, \ldots, M]$ be an arbitrary set of active coefficients. Using linear systems theory we can calculate the steady-state response $\widetilde{\boldsymbol{u}}^{\mathcal{M}} = \lim_{t\to\infty} \boldsymbol{u}(t)$ assuming that $\mathcal{M}$ stays fixed. If $|\widetilde{u}_m^{\mathcal{M}}| < \lambda$ for any $m \in \mathcal{M}$ or if $|\widetilde{u}_m^{\mathcal{M}}| > \lambda$ for any $m \notin \mathcal{M}$, then $\mathcal{M}$ cannot describe the set of active nodes in the steady-state response and we call it *inconsistent*. We show in Appendix E that when the stability criterion is met, the following statement is true for the HLCA: *If* $\boldsymbol{s} = \boldsymbol{\phi}_m$*, then any set of active coefficients* $\mathcal{M}$ *with* $m \in \mathcal{M}$ *and* $|\mathcal{M}| > 1$ *is inconsistent.* In other words, the HLCA may use the m^{th} node or a collection of other nodes to represent $\boldsymbol{s}$, but it cannot use a combination of both. This result extends intuitively beyond one-sparse signals: each component in an optimal decomposition is represented by either the optimal node or another collection of nodes, but not both. While not necessarily finding the optimal representation, the system does not needlessly employ both the optimal and extraneous nodes.

We have also verified numerically that the LCAs achieve a combination of error and sparsity comparable with known methods. We employed a dictionary consisting of the bandpass band of a steerable pyramid (Simoncelli and Freeman, 1995)[5] with one level and four orientation bands (i.e., the dictionary is approximately four times overcomplete with 4096 elements). Image patches (32×32) were selected at random from a standard set of test images. The selected image patches were decomposed using the steerable pyramid and reconstructed using just the bandpass band.[6] The bandpass image patches were also normalized to have unit energy. Each image patch was used as the fixed input to the LCA system equation (5) using either a soft or hard thresholding function (with variable threshold values) and with a biologically plausible membrane time constant of $\tau = 10$ ms (Dayan and Abbott, 2001). We simulated the system using a simple Euler's method approach (i.e., first order finite difference approximation) (Süli and Mayers, 2003) with a time step of 1 ms.

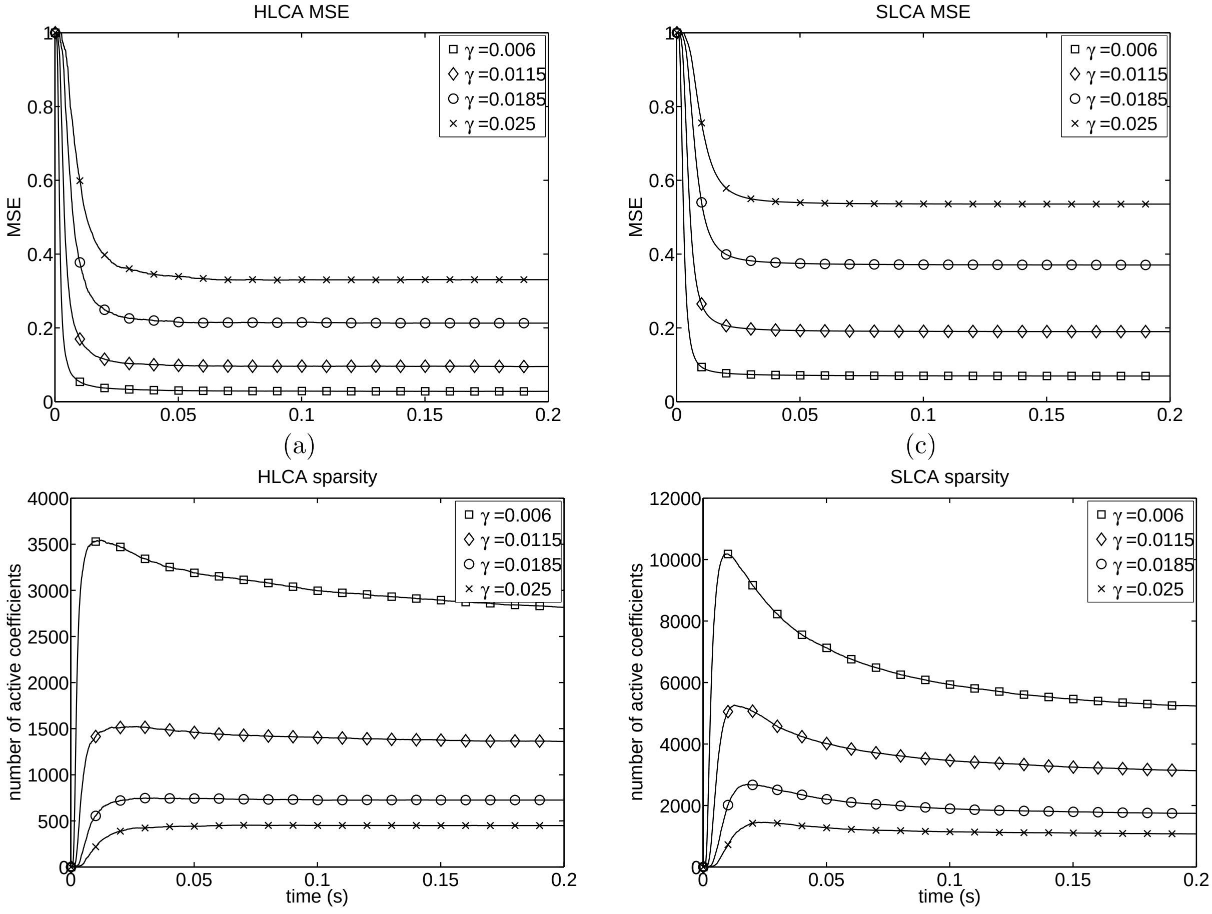
Figure 5 shows the time evolution of the reconstruction MSE and ℓ^0 sparsity for SLCA and HLCA responding to an individual image, and Figure 6 shows the mean steady-state tradeoff between ℓ^0 sparsity and MSE. For comparison, we also plotted the results obtained from using MP,[7] a standard BPDN interior-point solver[8] followed by thresholding to enforce ℓ^0 sparsity (denoted "BPDNthr") and SLCA with the

[5] We used the implementation of the steerable pyramid given in the code available at `http://www.cns.nyu.edu/~eero/STEERPYR/` with the "circular" boundary handling option.

[6] We eliminate the lowpass band because it accounts for a large fraction of the total image energy in just a few dictionary elements and it is unlikely that much gain could be achieved by sparsifying these coefficients.

[7] We iterated MP until it had the same MSE as the LCA implementation to compare the sparsity levels of the results.

[8] We used the implementation of a primal-dual BPDN solver given in the code available at `http://sparselab.stanford.edu/`.



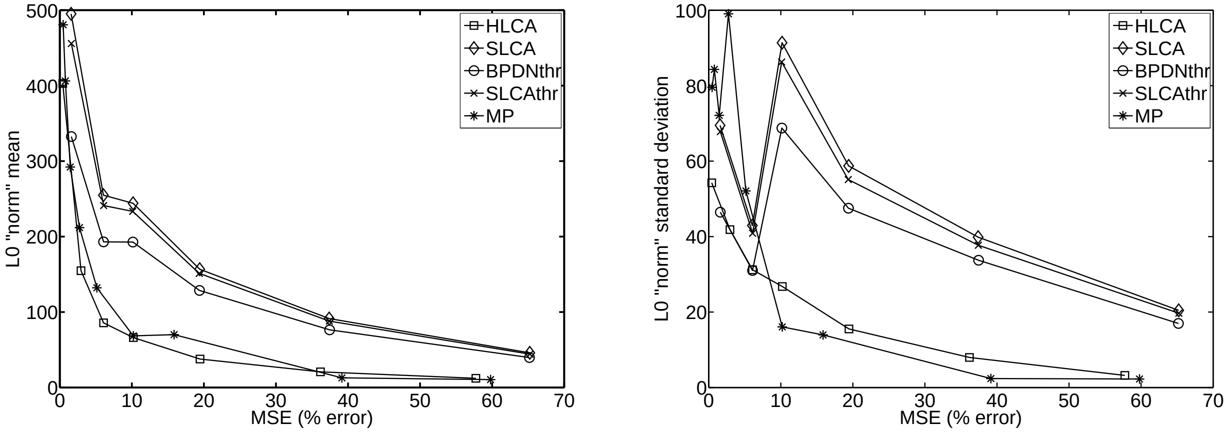
(a)

(c)

(b)

(d)

Figure 5: The time response of the HLCA and SLCA ($\tau = 10$ ms) for a single fixed (32×32) image patch with a dictionary that is four time overcomplete with 4096 elements. (a) The MSE decay and (b) the ℓ^0 sparsity for HLCA. (c) The MSE decay and (d) the ℓ^0 sparsity for SLCA. The error converges within 1-2 time constants and the sparsity often approximately converges within 3-4 time constants. In some cases sparsity is reduced with a longer running time.



(a) (b)

Figure 6: Mean tradeoff between MSE and ℓ^0-sparsity for normalized (32×32) patches from a standard set of test images. The dictionary was four times overcomplete with 4096 elements. For a given MSE range, we plot the mean (a) and standard deviation (b) of the ℓ^0 sparsity.

same threshold applied (denoted "SLCAthr"). Most importantly, note that the HLCA and MP are almost identical in their sparsity-MSE tradeoff. Though the connections between the HLCA and MP were pointed out in Section 3.4, these are very different systems and there is no reason to expect them to produce the same coefficients. Additionally, note that the SLCA is producing coefficients that are nearly as ℓ^0-sparse as what we can achieved by oracle thresholding the results of a BPDN solver even though the SLCA keeps most coefficients zero throughout the calculation.

4.3 Time-varying stimuli

4.3.1 Inertia

Biological sensory systems are faced with constantly changing stimuli due to both external movement and internal factors (e.g., organism movement, eye saccades, etc.). As discussed in Section 2.2, sparse codes with temporal variations that also reflect the smooth nature of the changing stimulus would be easier for higher level systems to understand and interpret. However, approximation methods that only optimize sparsity at each time step (especially greedy algorithms) can produce "brittle" representations that change dramatically with relatively small stimulus changes. In contrast, LCAs naturally produce smoothly changing outputs in response to smoothly changing time-varying inputs. Assuming that the system time constant τ is faster than the temporal changes in the stimulus, the LCA will evolve to capture the stimulus change and converge to a new sparse representation. While local minima in an energy function are typically problematic, the LCAs can use these local minima to find coefficients that are "close" to their previous coefficients even if they are not optimally sparse. While permitting suboptimal coefficient sparsity, this property allows the LCA to exhibit inertia that smoothes the coefficient sequences.

The inertia property exhibited in LCAs can be seen by focusing on a single node in the system equation (10):

$$\dot{u}_m(t) = \frac{1}{\tau} \begin{cases} \langle \boldsymbol{\phi}_m, (\boldsymbol{s}(t) - \widehat{\boldsymbol{s}}(t)) \rangle - u_m(t) & \text{when } |u_m(t)| < \lambda \\ \langle \boldsymbol{\phi}_m, (\boldsymbol{s}(t) - \widehat{\boldsymbol{s}}(t)) \rangle - \alpha\lambda & \text{when } |u_m(t)| \geq \lambda. \end{cases}$$

A new residual signal drives the coefficient higher but suffers an additive penalty. Inactive coefficients suffer an increasing penalty as they get closer to threshold while active coefficients only suffer a constant penalty $\alpha\lambda$ that can be very small (e.g., the HLCA has $\alpha\lambda = 0$). This property induces a "king of the hill" effect:

when a new residual appears, active nodes move virtually unimpeded to represent it while inactive nodes are penalized until they reach threshold. This inertia encourages inactive nodes to remain inactive unless the active nodes cannot adequately represent the new input.

To illustrate this inertia, we applied the LCAs to a sequence of 144 × 144 pixel, bandpass filtered, normalized frames from the standard "foreman" test video sequence with the same experimental setup described in Section 4.2. The LCA input is switched to the next video frame every (simulated) 1/30 seconds. The results are shown in Figure 7, along with comparisons to MP and BPDN applied independently on each frame. The changing coefficient locations are nodes that either became active or inactive at each frame. Mathematically, the number of changing coefficients at frame n is: $\left|\mathcal{M}_{\boldsymbol{u}(n-1)} \oplus \mathcal{M}_{\boldsymbol{u}(n)}\right|$, where $\oplus$ is the "exclusive OR" operator and $\boldsymbol{u}(n)$ are the internal state variables at the end of the simulation for frame n.

This simulation highlights that the HLCA uses approximately the same number of active coefficients as MP but chooses coefficients that more efficiently represent the video sequence. The HLCA is significantly more likely to re-use active coefficient locations from the previous frame without making significant sacrifices in the sparsity of the solution. This difference is highlighted when looking at the ratio of the number of changing coefficients to the number of active coefficients, $\left|\mathcal{M}_{\boldsymbol{u}(n-1)} \oplus \mathcal{M}_{\boldsymbol{u}(n)}\right| / \left|\mathcal{M}_{\boldsymbol{u}(n)}\right|$. MP has a ratio of 1.7, meaning that MP is finding almost an entirely new set of active coefficient locations for each frame. The HLCA has a ratio of 0.5, meaning that it is changing approximately 25% of its coefficient locations at each frame. SLCA and BPDNthr have approximately the same performance, with regularity falling between HLCA and MP. Though the two systems can calculate different coefficients, the convexity of the energy function appears to be limiting the coefficient choices enough so that SLCA cannot smooth the coefficient time series substantially more than BPDNthr.

4.3.2 Markov state transitions

The simulation results indicate that the HLCA is producing time series coefficients that are much more regular than MP. This regularity is visualized in Figure 9 by looking at the time-series of example HLCA and MP coefficients. Note that though the two coding schemes produce roughly the same number of non-zero entries, the HLCA does much better than MP at clustering the values into consecutive runs of positive or negative values. This type of smoothness better reflects the regularity in the natural video sequence input.

We can quantify this increased regularity by examining the Markov state transitions. Specifically, each coefficient time-series is Markov chain (Norris, 1997) with three possible states at frame n:

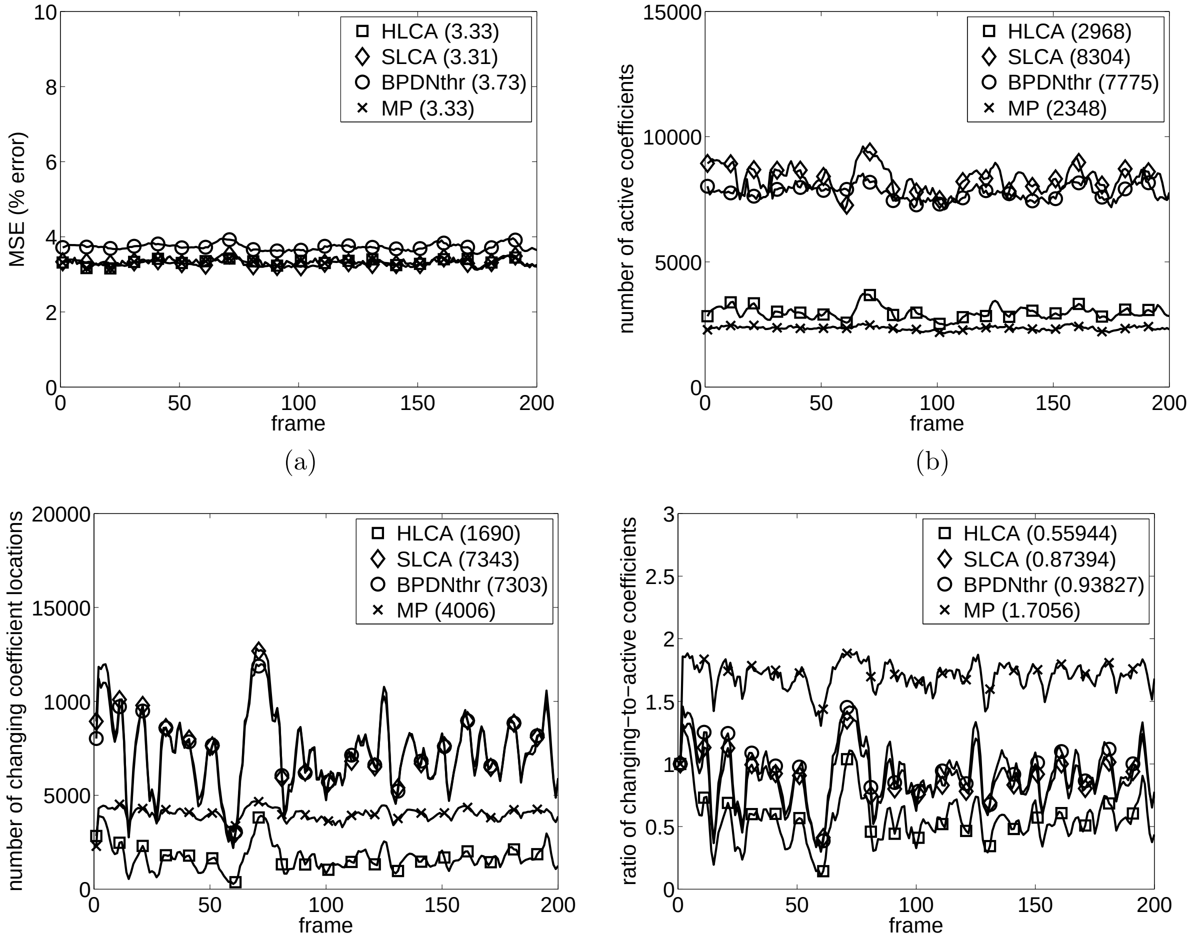
$$\sigma_m(n) = \begin{cases} - & \text{if } \ u_m(n) < -\lambda \\ 0 & \text{if } \ -\lambda \le u_m(n) \le \lambda \\ + & \text{if } \ u_m(n) > \lambda. \end{cases}$$

Figure 8 shows the marginal probabilities $P(\cdot)$ of the states and the conditional probabilities $P(\cdot|\cdot)$ of moving to a state given the previous state. The HLCA and MP are equally likely to have non-zero states, but the HLCA is over five times more likely than MP to have a positive coefficient stay positive ($P(+|+)$). Also, though the absolute probabilities are small, MP is roughly two orders of magnitude more likely to have a coefficient swing from positive to negative ($P(-|+)$) and vice-versa ($P(-|+)$).

To quantify the regularity of the active coefficient locations we calculate the entropy (Cover and Thomas, 1991) of the coefficient states at frame n conditioned on the coefficient states at frame $(n-1)$:

$$\begin{aligned} H(\sigma_m(n)\,|\sigma_m(n-1)) = & -P(+)\left[P(-|+) + P(0|+) + P(+|+)\right] \\ & - P(0)\left[P(-|0) + P(0|0) + P(+|0)\right] \\ & - P(-)\left[P(-|-) + P(0|-) + P(+|-)\right], \end{aligned} \tag{11}$$

plotted in Figure 9. This conditional entropy indicates how much uncertainty there is about the status of the current coefficients given the coefficients from the previous frame. Note that the conditional entropy for MP is almost double the entropy for the HLCA, while SLCA is again similar to BPDNthr. The principle contributing factor to the conditional entropy appears to be the probability a non-zero node remains in the



(a)

(b)

(c)

(d)

Figure 7: The HLCA and SLCA systems simulated on 200 frames of the "foreman" test video sequence. For comparison, MP coefficients and thresholded BPDN coefficients are also shown. Average values for each system are noted in the legend. (a) Per-frame MSE for each coding scheme, designed to be approximately equal. (b) The number of active coefficients in each frame. (c) The number of changing coefficient locations for each frame, including the number of inactive nodes becoming active and the number of active nodes becoming inactive. (d) The ratio of changing coefficients to active coefficients. A ratio near 2 (such as with MP) means that almost 100% of the coefficient locations are new at each frame. A ratio near 0.5 (such as with HLCA) means that approximately 25% of the coefficients are new at each frame.

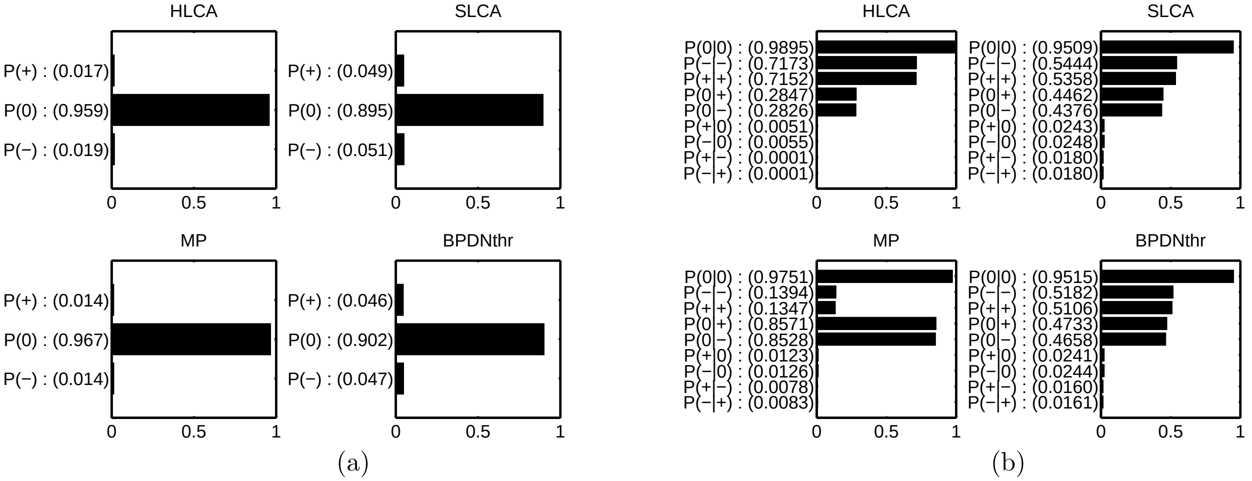


Figure 8: (a) The marginal probabilities denoting the fraction of the time coefficients spent in the three states: negative, zero and positive ($-$, 0, and $+$). (b) The transition probabilities denoting the probability of a node in one state transitioning to another state on the next frame. For example, $P(0|+)$ is the probability that a node with an active positive coefficient will be inactive (i.e., zero) in the next frame.

same state (i.e., $P(+|+)$ and $P(-|-)$). To illustrate, Figure 9 shows the change in conditional entropy is almost linear with varying $P(+|+)$ (assuming $P(-|-) = P(+|+)$ and all other transition probabilities are kept fixed).

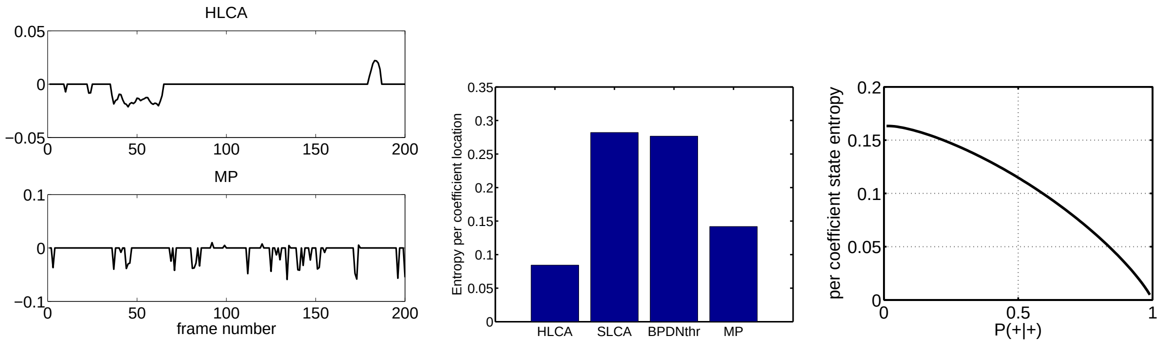
The substantial decrease in the conditional entropy for the HLCA compared to MP quantifies the increased regularity in time-series coefficients due to the inertial properties of the LCAs. The HLCA in particular encourages coefficients to maintain their present state (i.e., active or inactive) if it is possible to find an adequate stimulus representation. While some sparsity may be sacrificed in this strategy, the smoothness induced in the coefficients by grouping active states together in time better reflects the character of the natural time-varying stimuli and could be useful for higher-level computations.

5 Conclusions and future work

Sparse approximation is an important paradigm in neural coding, though plausible mechanisms to achieve these codes have remained unknown. We have proposed an architecture for a locally competitive algorithm that solves a family of sparse approximation problems (including BPDN as a special case). These LCAs may be readily implemented using a parallel network of simple analog circuit elements that could potentially be mapped onto the neural circuitry of sensory cortical areas such as V1. Though these LCA systems are nonlinear, we have shown that they are well-behaved under nominal operating conditions.

While the LCA systems (other than SLCA) are not generally guaranteed to find a globally optimal solution to their energy function, we have proven that the systems will be efficient in a meaningful sense. The SLCA system produces coefficients with sparsity levels comparable to BPDN solvers, but uses a natural physical implementation that is more energy efficient (i.e., it uses fewer non-zero inhibition signals between nodes). Perhaps most interestingly, the HLCA produces coefficients with almost identical sparsity as MP. This is significant because greedy methods such as MP are widely used in signal processing practice because of their efficiency, but HLCA offers a much more natural neural implementation.

LCAs are particularly appropriate for time-varying data such as video sequences. The LCA ODE not only encourages sparsity but also introduces an inertia into the coefficient time-series that we have quantified using both raw counts of changing coefficient location and through the conditional entropy of the coefficient states. By allowing slightly suboptimal sparsity in exchange for more regularity in the set of active coefficients, the



(a) (b) (c)

Figure 9: (a) An example time-series coefficient for the HLCA and MP (top and bottom, respectively) encodings for the test video sequence. HLCA clusters non-zero entries together into longer runs while MP switches more often between states. (b) The empirical conditional entropy of the coefficient states (−,0,+) during the test video sequence. (c) The conditional entropy is calculated analytically while varying $P(+|+)$ and equalizing all other transition probabilities to the values seen in HLCA and MP. The tendency of a system to group non-zero states together is the most important factor in determining the entropy.

LCAs produce smoother coefficient sequences that better reflect the structure of the time-varying stimulus. This property could prove valuable for higher levels of analysis that are trying to interpret the sensory scene from a set of sparse coefficients. Coefficient regularity could presumably be further improved by incorporating models of spatial dependencies from higher-level processing areas or models of temporal dynamics in the representation of natural scenes. Note that this approach differs from previous approaches in which the dynamics are modeled by the dictionary itself (Olshausen, 2003), as that approach also suffers from producing erratic coefficient sequences that may be difficult to interpret.

The current limitations of neurophysiological recording mean that exploring the sparse coding hypothesis must rely on testing specific proposed mechanisms. Though the LCAs we have proposed appear to map well to known neural architectures, they still lack the biophysical detail necessary to be experimentally testable. We will continue to build on this work by mapping these LCAs to a detailed neurobiological population coding model that can produce verifiable predictions. Furthermore, the combination of sparsity and regularity induced in LCA coefficients may serve as a critical front-end stimulus representation that enables visual perceptual tasks, including pattern recognition, source separation and object tracking.

By using simple computational primitives, LCAs also have the benefit of being implementable in analog hardware. An imaging system using VLSI to implement LCAs as a data collection front end has the potential to be extremely fast and energy efficient. Instead of digitizing all of the sensed data and using digital hardware to run a compression algorithm, analog processing would compress the data into sparse coefficients *before* digitization. In this system, time and energy resources would only be spent digitizing coefficients that are a critical component in the signal representation.

Acknowledgments

This work was funded by grants NGA MCA 015894-UCB, NSF IIS-06-25223 and CCF-0431150, DARPA/ONR N66001-06-1-2011 and N00014-06-1-0610, ONR N00014-06-1-0769 and N00014-06-1-0829, AFOSR FA9550-04-1-0148, and the Texas Instruments DSP Leadership University Program. The authors would like to thank the anonymous reviewers who provided comments that improved the manuscript. The authors are also grateful to the authors of the “SparseLab” and “matlabPyrTools” software toolboxes for making them available for public use.

A Detailed comparison of LCAs to closely related methods

Several recent sparse approximation methods are closely related to a discrete time approximation of the LCA system described in Section 3.4 when applied to a fixed stimulus. To present a detailed comparison between these methods, we first introduce the binary thresholding function

$$T_B(x) = \begin{cases} 0 & \text{if} \quad x < 0 \\ 1 & \text{if} \quad x \geq 0 \end{cases} .$$

Note the correspondence to the ideal hard thresholding function, $T_{(0,\infty,\lambda)}(x) = xT_B(x - \lambda)$. To make a direct comparison between HLCA and these iterative methods, we use this notation to write a discrete time approximation to the HLCA system equation for a single node at the discrete time steps $\{t_k\}$

$$\begin{aligned} u_m(t_{k+1}) &= (1 - \Delta)u_m(t_k) + \Delta \left[b_m - \sum_{n \neq m} \langle \boldsymbol{\phi}_m, \boldsymbol{\phi}_n \rangle u_n(t_k)\, T_B(u_n(t_k) - \lambda) \right], \qquad (12) \\ a_m(t_{k+1}) &= u_m(t_{k+1})\, T_B(u_m(t_{k+1}) - \lambda), \end{aligned}$$

where Δ is a small constant incorporating both the time constant of the dynamical system and the step size of the discrete time approximation.

The sparse-set coding network (SSCN) in (Rehn and Sommer, 2007) uses an iterative method where the update equation for a single node is given by

$$\begin{aligned} u_m(t_{k+1}) &= b_m - \sum_{n \neq m} \langle \boldsymbol{\phi}_m, \boldsymbol{\phi}_n \rangle b_n T_B \left(u_n(t_k) - \frac{\lambda}{b_n} \right), \qquad (13) \\ a_m(t_{k+1}) &= u_m(t_{k+1})\, T_B \left(u_m(t_{k+1}) - \frac{\lambda}{b_n} dv \right). \end{aligned}$$

The SSCN can be interpreted as a network implementation of the optimized orthogonal matching pursuit (OOMP) algorithm (Rebollo-Neira and Lowe, 2002). OOMP is a greedy method that at each iteration selects the dictionary element that minimizes the residual error when the optimal coefficients are calculated for the selected dictionary elements. The resulting SSCN update equation is very similar to the HLCA update term (i.e., the bracketed term) in (12), with three notable differences:

- the HLCA uses a fixed threshold λ for every node whereas the SSCN scales the threshold for each node by that node's driving term $\frac{\lambda}{b_n}$;
- the HLCA uses a graded inhibition term for active coefficients (i.e., the magnitude of the inhibition from node m scales with $u_m(t_k)$) whereas the SSCN uses a fixed inhibition magnitude for active nodes based on the driving term b_m; and
- the HLCA uses a charging circuit approach whereas the SSCN update directly modifies the current coefficient values.

It is not clear how the variable threshold in the SSCN or the graded inhibition in the HLCA affects the character of the resulting coefficients. While the charging circuit approach taken by the HLCA may appear to be a small implementational difference, it represents a different strategy for calculating coefficients. This can be seen by considering the first step a system would take starting from a set of zero-valued coefficients, $a_m(0) = 0, \forall m$. The HLCA would produce a small change (of magnitude Δ) based on the driving terms b_m. When some of these coefficients grow just above threshold (not all the way up to b_m) they can begin to inhibit to prevent other coefficients from becoming active. In the SSCN, the first step produces coefficient values that are equal to the full driving values b_m passed through a non-uniform hard threshold (i.e., they are either 0 or b_m). Thus, unless the threshold is initially very high, SSCN will need to prune away coefficients from the thresholded linear projections.

The algorithm in (Kingsbury and Reeves, 2003) uses an iterative method where the update equation for a single node is given by

$$\begin{aligned} u_m(t_{k+1}) &= u_m(t_k)\, T_B(u_m(t_k)-\lambda) + \Delta \left[b_m - \sum_{n\neq m} \langle \widetilde{\boldsymbol{\phi}}_n, \boldsymbol{\phi}_m \rangle u_n(t_k)\, T_B(u_n(t_k)-\lambda) \right], \qquad (14) \\ a_m(t_{k+1}) &= u_m(t_k)\, T_B(u_m(t_k)-\lambda)\,, \end{aligned}$$

where $u_m(t_0) = b_m$ and $\widetilde{\boldsymbol{\phi}}_m$ is the canonical dual vector for the dictionary element $\boldsymbol{\phi}_m$. The set of canonical dual vectors are calculated by performing a matrix inverse and essentially capture the calculation of the pseudoinverse operator. This update equation is also very similar to the HLCA system equation in (12), with two distinct differences:

- the strength of the inhibition term between two nodes is modulated by the inner product of the dictionary element with the corresponding dual vector for the other node whereas the HLCA uses the inner product between the dictionary elements; and
- the HLCA uses a charging circuit approach whereas the update equation in (14) has a charging term that is modulated by the binary thresholding function.

The use of the dual vectors in (14) complicates the implementation compared to the HLCA, especially in the presence of changes to the dictionary (e.g., with the death of a node in the system the entire set of dual vectors would change). While equation (14) appears to have a similar charging notion as is seen in the HLCA, the presence of the binary thresholder in this term changes the strategy for calculating the coefficients. Again, this can be seen most clearly by considering the first step each system would take starting from zero-valued coefficients. In the system of equation (14), the update term (i.e., the bracketed term) produces a small change based on the driving terms b_m. However, if the values produced at this first step are below threshold, they will only affect the next step through the inhibition term and will not continue to aggregate in the coefficient values. In other words, the charging behavior only applies to the nodes that are above threshold. This characteristic likely results in the observation by the authors that convergence is improved by efforts to explicitly decrease the sparsity of the solution during the iterations through techniques such as scaling the threshold (see Section 4 in (Kingsbury and Reeves, 2003)).

Several authors (Herrity et al., 2006; Hale et al., 2007; Figueiredo and Nowak, 2003; Daubechies et al., 2004; Blumensath and Davies, 2008) have recently introduced methods based on an Iterative Thresholding Algorithm (ITA), where the update equation for a single node is given by

$$\begin{aligned} u_m(t_{k+1}) &= b_m - \sum_{n\neq m} \langle \boldsymbol{\phi}_m, \boldsymbol{\phi}_n \rangle u_n(t_k)\, T_B(u_n(t_k)-\lambda)\,, \qquad (15) \\ a_m(t_{k+1}) &= u_m(t_{k+1})\, T_B(u_m(t_{k+1})-\lambda)\,. \end{aligned}$$

The ITA as stated (i.e., using a hard thresholder) converges to a local minimum of the ℓ^0 optimization problem. It can also be written using a soft thresholding function to converge to a global minimum of the ℓ^1 optimization problem (BPDN). The ITA is also very similar to the LCA algorithm, having the same system equation as the LCA update term. The primary difference between (15) and the LCA system equation in (12) is that the LCA uses a charging circuit approach whereas the ITA directly updates the coefficients. Again, this difference can be seen most explicitly by considering the first step each system would take starting from zero-valued coefficients. In the ITA, the first step produces coefficient values that are equal to the full driving values b_m passed through a uniform threshold. Thus, unless the threshold is initially very high, ITA also needs to prune away coefficients from the thresholded linear projections whereas the LCA inhibition terms work to keep many coefficients from ever becoming active. This charging circuit behavior also allows the LCA to be easily written in terms of simple analog computational elements such as resistors, capacitors and amplifiers.

An important distinction between the LCAs and all three of these systems is that the LCAs explicitly correspond to the sparse approximation objective function (6) employing a wide variety of coefficient cost functions. Furthermore, there is a tight and explicit connection between these cost functions and the thresholding function used by the system. The SSCN minimizes an objective function that is an approximation to (6), and it is designed specifically to use the ℓ^0 norm as the coefficient cost function. The ITA has not been derived for general cost functions, but has been derived separately for ℓ^0 and ℓ^1 optimization.

B Relating cost functions and threshold functions

To see the correspondence between a particular choice of a threshold function $T_\lambda(\cdot)$ and the sparsity-inducing cost function $C(\cdot)$, we begin by assuming we want to minimize an energy function of the form:

$$\begin{aligned} E &= \frac{1}{2}\left|\left|\boldsymbol{s}-\widehat{\boldsymbol{s}}\right|\right|^2 + \lambda\sum_m C(a_m) \\ &= \frac{1}{2}\left(\boldsymbol{s}^t\boldsymbol{s} - 2\boldsymbol{b}^t\boldsymbol{a} + \boldsymbol{a}^t\Phi^t\Phi\boldsymbol{a}\right) + \lambda\sum_m C(a_m)\,. \end{aligned}$$

For simplicity, we suppress the time variable in the notation. To find the changes in the active coefficients $\{a_m\}$ that will most significantly minimize the energy function, we take the derivative of the energy function with respect to the active coefficients,

$$\frac{dE}{da_m} = -b_m + \sum_n G_{m,n}a_n + \lambda\frac{dC(a_m)}{da_m} = -b_m + \sum_{n\neq m} G_{m,n}a_n + a_m + \lambda\frac{dC(a_m)}{da_m}, \tag{16}$$

where we assume the vectors are unit-norm $\left|\left|\boldsymbol{\phi}_m\right|\right|^2 = 1$. Looking back to the dynamic system in (4),

$$\dot{u}_m = \frac{1}{\tau}\left[b_m - u_m - \sum_{n\neq m} G_{m,n}a_n\right],$$

we can see that the dynamics on the internal state variables are proportional to the derivative of the energy function in (16), $\dot{u}_m \propto -\frac{dE}{da_m}$, if the active coefficients are related to the internal state variables by

$$u_m = a_m + \lambda\frac{dC(a_m)}{da_m}.$$

C Cost functions corresponding to ideal thresholding functions

The sigmoidal threshold function specified in (8) is invertible, meaning that active coefficients can be related back to their underlying state variables, $u_m = T^{-1}_{(\alpha,\gamma,\lambda)}(a_m)$, though not in closed form. For notational simplicity and without losing generality, we will assume in this section positive coefficients ($a_m > 0$). Though the ideal thresholding functions are not technically invertible, we can find the limit of the inverse function:

$$T^{-1}_{(\alpha,\infty,\lambda)}(a_m) = \lim_{\gamma\to\infty} T^{-1}_{(\alpha,\gamma,\lambda)}(a_m) = \begin{cases} \lambda & \text{if} \quad a_m < (1-\alpha)\lambda \\ \lambda + a_m - (1-\alpha)\lambda & \text{if} \quad a_m \geq (1-\alpha)\lambda. \end{cases}$$

Using the correspondence from Appendix B,

$$\lambda\frac{dC(a_m)}{da_m} = u_m - a_m = T^{-1}_{(\alpha,\gamma,\lambda)}(a_m) - a_m,$$

we integrate to find the ideal cost function

$$\begin{aligned} C(a_m) &= \frac{1}{\lambda}\int_0^{a_m}\left(T^{-1}_{(\alpha,\infty,\lambda)}(x) - x\right)dx \\ &= \frac{1}{\lambda}\left(\int_0^{a_m}(\lambda - x)\,dx + \int_{(1-\alpha)\lambda}^{a_m}(x-(1-\alpha)\lambda)\,dx\right) \\ &= \alpha a_m + \frac{\lambda\,(1-\alpha)^2}{2}. \end{aligned}$$

D Stability of LCAs

D.1 Equilibrium points

For a given set of active and inactive coefficients the LCA system equations are linear and only change when a node crosses threshold (from above or below). A sub-field of control theory specifically addresses these *switched systems* (Decarlo et al., 2000). To express the HLCA system as a switched system, we define $\mathcal{M} \subseteq [1, \ldots, M]$ as the current set of active nodes (i.e., $m \in \mathcal{M}$ if $|u_m(t)| \geq \lambda$). We also define a $(M \times M)$ selection matrix $S_{\mathcal{M}}$ as being all zeros except for ones on the diagonal corresponding to the active nodes,

$$[S_{\mathcal{M}}]_{m,n} = \begin{cases} 1 & \text{if } m = n \text{ and } m \in \mathcal{M} \\ 0 & \text{if } m \neq n \text{ or } m \notin \mathcal{M}. \end{cases}$$

Defining the system matrix $A_{\mathcal{M}} = \frac{1}{\tau}\left[(I - \Phi^t\Phi)\, S_{\mathcal{M}} - I\right]$, the HLCA is written as a *switched linear system*,[9]

$$\dot{\boldsymbol{u}}(t) = \frac{1}{\tau}\Phi^t \boldsymbol{s}(t) + A_{\mathcal{M}}\boldsymbol{u}(t)\,.$$

There are only finitely many possible sets $\mathcal{M}$ which we further limit by only allowing sets satisfying the stability criterion (active nodes must not form linearly dependent subdictionaries). We also assume that a given fixed input $\boldsymbol{s}$ induces equilibrium points $\boldsymbol{u}^*$ that do not have any components identically equal to threshold $u^*_m \neq \lambda$. This condition appears true with overwhelming probability, and implies that there exists $r > 0$ such that $|u^*_m| - r \geq \lambda$ for all $m \in \mathcal{M}$ and $|u^*_m| + r \leq \lambda$ for all $m \notin \mathcal{M}$.

For a given $\mathcal{M}$, linear systems theory indicates that the system

$$\dot{\boldsymbol{u}} = A_{\mathcal{M}}\boldsymbol{u}$$

has a single equilibrium point (i.e., is *asymptotically stable*) only if $A_{\mathcal{M}}$ has negative eigenvalues (Franklin et al., 1986). The matrix $A_{\mathcal{M}}$ has no positive eigenvalues, so we must show that it is full rank ($A_{\mathcal{M}}\boldsymbol{u} \neq 0, \forall \boldsymbol{u} \in \mathbb{R}^M$). We begin by determining the nullspace $\mathcal{N}(\cdot)$ of the composite matrix $\Phi^t\Phi S_{\mathcal{M}}$. The nullspace of Φ^t is empty, $\mathcal{N}(\Phi^t) = \emptyset$, because $\text{span}\{\boldsymbol{\phi}_m\} = \mathbb{R}^N$. Because the collection $\{\boldsymbol{\phi}_m\}_{m\in\mathcal{M}}$ is linearly independent and the matrix $\Phi S_{\mathcal{M}}$ consists of only those selected vectors on the columns, $\mathcal{N}(\Phi S_{\mathcal{M}}) = \text{span}\{\boldsymbol{e}_m\}_{m\notin\mathcal{M}}$, where $\boldsymbol{e}_m$ are the canonical basis elements. Therefore the composite matrix also has a nullspace of $\mathcal{N}(\Phi^t\Phi S_{\mathcal{M}}) = \text{span}\{\boldsymbol{e}_m\}_{m\notin\mathcal{M}}$. Without losing generality, we assume that the first $|\mathcal{M}|$ entries are active, $\mathcal{M} = 1, \ldots, |\mathcal{M}|$. Consider first the case when all non-trivial internal state vectors only have non-zero values in the first $|\mathcal{M}|$ positions, $\boldsymbol{u} \in \text{span}\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{|\mathcal{M}|}\}$. In this case, $\boldsymbol{u} \notin \mathcal{N}(\Phi^t\Phi S_{\mathcal{M}})$, implying that $A_{\mathcal{M}}\boldsymbol{u} = -\Phi^t\Phi S\boldsymbol{u} \neq 0$. Consider next the case when all non-trivial internal state vectors only have non-zero values in the last $(M - |\mathcal{M}|)$ positions, $\boldsymbol{u} \in \text{span}\{\boldsymbol{e}_{|\mathcal{M}|+1}, \ldots, \boldsymbol{e}_M\}$. In this case, $\boldsymbol{u} \in \mathcal{N}(\Phi^t\Phi S_{\mathcal{M}})$, meaning that $A_{\mathcal{M}}\boldsymbol{u} = -\boldsymbol{u} \neq 0$. Taking these two cases together, we see that $A_{\mathcal{M}}\boldsymbol{u} \neq 0, \forall \boldsymbol{u} \in \mathbb{R}^M$, implying that $A_{\mathcal{M}}$ only has negative eigenvalues so that the system in question has a single equilibrium point.

[9] All LCA systems can be written as a similar switched system, but the thresholding functions with additive correction terms require a cumbersome definition of proxy state variables that we omit here.

Given a particular set of active nodes $\mathcal{M}$, we therefore have a single equilibrium point $\boldsymbol{u}^*$ defined by the system matrix $A_{\mathcal{M}}$. All other points within a neighborhood of this equilibrium point correspond to the same set of active nodes (and therefore the same system matrix). Therefore, since each system matrix has a single equilibrium, there can be no other equilibrium points with coordinates within a neighborhood of $\boldsymbol{u}^*$,

$$|u_m^* - u_m| < r \quad \text{for any} \quad m \implies f(\boldsymbol{u}) \neq 0.$$

We know then that there are a finite number of equilibrium points, and each equilibrium point is isolated because there can be no other equilibrium points infinitely close.

Finally we consider the stability of the system in the neighborhood of the equilibrium point $\boldsymbol{u}^*$. Because we know that the linear subsystem is asymptotically stable, we must show that there exists a $\epsilon > 0$ such that for any $\boldsymbol{u}(0) \in B_\epsilon(\boldsymbol{u}^*)$, the set of active nodes $\mathcal{M}$ never changes so $A_{\mathcal{M}}$ stays fixed. We must therefore ensure that we can specify a $\epsilon > 0$ such that for a fixed $A_{\mathcal{M}}$ the internal states never change state, $|u_m(t)\,| > \lambda, \forall m \in \mathcal{M}$ and $|u_m(t)\,| < \lambda, \forall m \notin \mathcal{M}$. The tools of linear systems theory give this evolution (Franklin et al., 1986):

$$\begin{aligned}
\boldsymbol{u}(t) &= e^{A_{\mathcal{M}}t}\boldsymbol{u}(0) + \int_0^t e^{(t-\tau)A_{\mathcal{M}}}\Phi^t \boldsymbol{s}\, d\tau \\
&= e^{A_{\mathcal{M}}t}\boldsymbol{u}(0) + e^{A_{\mathcal{M}}t}\left(\int_0^t e^{-A_{\mathcal{M}}\tau} d\tau\right)\Phi^t \boldsymbol{s} \\
&= e^{A_{\mathcal{M}}t}\boldsymbol{u}(0) + e^{A_{\mathcal{M}}t}\left(-A^{-1}e^{-A_{\mathcal{M}}t} + A^{-1}\right)\Phi^t \boldsymbol{s} \\
&= e^{A_{\mathcal{M}}t}\boldsymbol{u}(0) + -A^{-1}\Phi^t \boldsymbol{s} + e^{A_{\mathcal{M}}t}A^{-1}\Phi^t \boldsymbol{s} \\
&= e^{A_{\mathcal{M}}t}\left(\boldsymbol{u}(0) - \boldsymbol{u}^*\right) + \boldsymbol{u}^*,
\end{aligned}$$

where $\lim \boldsymbol{u}(t) = -A^{-1}\Phi^t \boldsymbol{s} = \boldsymbol{u}^*$ for a linear system. From this, we bound the energy of the difference signal

$$||\boldsymbol{u}(t) - \boldsymbol{u}^*|| = \left|\left|e^{A_{\mathcal{M}}t}\left(\boldsymbol{u}(0) - \boldsymbol{u}^*\right)\right|\right| \leq e^{\mu_{\max}t}\,||\boldsymbol{u}(0) - \boldsymbol{u}^*|| \leq ||\boldsymbol{u}(0) - \boldsymbol{u}^*|| \leq \epsilon,$$

where $\mu_{\max}$ is the largest magnitude eigenvector of $A_{\mathcal{M}}$. This energy bound also serves as a crude bound on the individual elements of the internal state vector

$$|u_m(0) - u_m^*| \leq ||\boldsymbol{u}(t) - \boldsymbol{u}^*|| \leq \epsilon.$$

We conclude that if $\epsilon < r$, the system will not change state and behaves as a fixed, asymptotically stable linear system. Therefore, the system is locally asymptotically stable around each equilibrium point, $\boldsymbol{u}(0) \in B_r(\boldsymbol{u}^*)$.

D.2 Input-output stability

Consider the time derivative of the energy computed through the chain rule,

$$\frac{d}{dt}E(t) = \frac{dE}{d\boldsymbol{a}}\frac{d\boldsymbol{a}}{d\boldsymbol{u}}\dot{\boldsymbol{u}} = -\left(\frac{dE}{d\boldsymbol{a}}\right)^2 \frac{d\boldsymbol{a}}{d\boldsymbol{u}},$$

where the last equality follows from the definition of the LCA dynamics $\dot{\boldsymbol{u}} = -\frac{dE}{d\boldsymbol{a}}$. Therefore, as long as $T_\lambda(\cdot)$ is non-decreasing, $\frac{d\boldsymbol{a}}{d\boldsymbol{u}} \geq 0$, implying that the energy function will be non-increasing with time $\frac{d}{dt}E(t) \leq 0$.

To assess input-output stability, define τ_D to be the average dwell time between changes to the set of active nodes $\mathcal{M}$. For switched linear systems, sufficient conditions for input-output stability require each subsystem to be asymptotically stable and that the system doesn't switch "too often" (Hespanha and Morse, 1999). Specifically, we restate here a theorem from switched system theory in language corresponding to the LCAs.

Average dwell time theorem (Theorem 2 combined with Lemma 1 in (Hespanha and Morse, 1999)). *Given a collection of system matrices $A_{\mathcal{M}}$ and a positive constant ω_0 such that $A_{\mathcal{M}} + \omega_0 I$ is asymptotically stable for all t, then, for any $\omega \in [0, \omega_0]$, there is a finite constant τ_D^* such that as long as $\tau_D \geq \tau_D^*$, the switched system has a bounded internal state for piecewise constant input signals $\boldsymbol{s}(t)$:*

$$\left(\int_0^t e^{2\omega\tau} \left|\left|\boldsymbol{u}(\tau)\right|\right|^2 d\tau \right)^{1/2} \leq \kappa_1 \left(\int_0^t e^{2\omega\tau} \left|\left|\boldsymbol{s}(\tau)\right|\right|^2 d\tau \right)^{1/2} + \kappa_2 \left|\left|\boldsymbol{s}(0)\right|\right|,$$

where κ_1 and κ_2 are finite constants. Similar statements can be made using ℓ^∞ norms instead of ℓ^2 norms.

The average dwell time theorem guarantees that the LCA system will remain stable as long as each subsystem is asymptotically stable and τ_D is not too small. Appendix D.1 shows that the system matrix $A_{\mathcal{M}}$ has only strictly negative eigenvalues. The modified system matrix $\widetilde{A} = A_{\mathcal{M}} + \omega_0 I$ has a minimum magnitude eigenvalue of $\widetilde{\mu}_{\min} = \mu_{\min} + \omega_0$. Clearly there exists a $\omega_0 > 0$ such that $\widetilde{\mu}_{\min} > 0$ if and only if $\mu_{\min} > 0$. In other words, there exists $\omega_0 > 0$ so that $\widetilde{A}$ is asymptotically stable (thus satisfying the average dwell time theorem) if the stability criterion is met for every subsystem of the switched system.

E Steady-state sparsity of LCA systems

The LCA at steady-state looks like a fixed linear system. If we know *a priori* the set of active nodes corresponding to the steady-state response $\mathcal{M}$ then the steady-state internal state variables are given by

$$\widetilde{\boldsymbol{u}} = \lim_{t\to\infty} \boldsymbol{u}(t) = -\frac{1}{\tau} A_{\mathcal{M}}^{-1} \Phi^t \boldsymbol{s},$$

where $A_{\mathcal{M}}$ is defined as in Appendix D. While we cannot determine the set of active nodes in the limit, we can distinguish sets of nodes that *cannot* be active. When calculating the steady-state values $\widetilde{\boldsymbol{u}}$ assuming a fixed $\mathcal{M}$, if a node not in $\mathcal{M}$ is above threshold in $\widetilde{\boldsymbol{u}}$ (or a node in $\mathcal{M}$ is below threshold), the system matrix would have changed. In this case we call $\mathcal{M}$ *inconsistent.* It is important to note a subtle point: finding an inconsistency does not indicate the correct steady-state active set, but only indicates that it cannot be $\mathcal{M}$.

Given a set of candidate active nodes $\mathcal{M}$, we assume (without losing generality) the active nodes are indexed consecutively from the beginning, $\mathcal{M} = 1, \ldots, |\mathcal{M}|$. We employ the usual canonical basis elements $\boldsymbol{e}_m \in \mathbb{R}^M$ that contain a single non-zero entry in the m^{th} position (e.g., $\boldsymbol{e}_1 = [1, 0, \ldots, 0]^t$). We will also employ what we call the *Grammian basis elements* $\boldsymbol{v}_m \in \mathbb{R}^M$ that contain the inner products of one dictionary element with all the others, $\boldsymbol{v}_m = [\langle \boldsymbol{\phi}_1, \boldsymbol{\phi}_m \rangle, \langle \boldsymbol{\phi}_2, \boldsymbol{\phi}_m \rangle, \ldots, \langle \boldsymbol{\phi}_M, \boldsymbol{\phi}_m \rangle]^t = [G_{1,m}, G_{2,m}, \ldots, G_{M,m}]^t$. The system matrix can be expressed entirely in terms of these basis elements,

$$A_{\mathcal{M}} = \frac{1}{\tau} \left[\left(I - \Phi^t \Phi \right) S_{\mathcal{M}} - I \right] = -\frac{1}{\tau} \left[\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{|\mathcal{M}|}, \boldsymbol{e}_{|\mathcal{M}|+1}, \ldots, \boldsymbol{e}_M \right],$$

where $S_{\mathcal{M}}$ is the corresponding selection matrix (defined in Appendix D.1). The inverse of this system matrix has several important properties. First, for inactive nodes $m \notin \mathcal{M}$, the corresponding canonical basis vector is an eigenvector of the inverse system matrix $A^{-1}\boldsymbol{e}_m = -\tau \boldsymbol{e}_m$. Similarly, for active nodes $m \in \mathcal{M}$, the inverse system matrix transforms the corresponding Grammian basis vector into the canonical basis vector, $A^{-1}\boldsymbol{v}_m = -\tau \boldsymbol{e}_m$. We also note that the set $\left(\{\boldsymbol{v}_m\}_{m=1}^{|\mathcal{M}|} \bigcup \{\boldsymbol{e}_m\}_{m=|\mathcal{M}|+1}^{M} \right)$ is a basis for the space $\mathbb{R}^M$.

For now, let the input signal be proportional to a single dictionary element, $\boldsymbol{s} = \alpha \boldsymbol{\phi}_n$, meaning that $\Phi^t \boldsymbol{s} = \alpha \boldsymbol{v}_n$. We will assume that the scaling coefficient is greater than the chosen threshold, $\alpha > \lambda$, so the signal strength is considered significant. There exists a unique set of coefficients $\{\beta_m\}$ such that

$$\alpha \boldsymbol{v}_n = \beta_1 \boldsymbol{v}_1 + \cdots + \beta_{|\mathcal{M}|} \boldsymbol{v}_{|\mathcal{M}|} + \beta_{|\mathcal{M}|+1} \boldsymbol{e}_{|\mathcal{M}|+1} + \cdots + \beta_M \boldsymbol{e}_M.$$

Looking at each element of this expression in turn is illuminating:

$$\begin{aligned}
\alpha\left[\boldsymbol{v}_n\right]_1 &= \langle \boldsymbol{\phi}_1, \alpha\boldsymbol{\phi}_n\rangle &&= \langle \boldsymbol{\phi}_1, \big(\beta_1\boldsymbol{\phi}_1 + \cdots + \beta_{|\mathcal{M}|}\boldsymbol{\phi}_{|\mathcal{M}|}\big)\rangle \\
& && \vdots \\
\alpha\left[\boldsymbol{v}_n\right]_{|\mathcal{M}|} &= \langle \boldsymbol{\phi}_{|\mathcal{M}|}, \alpha\boldsymbol{\phi}_n\rangle &&= \langle \boldsymbol{\phi}_{|\mathcal{M}|}, \big(\beta_1\boldsymbol{\phi}_1 + \cdots + \beta_{|\mathcal{M}|}\boldsymbol{\phi}_{|\mathcal{M}|}\big)\rangle \\
\alpha\left[\boldsymbol{v}_n\right]_{|\mathcal{M}|+1} &= \langle \boldsymbol{\phi}_{|\mathcal{M}|+1}, \alpha\boldsymbol{\phi}_n\rangle &&= \langle \boldsymbol{\phi}_{|\mathcal{M}|+1}, \big(\beta_1\boldsymbol{\phi}_1 + \cdots + \beta_{|\mathcal{M}|}\boldsymbol{\phi}_{|\mathcal{M}|} + \beta_{|\mathcal{M}|+1}\boldsymbol{\phi}_{|\mathcal{M}|+1}\big)\rangle \\
& && \vdots \\
\alpha\left[\boldsymbol{v}_n\right]_M &= \langle \boldsymbol{\phi}_M, \alpha\boldsymbol{\phi}_n\rangle &&= \langle \boldsymbol{\phi}_M, \big(\beta_1\boldsymbol{\phi}_1 + \cdots + \beta_{|\mathcal{M}|}\boldsymbol{\phi}_{|\mathcal{M}|} + \beta_M\boldsymbol{\phi}_M\big)\rangle.
\end{aligned}$$

The coefficients $\beta_1, \ldots, \beta_{|\mathcal{M}|}$ correspond to the best approximation of $\boldsymbol{s}$ in the subspace spanned by $\{\boldsymbol{\phi}_m\}_{m=1}^{|\mathcal{M}|}$.

Consider first the case when $n \in \mathcal{M}$. The coefficients are optimal: $\beta_n = 1$ and $\beta_m = 0$ for all $m \neq n$. Assuming the fixed system matrix $A_{\mathcal{M}}$, the steady-state internal state variables are given by

$$\widetilde{\boldsymbol{u}} = -\frac{1}{\tau}A_{\mathcal{M}}^{-1}\Phi^t\boldsymbol{s} = -\frac{1}{\tau}A_{\mathcal{M}}^{-1}\alpha\boldsymbol{v}_n = \alpha\boldsymbol{e}_n.$$

If the LCA selects the optimal set of nodes, the coefficient values are optimal. Now consider the case when the vector is not part of the active set, $n \notin \mathcal{M}$. The coefficients $\{\beta_m\}$ correspond to the steady-state values:

$$-\frac{1}{\tau}A_{\mathcal{M}}^{-1}\Phi^t\boldsymbol{s} = \beta_1\boldsymbol{e}_1 + \cdots + \beta_{|\mathcal{M}|}\boldsymbol{e}_{|\mathcal{M}|} + \beta_{|\mathcal{M}|+1}\boldsymbol{e}_{|\mathcal{M}|+1} + \cdots + \beta_M\boldsymbol{e}_M.$$

Looking at the entries of $\alpha\boldsymbol{v}_n$, each index not in the active set, $m \notin \mathcal{M}$, has the coefficient

$$\beta_m = \alpha\langle\boldsymbol{\phi}_m, \boldsymbol{\phi}_n\rangle - \big(\beta_1\langle\boldsymbol{\phi}_1, \boldsymbol{\phi}_{|\mathcal{M}|+1}\rangle + \cdots + \beta_{|\mathcal{M}|}\langle\boldsymbol{\phi}_{|\mathcal{M}|}, \boldsymbol{\phi}_{|\mathcal{M}|+1}\rangle\big)\,.$$

The set $\mathcal{M}$ is consistent only if $\beta_m > \lambda$ for all $m \in \mathcal{M}$, and $\beta_m > \lambda$ for all $m \notin \mathcal{M}$.

These results lead us to several observations that help qualify the sparsity of the LCA solutions:

1. When $n \in \mathcal{M}$, $\widetilde{u}_m = 0 < \lambda$ for all $m \neq n$ means that if the LCA finds the optimal node, it will not include any extraneous nodes.

2. When $n \in \mathcal{M}$, $\widetilde{u}_n = \alpha$ means that if the optimal node is correctly selected by the LCA in the steady state, the system will find the optimal coefficient for that node.

3. When $n \notin \mathcal{M}$, $\beta_m > \lambda$ for all $m \in \mathcal{M}$ and $\beta_m < \lambda$ for all $m \notin \mathcal{M}$ means that the input signal can be represented by dictionary elements $\{\boldsymbol{\phi}_m\}_{m=1}^{|\mathcal{M}|}$ so the residual projection onto any other vector is less than λ. Any set of active nodes that cannot represent the input signal to this accuracy is inconsistent.

To minimize notation, we have only discussed one-sparse input signals. However, the analysis performed here is entirely linear and the same principles apply to input signals containing more than one dictionary component. In particular, a set of active nodes is inconsistent if: it cannot represent every component of the input signal so that the residual projection onto every other dictionary element is less than λ; or it contains every component of the input signal in addition to other extraneous components. Also, if the active set recovers the correct indices, the LCA steady-state coefficients will find the optimal coefficients.

References

Bacciotti, A. and Rosier, L. (2001). *Liapunov functions and stability in control theory.* Springer, New York.

Berger, T. (1971). *Rate Distortion Theory.* Prentice Hall, Englewood Cliffs, NJ.

Blumensath, T. and Davies, M. (2008). Iterative thresholding for sparse approximations. *The Journal of Fourier Analysis and Applications*. In Press.

Candès, E. and Donoho, D. (2004). New tight frames of curvelets and optimal representations of objects with piecewise C^2 singularities. *Communications on Pure and Applied Mathematics*, 57(2):219–266.

Chen, S., Donoho, D., and Saunders, M. (2001). Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 43(1):129–159.

Christensen, O. (2002). *An Introduction to Frames and Riesz Bases*. Birkhauser, Boston, MA.

Cichocki, A. and Unbehauen, R. (1993). *Neural Networks for Optimization and Signal Processing*. Wiley.

Cohen, M. and Grossberg, S. (1983). Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(5):815–825.

Cover, T. and Thomas, J. (1991). *Elements of Information Theory*. John Wiley & Sons, Inc., New York, NY.

Daubechies, I., M. Defrise, M., and De Mol, C. (2004). An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457.

Davis, G., Mallat, S., and Zhang, Z. (1994). Adaptive time-frequency decompositions with matching pursuit. *Optical Engineering*, 33(7).

Dayan, P. and Abbott, A. (2001). *Theoretical Neuroscience*. MIT Press, Cambridge, MA.

Decarlo, R., Cranicky, M., Pettersson, S., and Lennartson, B. (2000). Perspectives and results on the stability and stabilizability of hybrid systems. *Proceedings of the IEEE*, 88(7):1069–1082.

Delgutte, B., Hammond, B., and Cariani, P. (1998). *Psychophysical and Physiological Advances in Hearing*, chapter Neural coding of the temporal envelope of speech: Relation to modulation transfer functions, pages 595–603. Whurr Publishers, Ltd.

DeVore, R. and Temlyakov, V. (1996). Some remarks on greedy algorithms. *Advances in Computational Mathematics*, 5:173–187.

Donoho, D. (1995). Denoising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627.

Donoho, D. and Elad, M. (2003). Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ^1 minimization. *Proceedings of the National Academy of Sciences of the United States of America*, 100(5):2197–2202.

Donoho, D., Tsaig, Y., Drori, I., and Starck, J. (2006). Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit.

Elad, M. (2006). Why simple shrinkage is still relevant for redundant representations? *IEEE Transactions on Information Theory*, 52(12):5559–5569.

Feichtinger, H., Türk, A., and Strohmer, T. (1994). Hierarchical parallel matching pursuit. In *Proceedings of SPIE*, volume 2302, pages 222–232, San Diego, CA.

Field, D. (1994). What is the goal of sensory coding? *Neural Computation*, 6:559–601.

Figueiredo, M. and Nowak, R. (2003). An EM algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12:906–916.

Fischer, S., Cristóbal, G., and Redondo, R. (2004). Sparse overcomplete Gabor wavelet representation based on local competitions. *IEEE Transactions on Image Processing*, 15(2):265–272.

Franklin, G., Powell, J., and Emami-Naeini, A. (1986). *Feedback Control of Dynamic Systems*. Addison-Wesley Publishing Company, Reading, MA.

Hale, E., Yin, W., and Zhang, Y. (2007). A fixed-point continuation method for l1-regularized minimization with applications to compressed sensing. Technical Report TR07-07, Department of Computational and Applied Mathematics, Rice University.

Herrity, K., Gilbert, A., and Tropp, J. (2006). Sparse approximation via iterative thresholding. In *Proceedings of the 2006 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Toulouse, France.

Hespanha, J. and Morse, A. (1999). Stability of switched systems with average dwell time. In *Proceedings of the 38th Conference on Decision and Control*.

Hopfield, J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences of the United States of America*, 81(10):3088–3092.

Khalil, H. (2002). *Nonlinear Systems*. Prentice Hall, Upper Saddle Tiver, NJ, third edition.

Kingsbury, N. and Reeves, T. (2003). Redundant representation with complex wavelets: How to achieve sparsity. In *Proceedings of the International Conference on Image Processing (ICIP)*.

Kreutz-Delgado, K., Murray, J., Rao, B., Engan, K., Lee, T., and Sejnowski, T. (2003). Dictionary learning algorithms for sparse representation. *Neural Computation*, 15:349–396.

Lewicki, M. (2002). Efficient coding of natural sounds. *Nature Neuroscience*, 5:356–363.

Li, J., Michel, A., and Porod, W. (1988). Qualitative analysis and synthesis of a class of neural networks. *IEEE Transactions on Circuits and Systems*, 35(8):976–986.

Mallat, S. and Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415.

Natarajan, B. (1995). Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234.

Norris, J. (1997). *Markov Chains*. Cambridge University Press, New York.

Olshausen, B. (2003). Principles of image representation in visual cortex. In Chalupa, L. and Werner, J., editors, *The Visual Neurosciences*, pages 1603–1615. MIT Press.

Olshausen, B. and Field, D. (1996). Emergence of simple cell receptive properties by learning a sparse code for natural images. *Nature*, 381:607–609.

Olshausen, B. and Field, D. (2004). Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, 14:481–487.

Pece, A. and Petkov, N. (2000). Fast atomic decomposition by the inhibition method. In *In Proceedings of the 15th International Conference on Pattern Recognition*.

Perrinet, L. (2005). Efficient source detection using integrate-and-fire neurons. In *Artificial Neural Networks: Biological Inspirations? ICANN 2005*, volume 3696/2005 of *Lecture Notes in Computer Science*, pages 167–172.

Rao, B. and Kreutz-Delgado, K. (1999). An affine scaling methodology for best basis selection. *IEEE Transactions on Signal Processing*, 47(1):187–200.

Rebollo-Neira, L. and Lowe, D. (2002). Optimized orthogonal matching pursuit approach. *IEEE Signal Processing Letters*, 9(4):137–140.

Rehn, M. and Sommer, T. (2007). A network that uses few active neurones to code visual input predicts the diverse shapse of cortical receptive fields. *Journal of Computational Neuroscience*, 22(2):135–146.

Simoncelli, E. and Freeman, W. (1995). The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *IEEE Second International Conference on Image Processing*.

Süli, E. and Mayers, D. (2003). *An introduction to numerical analysis*. Cambridge University Press, New York.

Tipping, M. (2001). Sparse bayesian learning and the relevance vector machine. *The Journal of Machine Learning Research*, 1:211–244.

Tropp, J. (2004). Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242.

Tropp, J. (2006). Random subdictionaries of general dictionaries. Preprint.

Vinje, W. and Gallant, J. (2002). Natural stimulation of the nonclassical receptive field increases information transmission efficiency in V1. *Journal of Neuroscience*, 22:2904–2915.

Wipf, D. and Rao, B. (2004). Sparse Bayesian learning for basis selection. *IEEE Transactions on Signal Processing*, 52(8):2153–2164.

Yang, H. and Dillon, T. (1994). Exponential stability and oscillation of Hopfield graded response neural network. *IEEE Transactions on Neural Networks*, 5(5):719–729.