Dually Optimal Neuronal Layers: Lobe Component Analysis

Juyang Weng, *Fellow, IEEE*, and Matthew Luciw, *Student Member, IEEE*

***Abstract*—Development imposes great challenges. Internal "cortical" representations must be autonomously generated from interactive experiences. The eventual quality of these developed representations is of course important. Additionally, learning must be as fast as possible—to quickly derive better representation from limited experiences. Those who achieve both of these will have competitive advantages. We present a cortex-inspired theory called lobe component analysis (LCA) guided by the aforementioned dual criteria. A lobe component represents a high concentration of probability density of the neuronal input space. We explain how lobe components can achieve a dual—spatiotemporal ("best" and "fastest")—optimality, through mathematical analysis, in which we describe how lobe components' plasticity can be temporally scheduled to take into account the *history* of observations in the best possible way. This contrasts with using only the last observation in gradient-based adaptive learning algorithms. Since they are based on two cell-centered mechanisms—Hebbian learning and lateral inhibition—lobe components develop *in-place*, meaning every networked neuron is individually responsible for the learning of its signal-processing characteristics within its connected network environment. There is no need for a separate learning network. We argue that in-place learning algorithms will be crucial for real-world large-size developmental applications due to their simplicity, low computational complexity, and generality. Our experimental results show that the learning speed of the LCA algorithm is drastically faster than other Hebbian-based updating methods and independent component analysis algorithms, thanks to its dual optimality, and it does not need to use any second- or higher order statistics. We also introduce the new principle of fast learning from stable representation.**

***Index Terms*—Blind source separation, cortical models, feature extraction, Hebbian learning, optimality, plasticity.**

I. Introduction

In autonomous mental development (AMD), there is a growing interest in simulating the developmental process of feature detectors in sensorimotor pathways [1]–[4]. But it is becoming apparent that real-world development imposes restrictions that many existing learning algorithms cannot meet. For example, in an early sensory pathway, there is a need for developing feature detectors (neurons) for all sensed areas (receptive fields) across different positions and sizes in the sensor array. But the total number of neurons is so large that it is impractical for each neuron to have much extra storage space for its development, such as space necessary for the expectation maximization [5] technique, which requires each neuron to store a $d \times d$ covariance matrix if it has d input lines (and d is large). What is needed are algorithms with low time and space complexities, simplicity, efficiency, and biological plausibility. This raises the critical need for in-place algorithms, as we explain in Section II.

This paper presents the theory of *lobe component analysis* (LCA) for developing cortical feature layers and as a fundamental theory of cortical development. It also presents the main ideas of in-place development. This concept and the LCA algorithm were introduced in [6] and used as each layer in multilayer in-place learning networks (MILN) [7], [8]. The MILN-based model of six-layer cerebral cortex [9], which was informed by the work of Felleman and Van Essen [10], Callaway *et al.* [11], [12], and others (e.g., [13]), used LCA on both its supervised (L2/3) and unsupervised (L4) functional layers. However, LCA has not been formally and theoretically introduced, in-depth, until now. This is an archival paper presenting the LCA theory, its properties, its algorithm, and the associated experimental comparisons in their entirety. Most of the analyses of the theory presented here are new, and so are many experimental comparisons.

Each feature layer in developing cortex faces two conflicting criteria.

1) *Spatial:* with its limited number of neurons, the layer tries to learn the best internal representation from the environment.
2) *Temporal:* with, e.g., a child's limited time for learning, the layer must not only learn the best representation but also learn quickly, and do so without forgetting important mental skills acquired a long time ago.

Network learning models have faced a fundamental problem arising from these two conflicting criteria: the need for long-term memory (stable representation) and the need for fast adaptation (to learn quickly from just a few input samples) while integrating both long-term and short-term memories. This issue was previously characterized by Grossberg and Carpenter [14], [15]. The LCA theory, described in this paper, is meant to optimally address this open problem. The LCA algorithm incrementally computes an optimal solution at each time step of development—required to realize fast local adaptation needed for AMD.

The theory presented here starts from a well-accepted biological network and two well-known simple neuron learning mechanisms: Hebbian learning (see, e.g., [16, p. 1262]) and lateral inhibition (see, e.g., [16, p. 4623]). We show that each neuron, operating by these simple biological mechanisms, estimates what is called a lobe component, which corresponds to

Manuscript received December 17, 2008; revised March 11, 2009. First published April 28, 2009; current version published May 29, 2009.

The authors are with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824 USA (e-mail: weng@cse.msu.edu; luciwmat@cse.msu.edu).

Digital Object Identifier 10.1109/TAMD.2009.2021698

a high concentration of probability density of the input space. A lobe component is represented by input neural fibers of a neuron (vector projection) having near-optimal statistical efficiency. Since there are many lobe components in the input space, two key mechanisms make the entire biologically inspired network successful without sticking into local extrema. First is the temporal scheduling of plasticity that is neuron specific. This is realized by Hebbian learning and a (novel to LCA) *neuron-specific* plasticity schedule that biologically would be controlled by genes and cell physiology.[1] Second is the sequential interaction with other neurons that share the same input space and have the same scheduled near-optimal plasticity. Biologically, this interaction is realized by lateral inhibition.

Hebbian learning algorithms using a single learning rate may use the correct direction of synapse change but will not always take the best "step" towards the goal (optimal representation vector) at each update. The LCA algorithm presented has *optimal* estimation efficiency. Instead of a single learning rate, it uses both a learning rate and a retention rate, by which it optimally takes into account the entire observation history to converge to the most efficient estimation of the update history in the fastest possible way. Using a single learning rate can only consider each observation in turn and cannot tune this learning rate for optimal statistical efficiency.

This paper is organized as follows. In Section II, we provide a categorization and discussion of learning algorithms. Section III theoretically introduces a series of concepts pertaining to the lobe component theory and explains LCA's dual optimality. Section IV presents the near-optimal LCA algorithm derived from the theory presented in the previous sections. Experimental examples and comparisons are presented in Section V. Section VI discusses broader implications.

II. Types of Learning Algorithms

Consider a simple computational model of a neuron (indexed i) having n synaptic inputs. Its firing rate is modeled by

$$y_i = g(v_{i,1}x_1 + v_{i,2}x_2 + \cdots + v_{i,n}x_n) = g(\mathbf{v}_i \cdot \mathbf{x}) \qquad (1)$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is the vector of firing rates of each of the n input lines and the synaptic strength (weight) associated with each input line x_j is $v_{i,j}, j = 1, 2, \ldots, n$. The function g may handle undersaturation (noise suppression) or oversaturation. Traditionally, g has been a sigmoid function. For most of the analysis and for the experiments provided in this paper, g is not necessary.[2]

There are many learning algorithms that aim to determine these weights for a set of neurons using observations (data). In order to better understand the nature of existing learning algorithms ,we categorize them into five types.

1) *Type-1 batch:* A batch learning algorithm requires a batch of vector inputs $B = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_b\}$, where b is the batch size.
   The well-known batch back-propagation algorithm for feed-forward networks, the batch k-mean clustering algorithm, the batch principal component algorithm (PCA) (e.g., [17] and [18]), the batch LDA algorithms (e.g., [19]–[21]), and the batch EM algorithm [5] are examples of Type-1 learning algorithms. The state-of-the-art batch algorithms for independent component analysis (ICA) include FastICA by Hyvarinen and Oja [22], [23], which is among the fastest Type-1 ICA algorithms in terms of speed of convergence and its high capability to handle high-dimensional data.
2) *Type-2 block-incremental:* A type-2 learning algorithm, L_2 breaks a series of input vectors into blocks of certain size b $(b > 1)$ and computes updates incrementally between blocks. Within each block, the processing by L_2 is in a batch fashion.
   The Extended Infomax algorithm by Sejnowski *et al.* [24], [25] is a well-known Type-2 ICA algorithm.
3) *Type-3 incremental:* Type-3 is the extreme case of Type-2 in the sense that block size $b = 1$.
   Most per-frame adaptation algorithms for neural networks belong to Type-3, such as the adaptive back-propagation algorithm for feed-forward network. The NPCA-RLS algorithm by Karhunen [26] for ICA is a Type-3 algorithm.
4) *Type-4 incremental and free of higher order statistics:* A Type-4 learning algorithm is a Type-3 algorithm, but it is not allowed to compute the second- or higher order statistics of the input $\mathbf{x}$.
   Fuzzy ART [27] is a Type-4 algorithm. The candid covariance-free (CCI) PCA algorithm [28] is a Type-4 algorithm for PCA.
5) *Type-5 in-place neuron learning:* A Type-5 learning algorithm L_5 is a Type-4 algorithm, but further, the learner L_5 must be implemented by the signal-processing neuron. By in-place development, we mean that an (artificial) neuron has to learn on its own while interacting with nearby neurons to develop into a feature detector. In other words, in an in-place learning network, each signal-processing neuron itself is embedded with its own adaptation mechanism, and therefore, there is no need for an extra network to handle its adaptation.
   The CCI LCA algorithm presented in this paper is an in-place learning algorithm provided that the cell-specific and experience-specific plasticity can be scheduled by each neuron itself [3].

[1]The LCA algorithm therefore predicts that each neuron has a scheduled plasticity profile, whose plasticity at any time is determined by the cell's firing "age." This does not mean a cell has to "keep track" of a point on a temporal continuum. Firing age is merely an implicit property of each cell.

[2]We provide a discussion in the Appendix about how including g can change the lobe component to a robust version.

[3]To better understand the biological motivation for "plasticity scheduling": cell regulated time-variant plasticity is roughly described by the term "critical window," which means an early developmental time window during which the cortical areas are sufficiently plastic to quickly change according to inputs [29], [30]. While a cell ages, its plasticity decreases, yet many mature cells still exhibit some degree of plasticity [31].

What is the biological motivation for this in-place learning principle? It is known that every single cell in the human body (as long as it has a nucleus) contains the complete genetic information—the entire developmental program—sufficient to develop from the single cell into an adult. This is called the principle of genomic equivalence [32]. This impressive biological property has been dramatically demonstrated by cloning. As no genome is dedicated to more than one cell, the animal developmental program (i.e., genes program) is cell centered. In particular, each neuron (a single cell) must learn on its own through interactions with its environment. Any multicell mechanism is an emergent property of cell-centered development regulated by the genes. Each cell does not need a dedicated extracellular learner. We called this property the in-place learning property [6], [9]—every signal-processing cell in place is fully responsible for development in general and learning in particular.

The five types of algorithms have progressively more restrictive conditions, with batch (Type-1) being the most general and in-place (Type-5) being the most restrictive. As most data are stored in precompiled datasets, many algorithms have had the luxury to operate as Type-1. But to be useful for AMD, an algorithm must be able to deal with real-world data in real time. No sensory data can be explicitly stored for development. Thus, it is desirable that a developmental system uses an in-place developmental program due to its simplicity and biological plausibility. Further, biological in-place learning mechanisms can facilitate our understanding of biological systems.

Computationally, LCA leads to the lowest possible space and time complexities of neuronal learning due to its dual optimality. This is shown in this paper. In contrast, all Bayesian approaches (e.g., EM) require explicit estimation of second- and/or higher order statistics, which are stored extracellularly. They require a complex extracellular learning algorithm (not in-place) and extracellular storage (i.e., square the number of synapses for a covariance matrix), and do not learn using optimal update step lengths.

III. Theory and Concepts

Conceptually, the fate and function of a neuron is not determined by a "hand-designed" meaning from the external environment. This is another consequence of genomic equivalence. The genome in each cell regulates the cell's mitosis, differentiation, migration, branching, and connections but does not regulate the meaning of what the cell does when it receives signals from other connected cells. For example, we can find a V1 cell (neuron) that responds to an edge of a particular orientation. This is just a facet of many emergent properties of the cell that are consequences of the cell's own biological properties and the activities of its environment. As we will see next, our theory does not assume that a neuron detects a prespecified feature type (such as an edge or motion).

A neuronal layer is shown in Fig. 1. Suppose a sequentially arriving series of vectors $\mathbf{x}(1), \mathbf{x}(2), \ldots$, where each input vector $\mathbf{x}(t)$, is drawn from a high-dimensional random space $\mathcal{X}$. Assuming a layer update takes a unit time, its response from $\mathbf{x}(t)$ is $\mathbf{y}(t+1)$. The state of a neuronal layer, which includes

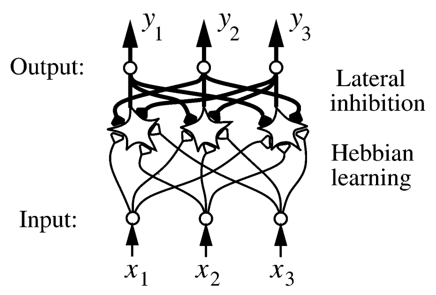


Fig. 1. A neuronal layer has $\mathbf{x}(t)$ as input at time t and generates response $\mathbf{y}(t+1)$. White triangles represent excitatory synapses and black triangles represent inhibitory synapses.

the values of the synaptic weights and the neuron ages, is denoted by $L(t)$. Denoted by $(\mathbf{y}(t), L(t)) = F(\mathbf{x}(t-1), L(t-1))$, F represents the lobe component analysis discussed in this work.

A. *Local Approximation of High-Dimensional Density*

A central issue of cortical representation by F is to estimate the probability density of $\mathbf{x}$. Given a finite resource (e.g., number of neurons), F must generate a representation that characterizes the probability distribution of high-dimensional input $\mathbf{x}$ efficiently using a limited representational resource.

High-dimensional density estimation is an important and yet very challenging problem that has been extensively studied in mathematical statistics, computer science, and engineering (see, e.g., Silverman [33] for a survey). These traditional methods are problematic when they are applied to real-world high-dimensional data. The problems include the following.

1) The lack of a method for high-dimensional density estimation that satisfies these three stringent operative requirements: incremental, covariance-free and undersample. By undersample, we mean that the incremental algorithm must work even when the number of samples is smaller than the dimension d.
2) The lack of an effective method to determine the model parameters (e.g., the means, covariances, and weights in the well-known mixture-of-Gaussian models).
3) The lack of a method that gives a *correct convergence* (a good approximation for high-dimensional data), not just convergence to a local extremum (as with the EM method for mixture-of-Gaussians).
4) The lack of a method that is optimal not only in terms of the objective function defined but also in terms of the *convergence speed* in the sense of statistical efficiency.

LCA utilizes a local approach to estimating the density of $\mathcal{X}$. A local representation of $\mathbf{x} \in \mathcal{X}$ only represents some properties of a local region in $\mathcal{X}$. Why local? A major advantage of a local method is to decompose a complex global problem of approximation and representation into multiple, simpler, local ones so that lower order statistics (means) are sufficient. This is critical for Type-4 and Type-5 algorithms, since even the second-order statistics are not plausible for a biologically inspired network. For example, ICA is a global method.

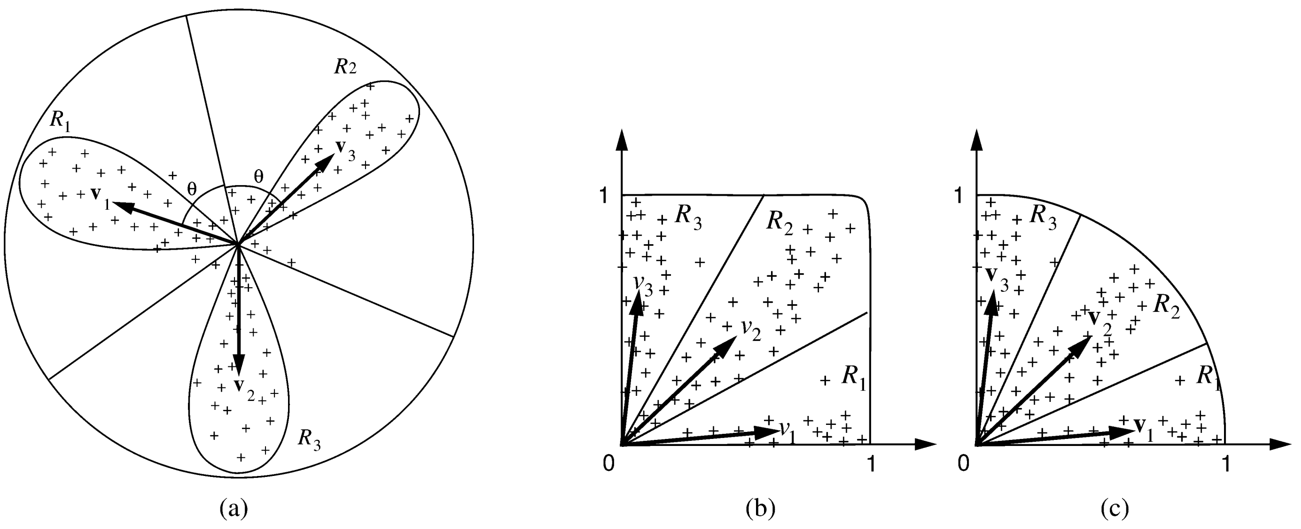


Fig. 2. Lobe components and three examples of different normalizations of input lines: whitened, nonnegative line-wise normalized, and nonnegative using a cross-neuron sigmoidal. (a) The sample space of a zero-mean whitened random vector $\mathbf{x}$ in two-dimensional (2-D) space can be illustrated by a circle. Each mark $+$ indicates a random sample of x. The distribution is partitioned into $c = 3$ lobe regions $R_i, i = 1, 2, 3$, where R_i is represented by the lobe component (vector) $\mathbf{v}_i$. (b) The sample space of nonnegative line-wise normalized random vector $\mathbf{x}$ in 2-D space. Each mark $+$ indicates a random sample of $\mathbf{x}$. The distribution is partitioned into $c = 3$ (nonsymmetric) lobe regions $R_i, i = 1, 2, 3$, where R_i is represented by the lobe component (vector) $\mathbf{v}_i$. (c) The region of R_2 is normalized by the deviation of projections along $\mathbf{v}_i$ using a cross-neuron sigmoidal $g(x)$ (see the Appendix).

B. Lobe Components

Given its limited resource of c neurons, LCA will divide the sample space $\mathcal{X}$ into c mutually nonoverlapping regions, which we call *lobe regions*

$$\mathcal{X} = R_1 \cup R_2 \cup \cdots \cup R_c \tag{2}$$

where $R_i \cap R_j = \phi$, if $i \neq j$, as illustrated in Fig. 2(a). Each region is represented by a single unit feature vector $\mathbf{v}_i$, called the *lobe component*. We model these lobe components as column vectors $\mathbf{v}_i, i = 1, 2, \ldots, c$. These lobe components are not necessarily orthogonal and not necessarily linearly independent. They span a lobe feature subspace S

$$S = \text{span}\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c\}. \tag{3}$$

Typically, the dimension of the subspace S can be smaller or larger than the input space $\mathcal{X}$, depending on the available resources.

If the distribution of $\mathbf{x}$ is Gaussian with a unit covariance matrix, the samples will be equally dense in all directions. In general, the distribution is not Gaussian and the probability density may concentrate along certain directions (although the global covariance of projections along any given direction is unit). Each major cluster along a direction is called a lobe, illustrated in Fig. 2(a) as a petal "lobe." Each lobe may have its own fine structure (e.g., sublobes). The shape of a lobe can be of any type, depending on the distribution, not necessarily like the petals in Fig. 2(a). In that figure, to facilitate understanding, we illustrate the lobe component concept using several different types of normalized input lines. However, none of these normalizations is essential. The sample space of a white[4] zero-mean random vector $\mathbf{x}$ in k-dimensional space can be illustrated by a k-dimensional hypersphere, as shown in Fig. 2(a). Fig. 2(b) shows the lobe components for the case of nonnegative line representation where each input vector is line-wise normalized. Fig. 2(c) shows a case where the input is nonnegative and normalized using a cross-neuron sigmoidal (see the Appendix).

[4]As in ICA, the vector $\mathbf{x}$ may also be whitened, so that the covariance matrix is a identify matrix $E[\mathbf{x}\mathbf{x}^\top] = I$.

If we assume that $\mathbf{x}$ and $-\mathbf{x}$ are equally likely, the distribution is then symmetric about the origin. In this case, we can define symmetric lobes so that $\mathbf{x}$ and $-\mathbf{x}$ belong to the same lobe. But, in general, this is not necessarily true. Given an arbitrary high-dimensional space $\mathcal{X}$, the distribution of $\mathbf{x} \in \mathcal{X}$ may not necessarily have factorizable components. In other words, there exist no directions $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$, so that their projection from $\mathbf{x} \in \mathcal{X} : u_i = \mathbf{x}^\top \mathbf{v}_i, i = 1, 2, \ldots, d$, is statistically independent so that their probability density function (pdf) is factorizable

$$p(\mathbf{x}) = p_1(x_1)p_2(x_2)\cdots p_d(x_d).$$

In many high-dimensional applications (e.g., using natural images), the pdf is typically not factorizable.

Once the lobe components are estimated, the discrete probability in the input space $\mathcal{X}$ can be estimated in the following way. Each region R_i keeps the number of hits $n(i)$, which records the number of times the samples of $\mathbf{x}$ fall into region R_i. Then, the continuous distribution of $\mathbf{x}$ can be estimated by a discrete probability distribution of c regions

$$P\{\mathbf{x} \in R_i\} = \frac{n(i)}{n}, \quad i = 1, 2, \ldots, c$$

where n is the total number of samples. As we can see, the larger the number c, the more regions can be derived and, thus, the finer approximation of the probability density function of $\mathcal{X}$.

C. Optimal Spatial Representation: Lobe Components

The next issue is how to mathematically compute the lobe components from observations. This defines the partition of the input space $\mathcal{X}$ by a set of regions $R_1, R_2, \ldots, R_c$. In other words, the regions are never explicitly computed. To facilitate understanding, we will first assume that the regions are given. We then derive the best representation. Then, we allow the regions to update (change).

Given any input vector $\mathbf{x}$ that belongs to a given, fixed region R_i, we would like to approximate $\mathbf{x}$ by a fixed vector $\mathbf{v}_i$ in the form of $\alpha\mathbf{v}_i$. It is well known that the value of α that minimizes $E\|\mathbf{x}-\alpha\mathbf{v}_i\|^2$ is $\alpha = \mathbf{x}\cdot\mathbf{v}_i$. In this sense, $\mathbf{x}$ is best approximated by $\hat{\mathbf{x}} = (\mathbf{x}\cdot\mathbf{v}_i)\mathbf{v}_i$.

Suppose that the set of unit principal component vectors $\{\mathbf{v}_i\}$ is given and fixed. We define the regions $\{R_j\}$ as the set that minimizes the approximation error $\|\mathbf{x}-(\mathbf{x}\cdot\mathbf{v}_i)\mathbf{v}_i\|$ among all i

$$R_j = \left\{\mathbf{x} \,|\, j = \arg\min_{1\le i\le c} \|\mathbf{x} - (\mathbf{x}\cdot\mathbf{v})\mathbf{v}_i\|\right\}.$$

It can be proved readily that the boundary between two neighboring regions R_i and R_k represented by $\mathbf{v}_i$ and $\mathbf{v}_k$, respectively, is the hyperplane that forms equal angles from $\mathbf{v}_i$ and $\mathbf{v}_k$, as shown in Fig. 2(b). Equivalently, as the $\mathbf{v}_i$s are unit, we have

$$R_j = \left\{\mathbf{x} \,|\, j = \arg\max_{1\le i\le c} (\mathbf{x}\cdot\mathbf{v}_i)\right\}.$$

Therefore, from the above way to compute the region R_i, we can see that a vector can be considered to belong to the region R_i (represented by its unit vector $\mathbf{v}_i$) based on the inner product $\mathbf{x}\cdot\mathbf{v}_i$. Therefore, given any input vector $\mathbf{x}$ and all the neuron responses, we can determine which lobe component it should contribute to—or, equivalently, which region R_i it lies within—based on which neuron gives the maximum response $\mathbf{x}\cdot\mathbf{v}_i$.

Conversely, when we know that $\mathbf{x}$ belongs to the region R_i, represented by $\mathbf{v}_i$, we ask: what $\mathbf{v}_i$ represents R_i in the best way? We define the unit vector $\mathbf{v}_i$ as the one that minimizes the squared error of approximation for all possible $\mathbf{x} \in R_i$. The squared approximation error of $\mathbf{x}$ can be rewritten as

$$\|\mathbf{x} - (\mathbf{x}\cdot\mathbf{v}_i)\mathbf{v}_i\|^2 = \mathbf{x}^\top\mathbf{x} - (\mathbf{x}\cdot\mathbf{v}_i)^2.$$

Thus, the expected error of the above approximation over R_i is

$$\begin{aligned} &E\|\mathbf{x} - (\mathbf{x}\cdot\mathbf{v}_i)\mathbf{v}_i\|^2 \\ &= E[\mathbf{x}^\top\mathbf{x} - (\mathbf{x}\cdot\mathbf{v}_i)^2] \\ &= E[\mathbf{x}^\top\mathbf{x} - \mathbf{v}_i^\top\mathbf{x}\mathbf{x}^\top\mathbf{v}_i] \\ &= E[\mathbf{x}^\top\mathbf{x}] - \mathbf{v}_i^\top E[\mathbf{x}\mathbf{x}^\top]\mathbf{v}_i \\ &= \mathrm{trace}\,(\Sigma_x) - \mathbf{v}_i^\top\Sigma_x\mathbf{v}_i \end{aligned} \tag{4}$$

where Σ_x is the correlation matrix of $\mathbf{x} \in R_i$ $\Sigma_x = E[\mathbf{x}\mathbf{x}^\top]$ conditioned on $\mathbf{x} \in R_i$.

Since trace(Σ_x) is constant, the unit $\mathbf{v}_i$ that minimizes the above expression in (4) is the one that maximizes $\mathbf{v}_i^\top\Sigma_x\mathbf{v}_i$. From the standard theory of PCA (e.g., see [34]), we know that the solution $\mathbf{v}_i$ is the unit eigenvector of conditional Σ_x associated with the largest eigenvalue $\lambda_{i,1}$

$$\lambda_{i,1}\mathbf{v}_i = \Sigma_x\mathbf{v}_i. \tag{5}$$

In other words, $\mathbf{v}_i$ is the first principal component of Σ_x, where expectation of Σ_x is over R_i. PCA theory tells us that the eigenvalue $\lambda_{i,1}$ is the averaged "power" of projection onto the unit $\mathbf{v}_i$, i.e., $\lambda_{i,1} = E(\mathbf{x}\cdot\mathbf{v}_i)^2$, conditioned on $\mathbf{x} \in R_i$.

In the above analysis, the region R_i is given. The vector $\mathbf{v}_i$ that minimizes the approximation error is the conditional principal component, conditioned on $\mathbf{x} \in R_i$. This proves that the lobe components in Fig. 2 are *spatially* optimal in the following sense. Given all c regions, we consider that each input vector $\mathbf{x}$ is represented by the winner feature $\mathbf{v}_j$, which has the highest response y_j

$$j = \arg\max_{1\le i\le c} y_i$$

where y_i is the projection of input $\mathbf{x}$ onto the normalized feature vector $\mathbf{v}_i$: $y_i = \mathbf{x}\cdot(\mathbf{v}_i/\|\mathbf{v}_i\|)$. The form of approximation of $\mathbf{x}$ is represented by $\hat{\mathbf{x}} = y_i\mathbf{v}_i/\|\mathbf{v}_i\|$. The error of this representation for $\mathbf{x}, e(\mathbf{x}) = \|\hat{\mathbf{x}} - \mathbf{x}\|$, is minimized by the lobe components, which are the principal components of their respective regions.

In summary, the spatial optimality requires that the spatial resource distribution in the cortical level is optimal in minimizing the representational error. For this optimality, the cortical-level developmental program modeled by CCI LCA computes the best feature vectors $V = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c)$ so that the expected square approximation error $\|\hat{\mathbf{x}}(V) - \mathbf{x}\|^2$ is statistically minimized

$$\begin{aligned} V^* &= (\mathbf{v}_1^*, \mathbf{v}_2^*, \ldots, \mathbf{v}_c^*) \\ &= \arg\min_V E\|\hat{\mathbf{x}}(V) - \mathbf{x}\|^2. \end{aligned} \tag{6}$$

This spatial optimality leads to Hebbian learning of optimal directions. We next address the issue of determining the best step size along the learning trajectory.

D. Temporal Optimality: Automatic Step Sizes

Intuitively speaking, the spatial optimality we have discussed until now means that with the same cortical size, all human children will *eventually* perform at the best level allowed by the cortical size. However, to reach the same skill level, one child may require more teaching than another. Spatiotemporal optimality is deeper. It requires the best performance for every time t. That is, the child learns the quickest allowed by the cortical size at every stage of his age.

To deal with both criteria of long-term memory and fast adaptation, we require an incremental and optimal solution. Motivated by biological synaptic learning, let $\mathbf{u}(t)$ be the neuronal internal observation (NIO), which for LCA is defined as *response-weighted input* $\mathbf{u}(t)$

$$\mathbf{u}(t) \overset{\mathrm{def}}{=} \frac{\mathbf{x}(t)\cdot\mathbf{v}(t-1)}{\|\mathbf{v}(t-1)\|}\mathbf{x}(t). \tag{7}$$

The synaptic weight vector $\mathbf{v}(t)$ is estimated from a series of observations $U(t) = \{\mathbf{u}(1), \mathbf{u}(2), \ldots, \mathbf{u}(t)\}$ drawn from a probability density $p(\mathbf{u})$. Let $S(t)$ be the set of all possible estimators for the parameter vector $\mathbf{v}$ (synaptic weight vector) from the set of observations $U(t)$. Suppose the learning rate η_t is for NIO $\mathbf{u}(t)$ at time t. How can we automatically determine all the learning rates $\eta_1, \eta_2, \ldots, \eta_t, \ldots$ so that the estimated neuronal weight vector $\hat{\mathbf{v}}(t)$ at every time t has the minimum error while the search proceeds along its nonlinear trajectory toward its intended target weight vector $\mathbf{v}^*$? Mathematically, this means that every update at time t reaches

$$\text{minimum-error}(t) = \min_{\hat{\mathbf{v}}(U(t)) \in S(t)} \|\hat{\mathbf{v}}(U(t)) - \mathbf{v}^*\|^2 \quad (8)$$

for all $t = 1, 2, 3, 4, \ldots$.

This corresponds to a compounding of a series of challenging problems.

a) Unknown nonlinear relationship between inputs $\mathbf{x}$ and the neuronal synaptic weight vector $\mathbf{v}$.
b) The global trajectory minimum error problem in (8).
c) The incremental estimation problem: the neuronal input $\mathbf{x}(t)$ must be used by each neuron to update its synapse $\mathbf{v}(t-1)$ and then $\mathbf{x}(t)$ must be discarded right after that.
d) No second- or higher order statistics of input vector $\mathbf{x}$ can be estimated by each neuron due to the in-place learning principle. Otherwise, each neuron with n input synapses (e.g., n is on the order of 10^3 on average in the brain [16, p. 19] [35]) would require a very large dedicated extracellular storage space (e.g., the covariance matrix requires on the order of $n^2 = 1\,000\,000$ extracellular storage units).

Standard techniques for a nonlinear optimization include gradient-based methods or higher order (e.g., quadratic) methods, but none of them is appropriate for b) and d). The biologically inspired theory of CCI LCA aims at such a closed-form solution with a)–d) under consideration.

From (5), replacing the conditional correlation matrix by the sample conditional correlation matrix, we have our estimation expression

$$\begin{aligned}\lambda_{i,1}\mathbf{v}_i &= \frac{1}{n}\sum_{t=1}^{n}\mathbf{x}(t)\mathbf{x}(t)^\top\mathbf{v}_i \\ &= \frac{1}{n}\sum_{t=1}^{n}(\mathbf{x}(t)\cdot\mathbf{v}_i)\mathbf{x}(t) \qquad (9)\end{aligned}$$

where $\mathbf{v}_i$ is a unit vector.

From this point on, we would like to define a *candid* version of the lobe component by assigning the length of the lobe component to be $\lambda_{i,1}$. That is

$$\mathbf{v}_i = \lambda_{i,1}\frac{\mathbf{v}_i}{\|\mathbf{v}_i\|}.$$

Then, the expression in (9) becomes

$$\mathbf{v}_i = \frac{1}{n}\sum_{t=1}^{n}\frac{\mathbf{x}(t)\cdot\mathbf{v}_i}{\|\mathbf{v}_i\|}\mathbf{x}(t) \qquad (10)$$

where $\mathbf{v}_i$ has a length $\lambda_{i,1}$. Equation (10) states that the candid version of $\mathbf{v}_i$ is equal to the average on the right side. By *candid,* we mean that we keep the power (energy) of the projections onto $\mathbf{v}_i$ along with $\mathbf{v}_i$ and, thus, the estimator for $\lambda_{i,1}$ is computed as the length of $\mathbf{v}_i$. The length of the vector gives the estimated eigenvalue of the principal component. It is updated along with its direction, thus keeping the original information. A scheme in which a vector is set to unit length after each update is therefore not candid. This scheme is needed for the optimal efficiency to be discussed in Section III-H.

We can see that the best candid lobe component vector $\mathbf{v}_i$, whose length is the "power estimate" $\lambda_{i,1}$, can be estimated by the *average* of the input vector $\mathbf{x}(t)$ weighted by the linearized (without sigmoidal g) response $\mathbf{x}(t)\cdot\mathbf{v}_i$ whenever $\mathbf{x}(t)$ belongs to R_i. This average expression is very important in guiding the adaptation of $\mathbf{v}_i$ in the optimal statistical efficiency, as explained in Section III-E.

The above result states that if the regions $\{R_i\}$ are given, the optimal lobe components can be determined based on (10), but the regions $\{R_i\}$ are not given. Therefore, our modeled cortex must dynamically update $\{R_i\}$ based on the currently estimated lobe components $\{\mathbf{v}_i\}$.

We define the belongingness of any vector $\mathbf{x}$ to region R_i represented by lobe component $\mathbf{v}_i$ as follows.

Definition 1: Belongingness of $\mathbf{x}$ to R_i is defined as the response $y_i = \mathbf{x}\cdot\mathbf{v}_i\|\mathbf{v}_i\|^{-1}$, where $\mathbf{v}_i$ is the candid lobe component vector representing region R_i.

Given a series of regions R_i, each being represented by lobe component $\mathbf{v}_i, i = 1, 2, \ldots, c$, an input $\mathbf{x}$ belongs to R_j if

$$j = \arg\max_{1\le i\le c}\{y_i\}.$$

Thus, LCA must compute the directions of the lobe components and their corresponding energies sequentially and incrementally. For in-place development, each neuron does not have extra space to store all the training samples $\{\mathbf{x}(t)\}$. Instead, it uses its physiological mechanisms to update synapses incrementally.

Equation (10) leads to an important incremental estimation procedure. If the ith neuron $\mathbf{v}_i(t-1)$ at time $t-1$ has already been computed using previous $t-1$ inputs $\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(t-1)$, the new input $\mathbf{x}(t)$ enables a new NIO defined as *response-weighted input:* $\mathbf{u}(t)$ that we defined in (7).

Then, the *candid* version of $\mathbf{v}_i$ is equal to the average

$$\mathbf{v}_i(t) = \frac{1}{n}\sum_{t=1}^{n}\mathbf{u}(t). \qquad (11)$$

This mechanism not only enables us to compute the best candid $\mathbf{v}_i$ but also enables many lobe component vectors to compete when data $\mathbf{x}(t)$ are sequentially received. The vector $\mathbf{v}_i$ whose belongingness is the highest is the "winner," which best inhibits all other vectors. The winner uses the current input $\mathbf{x}(t)$ to update its vector, as in (11), but all others do not. In summary, unlike traditional views where working memory and long-term memory are two different kinds of memory, the LCA model indicates that working memory and long-term memory are dynamic in a cortical layer. At any time, the winner neurons are working memory and the other neurons are long-term memory.

Now, how can we schedule the updating to be temporally optimal? Before we solve this problem, we need to review the concept of statistical efficiency.

E. Statistical Efficiency

We will convert the nonlinear search problem of computing the optimal updating trajectory into an optimal estimation problem using the concept of statistical efficiency. Statistical efficiency is defined as follows. Suppose that there are two estimators Γ_1 and Γ_2 for vector parameter $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$ that are based on the same set of observations $S = \{\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(n)\}$. If the expected square error of Γ_1 is smaller than that of Γ_2, i.e., $E\|\Gamma_1 - \theta\|^2 < E\|\Gamma_2 - \theta\|^2$, the estimator Γ_1 is more *statistically efficient* than Γ_2.

Statistical estimation theory reveals that for many distributions (e.g., Gaussian and exponential distributions), the sample mean is the most efficient estimator of the population mean. This follows directly from [36, Th. 4.1, p. 429–430], which states that under some regularity conditions satisfied by many distributions (such as Gaussian and exponential distributions), the maximum likelihood estimator (MLE) $\hat{\boldsymbol{\theta}}$ of the parameter vector $\boldsymbol{\theta}$ is asymptotically efficient, in the sense that its asymptotic covariance matrix is the Cramér–Rao information bound (the lower bound) for all unbiased estimators via convergence in probability to a normal distribution

$$\sqrt{n}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}) \xrightarrow{p} N\{0, I(\boldsymbol{\theta})^{-1}\} \tag{12}$$

in which the Fisher information matrix $I(\boldsymbol{\theta})$ is the covariance matrix of the score vector $\{(\partial f(\mathbf{x}, \boldsymbol{\theta}))/(\partial\theta_1), \ldots, (\partial f(\mathbf{x}, \boldsymbol{\theta}))/(\partial\theta_k)\}$, and $f(\mathbf{x}, \boldsymbol{\theta})$ is the probability density of random vector $\mathbf{x}$ if the true parameter value is $\boldsymbol{\theta}$ (see, e.g., [36, p. 428]). The matrix $I(\boldsymbol{\theta})^{-1}$ is called information bound since under some regularity constraints, any unbiased estimator $\tilde{\boldsymbol{\theta}}$ of the parameter vector $\boldsymbol{\theta}$ satisfies $\mathrm{cov}(\tilde{\boldsymbol{\theta}} - \boldsymbol{\theta}) \geq I(\boldsymbol{\theta})^{-1}/n$ (see, e.g., [36, p. 428] or [37, pp. 203–204]).[5]

Since in many cases (e.g., Gaussian and exponential distributions) the MLE of the population mean is the sample mean, we estimate the mean $\boldsymbol{\theta}$ of vector $\mathbf{x}$ by the sample mean. Thus, we estimate an independent vector $\lambda_1 \mathbf{v}_i$ by the sample mean in (11), where $\mathbf{u}(t)$ is a random observation.

F. Automatic Scheduling of Optimal Step Sizes

Having expressed the above theory, now we pick up our discussion on how to schedule the step sizes for the fastest (temporally optimal) way to estimate the $\mathbf{v}_i(t), i = 1, 2, \ldots, c$ in (11). The mean in (11) is a batch method. For incremental estimation, we can use

$$\mathbf{v}(t) = \frac{t-1}{t}\mathbf{v}(t-1) + \frac{1}{t}\mathbf{u}(t). \tag{13}$$

In other words, to get the temporally optimal estimator $\bar{\mathbf{v}}(t)$, we need to select not only an automatically determined learning rate $w_1(t) = 1/t$ but also an automatically scheduled retention rate $w_2(t) = (t-1)/t$. In other words, w_1 and w_2 jointly determine the optimal scheduling of step sizes. The above (13) gives the straight incremental mean, which is temporally optimal in the sense of (8) due to statistical efficiency discussed above.

Therefore, Hebbian learning of direction $\mathbf{u}(t)$ in (11), defined in (7), turns out to be the direction of incremental update of the dually optimal lobe component developed here. However, a direction is not sufficient for the dual optimality. The automatically scheduled rate pair—the retention rate $w_1(t)$ and the learning rate $w_2(t)$—gives the optimal "step size" at any age t. This theoretical prediction is open to biological verification.

[5]For two real symmetric matrices A and B of the same size, $A \geq B$ means that $A - B$ is nonnegative definite, which implies, in particular, that the diagonal elements are all nonnegative, which gives the lower bound for the variance of every element of the vector estimator of $\boldsymbol{\theta}$.

G. Time-Variant Distribution

With the temporal optimality established in the sense of (8), we now note the above optimality is for a stationary distribution. But we do not know the distribution of $\mathbf{u}(t)$, and it is even dependent on the currently estimated $\mathbf{v}$ (i.e., the observations are from a nonstationary process). And, intuitively, the ability for a child to learn persists throughout the child's lifetime. So, we use the following CCI plasticity technique—an "amnesic" mean [28], which gradually "forgets" old "observations" (bad $\mathbf{u}(t)$ when t is small). Modify (13) to use a pair of rates: an amnesic retention rate and an amnesic learning rate

$$\mathbf{v}(t) = \frac{t-1-\mu(t)}{t}\mathbf{v}(t-1) + \frac{1+\mu(t)}{t}\mathbf{u}(t) \tag{14}$$

where $\mu(t)$ is the amnesic function depending on t. Tuning of $\mu(t)$ is scheduled by

$$\mu(t) = \begin{cases} 0, & \text{if } t \leq t_1 \\ c(t-t_1)/(t_2-t_1), & \text{if } t_1 < t \leq t_2 \\ c + (t-t_2)/r, & \text{if } t_2 < t \end{cases}. \tag{15}$$

As can be seen above and in Fig. 3(a), $\mu(t)$ has three intervals. When t is small, straight incremental average is computed, accumulating information to estimate the mean. As the time passed is small, straight mean is good enough for the early section. Then, $\mu(t)$ enters the rising section. It changes from zero to c linearly. In this section, neurons compete for the different partitions by increasing their learning rates for faster convergence. Lastly, t enters the third section—the long adaptation section—where $\mu(t)$ increases at a rate about $1/r$, meaning the second weight $(1+\mu(t))/t$ in (13) approaches a constant $1/r$ to trace a changing distribution. Fig. 3(b) shows the development of the amnesic average coefficient, where $w_1 = (t-1-\mu(t))/(t)$ and $w_2 = (1+\mu(t))/(t)$.

A point of caution is in order here. The time t is not real time. As will be clear later, t is the firing age of the neuron, as shown in (23). A biological neuron does not need to *store* this explicit firing age nor the real time. All it needs is to *update* the learning rate w_2 (which is an implicit property of the cell) nonlinearly according to its firing experience.

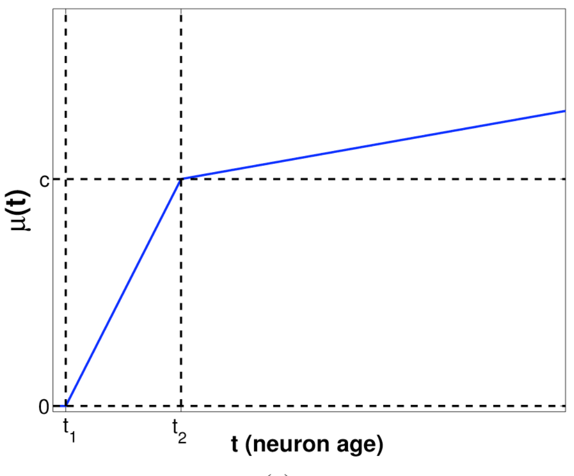
H. Efficiency of CCI Plasticity

First, we consider whether CCI plasticity-enabled mean is an unbiased estimator. From the recursive definition in (14), we can see that the amnesic mean $\bar{x}(n)$ is a weighted sum of the involved data x_t

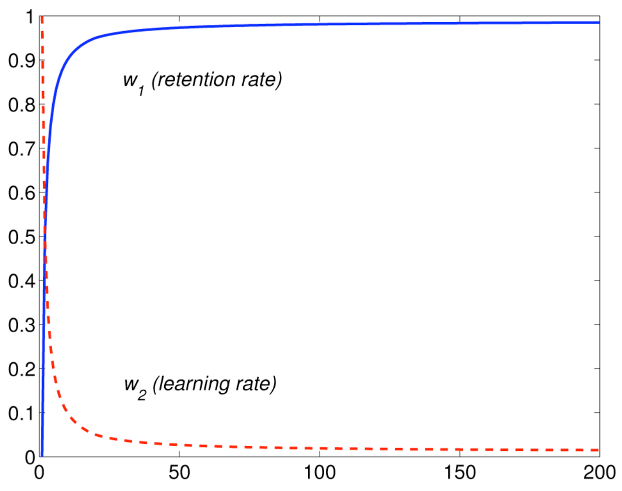
$$\bar{x}(n) = \sum_{t=1}^{n} \omega_t(n) x_t$$

where $\omega_t(n)$ is the weight of data item $x(t)$, which entered at time $t \leq n$ in $\bar{x}(n)$. It can be proven using induction on n that the weight $\omega_t(n)$ is given by the following expression:

$$\omega_t(n) = \frac{1+\mu(t)}{t} \prod_{j=t+1}^{n} \frac{j-1-\mu(j)}{j}. \tag{16}$$



(a)



(b)

Fig. 3. (a) The three-sectioned $\mu(t)$ made up of the early section, the rising section, and the long adaptation section. Each neuron has its own age-dependent plasticity schedule. (b) CCI plasticity coefficients: w_1 is retention rate and w_2 is learning rate. The x-axis is the number of updates (t) and the y-axis is w_1 and w_2. Note that the learning rate will converge to $1/r$ (not zero) and the retention rate will converge to $1-1/r$.

Since all the multiplicative factors above are nonnegative, we have $\omega_t(n) \geq 0, t = 1, 2, \ldots, n$. Using the induction on n, it can be proven that all the weights $\omega_t(n)$ sum to one for any $n \geq 1$

$$\sum_{t=1}^{n} \omega_t(n) = 1. \tag{17}$$

(When $n = 1$, we require that $\mu(1) = 0$.) Suppose that the samples x_t are independently and identically distributed (i.i.d.) with the same distribution as a random variable x. Then, CCI plasticity-enabled mean is an unbiased estimator of Ex

$$E\bar{x}(n) = E\sum_{t=1}^{n} \omega_t(n)x_t = \sum_{t=1}^{n} \omega_t(n)Ex_t$$
$$= \sum_{t=1}^{n} \omega_t(n)Ex = Ex.$$

Let $\text{cov}(x)$ denote the covariance matrix of x. The expected mean square error of the amnesic mean $\bar{x}(n)$ is

$$\text{cov}(\bar{x}(n)) = \left(\sum_{t=1}^{n} \omega_t^2(n)\right)\text{cov}(x)$$
$$= \varepsilon(n)\text{cov}(x) \tag{18}$$

where we defined the *error coefficient*

$$\varepsilon(n) = \sum_{t=1}^{n} \omega_t^2(n).$$

When $\mu(t) = 0$ for all n, the error coefficient becomes $\varepsilon(n) = 1/n$ and (18) returns to the expected square error of the regular sample mean

$$\text{cov}(\bar{x}(n)) = \frac{1}{n}\text{cov}(x). \tag{19}$$

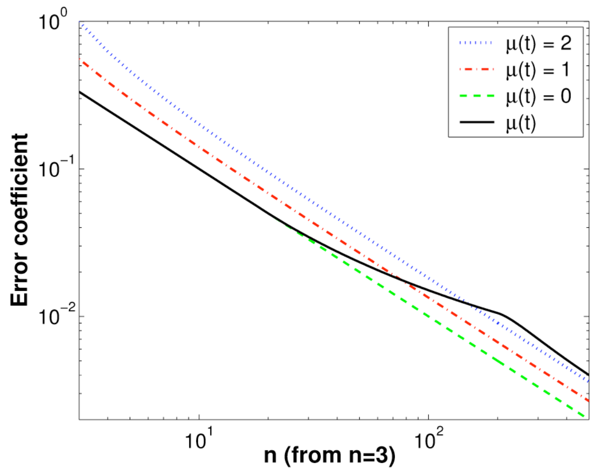


Fig. 4. The error coefficients $\varepsilon(n)$ for amnesic means with different amnesic functions $\mu(t)$. We also show when $\mu(t)$ varies with t, as in (15), using parameters $t_1 = 20, t_2 = 200, c = 2, r = 2000$. Note the logarithmic axes. A lower error coefficient is better, but when the distribution of the input changes with a large number of observations, adaptation is necessary for development.

It is then expected that the amnesic mean for a stationary process will not have the same efficiency as the straight sample mean for a stationary process. Fig. 4 shows the error coefficient $\varepsilon(n)$ for three different amnesic functions $\mu(t) \equiv 2, \mu(t) \equiv 1$ and $\mu(t) \equiv 0$. The smaller the error coefficient, the smaller the expected square error but also the less capability to adapt to a changing distribution. The three cases shown in Fig. 4 indicate that when $n > 10$, the amnesic mean with $\mu(t) \equiv 1$ increased about 50% (for the same n) from that for $\mu(t) \equiv 0$, and with $\mu(t) \equiv 2$ it increased about 100%.

From Fig. 4, we can see that a constant positive $\mu(t)$ is not best when t is small. The multisectional function $\mu(t)$ in (15) performs straight average for small t to reduce the error coefficient for earlier estimates. When t is very large, the amnesic function changes with t to track the slowly changing distribution.

The multisectional amnesic function $\mu(t)$ is more suited for practical signals with unknown nonstationary statistics (typical for development). It is appropriate to note that the exact optimality of the multisectional amnesic function is unlikely under an unknown nonstationary process (not i.i.d.) unless an assumption of certain types of nonstationary process is imposed, which is not, however, necessarily true in the reality of real-world development.

In summary, we should not expect an estimator suited for an unknown nonstationary process to have the same expected efficiency as for an i.i.d. stationary process. The distribution of signals received in many applications is typically nonstationary and, therefore, an amnesic mean with a multisectional (dynamic) amnesic function is better [see Fig. 7(b)].

The above can guide us in order to set the parameters of (15). It is unwise to introduce forgetting early, so as to contain initial $\varepsilon(n)$. This will maximize stability when few samples are available (e.g., let $t_1 = 20$). Note that, for larger c, the weight of new samples is increased and old samples are forgotten gradually. Typically, c can range from two to four. This parameter is useful in the initial organization phase, where the lobe regions are changing due to competition, which will tend to create a nonstationary distribution for each neuron. We can set t_2 to, e.g., 200, when we would expect the lobe components to be relatively well organized. For the long-term plasticity stage, r should not be too low, or too much forgetting will occur. It could range from 5000 to 15 000.

IV. Lobe Component Analysis Algorithm

A. CCI LCA Algorithm

The CCI LCA algorithm incrementally updates c neurons represented by the column vectors $\mathbf{v}_1(t), \mathbf{v}_2(t), \ldots, \mathbf{v}_c(t)$ from samples $\mathbf{x}(1), \mathbf{x}(2), \ldots$. It is desirable but not required that a neuron's input is linewise normalized so that every component in has a unit variance, but it does not need to be whitened. The length of $\mathbf{v}_i$ will be the variance of projections of the vectors $\mathbf{x}(t)$ in the ith region onto $\mathbf{v}_i$.

"Prenatal" initialization—Sequentially initialize c cells using first c inputs $\mathbf{v}_i(0) = \mathbf{x}(t)$ and set cell-update age $n_i = 1$ for $i = 1, 2, \ldots, c$.
"Live." For $t = c+1, c+2, \ldots$, do:
1. Neurons compute. Compute output (response) for all neurons: For all i with $1 \le i \le c$, compute the response[6]

$$y_i = \frac{\mathbf{x}(t) \cdot \mathbf{v}_i(t-1)}{\|\mathbf{v}_i(t-1)\|}. \tag{20}$$

2. Lateral inhibition for different features and sparse coding. For computational efficiency, use the following top-k rule. Rank $k+1$ top winners so that after ranking, $y_1 \ge y_2 \cdots \ge y_c$, as ranked responses. For superior computational efficiency, this noniterative ranking mechanism replaces repeated iterations that take place among a large number of two-way connected neurons in the same layer. Use a linear function to scale the response

$$y_i' = (y_i - y_{k+1})/(y_1 - y_{k+1}) \tag{21}$$

for $i = 1, 2, \ldots, k$. All other neurons do not fire $y_i = 0$ for $i = k+1, k+2, \ldots, c$. For experiments presented in this paper, $k = 1$. Note: this mechanism of top-k ranking plus scaling is an approximation of biological inhibition. It is not in-place but is very effective computationally when the network update rate is low.
3. Optimal Hebbian learning. Update only the top k winner neurons $\mathbf{v}_j$, for all j in the set of top k winning neurons, using its temporally scheduled plasticity

$$\mathbf{v}_j(t) = w_1 \mathbf{v}_j(t-1) + w_2 y_j \mathbf{x}(t) \tag{22}$$

where the cell's scheduled plasticity is determined automatically by its two update-age dependent weights, called retention rate and learning rate, respectively

$$w_1 = \frac{n(j) - 1 - \mu(n_j)}{n_j}, \quad w_2 = \frac{1 + \mu(n_j)}{n_j} \tag{23}$$

with $w_1 + w_2 \equiv 1$. Update the real-valued neuron "age" $n(j)$ only for the winners: $n_j \leftarrow n_j + y_j', j = 1, 2, \ldots, k$ ($y_j' = 1$ for the top winner).
4. Lateral excitation. Excitatory connections on the same layer are known to exist. To emulate these will encourage cortical representation smoothness. But we do not use these for most experiments in this paper. The discussion on this matter continues in Section IV-E.
5. Long-term memory. All other neurons i that do not update, keep their age and weights unchanged: $\mathbf{v}_i(t) = \mathbf{v}_i(t-1)$.

B. Time and Space Complexities

Given each k-dimensional input $\mathbf{x}$, the *time complexity* for updating c lobe components and computing all the responses from $\mathbf{x}$ is $O(ck)$. Since LCA is meant to run in real-time, this low update complexity is important. If there are t input vectors, the total amount of computation is $O(ckt)$.

Its *space complexity* is $O(ck)$, for c neurons with k-dimensional input $\mathbf{x}$. It is not even a function of the number of inputs t due to the nature of incremental learning.

In fact, the above space and time complexities are the *lowest possible*. Since c vectors need to be computed and each vector has k components, the space complexity cannot be lower than $O(ck)$. Further, the time complexity cannot be lower than $O(ckt)$ because the responses for each of t inputs need that many computations.

Suppose that each lobe component (vector) is considered as a neuron and the number of hits n_j is its clock of "maturity" or "age," which determines the single weight $w_1 (w_2 = 1 - w_1)$ for its updating. The CCI LCA algorithm is an *in-place development* algorithm, in the sense that the network does not need extra storage or an extra developer. The winner-take-all mechanism is a computer simulation of the lateral inhibition mechanism in the biological neural networks. The inhibition-

[6]Here we present linear response with motivation to simplify the system. A nonlinear sigmoidal function is optional, but no matter if a sigmoidal function is used or not, the entire single-layer system is a highly nonlinear system due to the top-k mechanism used.

winner updating rule is a computer simulation of the Hebbian mechanism in the biological neural networks.

C. Convergence Rate

Suppose that the distribution of k-dimensional random input $\mathbf{x}$ is stationary. In CCI LCA, the lobe component vector $\mathbf{v}_i$ converges to the eigenvalue scaled eigenvector $\lambda_{i,1}\mathbf{v}_{i,1}$ in the mean square sense and the speed of convergence is estimated as

$$E_i\|\mathbf{v}_i(t)-\lambda_{i,1}\mathbf{v}_{i,1}\|^2 \approx \frac{2}{t}k\sigma$$

where σ is the estimated average component-wise variance of observation $\mathbf{u}(t)=(\mathbf{x}(t)\cdot\mathbf{v}_i(t-1))\mathbf{x}(t)/\|\mathbf{v}_i(t-1)\|$. *Unlike the conventional sense of convergence,* the convergence is not to a local extremum. It is to the correct solution. The near-optimal convergence speed is due to the use of statistical efficiency in the algorithm design. If the distribution of y changes slowly, the above error estimate is still applicable.

D. Global Perspective: Maximum Mutual Information

A challenge with a local approach such as LCA is as follows: how can the global problem be effectively decomposed into simpler, local ones and how can the solutions to the local problems be integrated into a solution for the global optimal one? We have until this point discussed how LCA solves each local problem. We now provide a theoretical perspective on the global optimality.

Proposition 1: Mutual Information Proposition: In a sensory mapping, the input events occur in input space $\mathcal{R}$ (that, e.g., represents a family of receptive fields $\mathcal{R}\subset 2^{\mathcal{X}}$). The output space (e.g., response or firing space) is $\mathcal{L}$. We propose a major goal of layers of sensory mapping is to maximize the mutual information $I(\mathcal{R},\mathcal{L})$ between the random stimuli events in $\mathcal{R}$ and the random output events in $\mathcal{L}$.
How can we maximize the mutual information? From information theory, we have[7]

$$I(\mathcal{R},\mathcal{L})=H(\mathcal{L})-H(\mathcal{L}\,|\,\mathcal{R}). \tag{24}$$

To maximize the above, we can maximize $H(\mathcal{L})$ while minimizing $H(\mathcal{L}\,|\,\mathcal{R})$.

We divide the much larger number of discrete samples in $\mathcal{R}$ into c discrete bins $\mathcal{R}=R_1\cup R_2\cup\cdots\cup R_c$. In (24), the first term is the entropy of representation by c bins and the second term is the expected uncertainty of $l\in\mathcal{L}$, given input $r\in\mathcal{R}$. We want to maximize the entropy of the representation $H(\mathcal{L})$ and minimize $H(\mathcal{L}\,|\,\mathcal{R})$. To maximize the first term, we use the *equal probability principle*. The partition of $\mathcal{R}$ should be such that each R_i has the same probability. To minimize the second term, we use the *multiple support principle*. Given input image $r\in\mathcal{R}$, $H(\mathcal{L}\,|\,r)$ is zero.

1) Equal Probability Principle: Suppose that the output event $y\in\mathcal{L}$ is represented by event $y\in R_i$. To maximize the first term in (24), we know that a uniform distribution across the c regions has the maximum entropy $H(\mathcal{L})$ if every region has the same probability.

[7]$H(A\,|\,B)$ denotes the conditional entropy $H(A\,|\,B)=E\log[p(A\,|\,B)]$, where $p(A\,|\,B)$ is the probability density of A conditioned on B.

We have the following theorem.

Theorem 1: The Maximum Mutual Information Theorem: Suppose that the output is represented by discrete event $r\in R_i$. Then, the mutual information $I(\mathcal{R},\mathcal{L})$ is maximized if the following conditions are satisfied.

1) All the regions $R_1,R_2,\ldots,R_c$ have the same probability.
2) The representation $l\in\mathcal{L}$ is completely determined by event $r\in R_i$, for all i with $1\le i\le c$.

Proof: $I(\mathcal{R},\mathcal{L})=H(\mathcal{L})-H(\mathcal{L}\,|\,\mathcal{R}), I(\mathcal{R},\mathcal{L})$ is maximized if we maximize $H(\mathcal{L})$ while minimizing $(\mathcal{L}\,|\,\mathcal{R})$. Condition 1) is a necessary and sufficient condition to maximize $H(\mathcal{L})$ for a given limited c, the number of cells (or regions) [38, pp. 513–514]. Condition 2) is equivalent to $H(\mathcal{L}\,|\,\mathcal{R})=0$ for discrete distribution. Since the entropy of a discrete distribution is never zero, it reaches the maximum when $l\in\mathcal{L}$ is completely determined when $r\in R_i$, for all $1\le i\le c$. □

Condition 1) in Theorem 1 means that every neuron in the layer fires equally likely. Towards this goal of equal-probability partition, neurons update in-place using optimal Hebbian learning and winner-take-all competition (lateral inhibition) we discussed earlier. Smoothness in self-organization is a way to approach equal probability. However, due to the cost of updating, equal probability is approached but is not reached exactly. Condition 2) requires that the response from the layer completely determine the input. This means that the coding (response) is not random and catches the variation in the input space as much as possible.

E. Topographic LCA

Cortical lateral excitation can encourage equal probability, as discussed above, since it will "pull" more neurons to the higher density areas. Although it is critical, we only briefly mention it here, since it is mostly out of this paper's scope. One method of lateral excitation is as follows. Update the other neurons in a 3×3 neighborhood around every top-k winner, simulating 3×3 lateral excitation. Each neighboring neuron is updated as a fraction $r(d)=1-d/2$ of full update, where d is the distance between the updating neuron and the winner. The learning rate is $r(d)w_2(n)$, with $w_1(n)=1-w_2(n)$ and the (real valued) age is advance by $r(d)$.

V. Experimental Results

We now present comparisons of the CCI LCA algorithm with other incremental neuronal updating rules and compare with ICA algorithms. Results show the degree of benefit of the statistical near-optimal efficiency of the CCI LCA algorithm.

A. Comparison With Other Neuron Updating Rules

1) Introduction to Other Methods: The basic Hebbian form [39], [40] for updating the weight vector $\mathbf{v}$ of a neuron

$$\Delta\mathbf{v}=\eta y(\mathbf{v},\mathbf{x})\mathbf{x} \tag{25}$$

where $\mathbf{v}$ is the amount of update for the weight vector $\mathbf{v}$ by executing $\mathbf{v}\leftarrow\mathbf{v}+\Delta\mathbf{v}$, η the learning rate, and $\mathbf{x}$ the vector input (presynaptic activity).

Oja's classic neuron updating algorithm [41] is an algorithm that follows (25) for incrementally computing the first principle

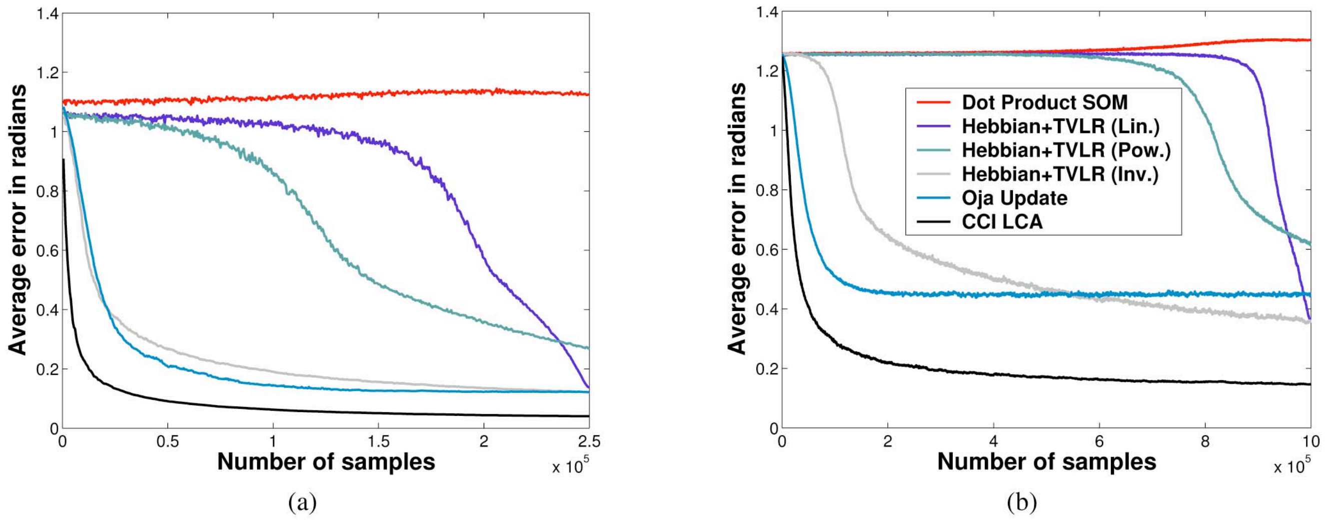


Fig. 5. Comparison of incremental neuronal updating methods (best viewed in color). The legend in the right figure applies to both figures. We compare in (a) 25 and (b) 100 dimensions. Methods used were i) "dot-product" SOM; ii) Oja's rule with fixed learning rate 10^{-3}; iii) standard Hebbian updating with three functions for tuning the time-varying learning rates: linear, power, and inverse; and iv) CCI LCA. LCA, with its temporal optimality, outperforms all other methods. Consider this a "race" from start (same initialization) to finish (0% error). Note how quickly it achieves short distance to the goal compared with other methods. For example, in (a) after 5000 samples, LCA has covered 66% of the distance, while the next closest method has only covered 17% distance. Similarly, in (b), CCI LCA beats the compared methods. For example, after 28 500 samples, when LCA has covered 56% distance, the next closest method has only covered 24% distance.

component, which is spatially optimal, as we discussed in earlier sections. Its NIO is response-weighted input

$$\Delta \mathbf{v} = \eta \mathbf{y}(t)(\mathbf{x}(t) - \mathbf{y}(t)\mathbf{v}(t)) \tag{26}$$

where $\mathbf{y}(t) = \mathbf{x}^T(t)\mathbf{v}(t)$ is the neuronal response. This version should be used with small η (e.g., $\eta = 10^{-3}$) for stability. If stable, the lengths of the vectors will tend to unit.

A stable two-step version of (26) that aligns directly with (25) and uses time-varying η is

$$\Delta \mathbf{v} = \eta(t)(\mathbf{x}^T(t)\mathbf{v}(t))\mathbf{x}(t), \quad \mathbf{v} \leftarrow \mathbf{v}/\|\mathbf{v}\|. \tag{27}$$

We called it "Hebbian with time-varying learning rate (TVLR)".

The "dot-product" version of the self-organizing map (SOM) updating rule [42, p. 115] is also considered as incremental neuronal learning

$$\mathbf{v}_i(t+1) = \frac{\mathbf{v}_i(t) + \eta(t)\mathbf{x}(t)}{\|\mathbf{v}_i(t) + \eta(t)\mathbf{x}(t)\|} \tag{28}$$

where $\mathbf{v}_i$ is the winning component vector at time t. Note a major difference between the dot-product SOM and the others: the NIO used by SOM's rule $\mathbf{u} = \mathbf{x}$ (not weighted by response). Without response-weighting, this updating rule did not perform successfully in our tests.

All of the above use a single learning rate parameter to adapt the neuron weights to each new updating input and a method to bound the strengths of synaptic efficacies (e.g., vector normalization). CCI LCA weights using the time-varying retention rate $w_1(t)$ and learning rate $w_2(t)$, where $w_1(t) \equiv w_2(t)$, in order to maintain the energy estimate. With the energy gone in the three schemes above, there is no way to adjust the learning rate $\eta(t)$ to be equivalent to the CCI scheduling. Therefore, the result of (26)–(28) cannot be optimal.

2) Stationary Distributions: The statistics of natural images are known to be highly non-Gaussian [43], and the responses of V1 neurons to natural input have a response profile characterized by high kurtosis. The Laplacian distribution is non-Gaussian and has high kurtosis, so we test estimation of the principle component of Laplacian distributions.

The data generated are from a d-dimensional Laplacian random variable. Each dimension has a pdf of $f(x \mid \mu', b) = (1)/(2b) \exp(-(\mid x - \mu' \mid)/(b))$. All dimensions had zero mean ($\mu' = 0$) and unit variance ($b = 1$) for fairness (LCA can handle higher variances, but the other methods will not do well since they are designed to extract components with unit energy). The true components to be extracted from this distribution are the axes. We do not use a rotation matrix for this experiment. So, the true components orthogonally span a d-dimensional space. We use a number of neurons equal to dimensionality, initialized to random samples drawn from the same distribution. For a fair comparison, all methods started from the same initialization. The training length (maximum number of data points) was $T = 10\,000d$, so that each neuron would on average have 10 000 updates. Dimension d was 25 or 100. Results were averaged over 50 trials. The results measure average correlation between each component, which is a unit vector, and the closest neuron (in inner product space).

For tuning the time-varying learning rate $\eta(t)$, we used three example suggested [44] learning rates for $\eta(t)$, which were "linear" $\eta(t) = \eta(0)(1 - t/T)$, "power" $\eta(t) = \eta(0)(0.005/\eta(0))^{t/T}$, and "inv" $\eta(t) = \eta(0)/(1 + 100t/T)$. The initial learning rate $\eta(0)$ was 0.1 or 0.5. Plasticity parameters for LCA's μ were $t_1 = 10, t_2 = 100, c = 5, r = 5000$.

Results are shown in Fig. 5. The "SOM" curve shows the best performing variant among the six different learning rate functions and initial learning rates, as suggested [44]. None of them led to extraction of the true components (the best one uses

$\eta(0) = 0.1$ and the linear tuning function—in both cases). For Oja's rule with time-varying learning rate, we show only $\eta(0) = 0.1$ since the alternate curves ($\eta(0) = 0.5$) were uniformly worse. These results show the effect of LCA's statistical efficiency. In 25 dimensions, when LCA has achieved 20% error, the best other Hebbian method has only achieved 60% error. Similarly, in 100 dimensions, when LCA has achieved 30% error, the best compared method is still at 70% error. The results for LCA will not be perfect due to the nonstationarity that occurs due to self-organization, but they are much better than the other methods.

3) Time-Varying Distributions: It is important for an agent to have the capability to adapt to new environments without catastrophic forgetting of what was already learned. This challenging problem has not been adequately addressed by existing self-organization methods. Our latest understanding from our brain-scale modeling can be summarized as follows.

a) Fast learning without representation change: Local lobe components that are computed by early cortical layers are low-level, which do not change substantially across ages. But the distribution of high-level features, computed by later cortical areas, can change substantially at higher ages. This is not, however, mainly due to synapse representational changes. Instead, this fast change is mainly due to fast association changes and attentionally modulated competition among actions. This computational concept is challenging, new, and closely related to the LCA theory here. We called it the principle of fast learning from stable representation.

For example, when an adult agent is confronted with a novel object, how can the agent learn the novel object quickly? Suppose that a person who has never seen a palm tree before has been told the name "palm tree" and can say "palm tree." Upon his arrival in Florida, how is he able to nearly immediately learn and recognize similar trees as palm trees? How can an LCA neuronal layer update so quickly in order to accommodate such fast learning? We know that the brain updates at least around 1 KHz (e.g., a spike lasts about 1 ms). Within the half-second time it may take for the individual to learn the palm tree concept, hundreds of rounds of network iterations occur. But even this will not be enough to learn a brand new representation.

Fast learning does not imply the distribution of neuronal synapses drastically updates. Instead, it occurs by the generation of new firing patterns based on the established stable cortical representations (early and later layers) and the *association* of the new firing patterns to the corresponding stable motor actions. Consider that LCA layer $l-1$ has already learned edges, leaves, trunks, etc., and LCA layer $l+1$ has already learned actions for verbal "pine tree," "palm tree," "new tree," etc. In our temporal MILN model [45], the intermediate LCA layer l takes input $\mathbf{p}$ as a combination of bottom-up input $\mathbf{x}$ of layer $l-1$ (from a stable representation) and top-down input $\mathbf{z}$ from layer $l+1$ (also from a stable representation). Suppose that while the newcomer is looking at the image of a palm tree, his friend says "palm tree!" Now, it is important to know the well-known phenomenon of "mirror neuron." Because of online learning, an auditory "palm tree" input must trigger the firing of verbal action "palm tree." This is because when he produced verbal "palm tree" he heard his own auditory "palm" at the same time and such an auditory-to-action association was established. With $\mathbf{p} = (\mathbf{x}, \mathbf{z})$ as the input to layer l, and noting that the representation of layer l is stable, a new response pattern is generated from the LCA output $\mathbf{y}$ from layer l. This firing pattern $\mathbf{y}$ in layer l strengthens the bottom-up weight vector of the firing representation of "palm tree" in layer $l+1$, through LCA's optimal Hebbian learning. A few rounds of network iterations are sufficient to surpass the bottom-up weight vector of the "default" non-firing representation of "new tree." This illustrates the power of the top-down attention (action) signal. Slight changes in synapses can greatly change the winner of attention selection. This theory of fast learning from stable representation needs to be demonstrated experimentally in the future.

b) Representation adaptation: Next, we demonstrate the change of the distribution of synapses, which is expected to be relatively slow according to our above discussion. We performed a comparison of how well the best performing of the algorithms we tested before adapt to a time-varying distribution. We set up a *changing environment* as follows.

There are five phases. In the first phase, until time 200 000, the data are drawn from 70 orthogonal Laplacian components that span a 70-dimensional space. In the second phase, from time 200 000 to 399 999, the data are drawn from one of ten *new* components—meaning we simply use a different rotation matrix and thus do not increase dimensionality, with a 50% chance *or* from one of the original 70 (using the original rotation matrix) with 50% chance. This is motivated by how a teacher will emphasize new material to the class, and only more briefly review old material. In the third phase, from time 400 000 to 599 999, the data are drawn from either ten brand new components or the original 70 (50% chance of either). The fourth phase, until time 799 999, is similar—ten previously unseen components are introduced. In the fifth phase, until $T = 1\,000\,000$, we draw from all 100 possible components (and each has a 1% probability).

We use 100 neurons over all phases (never increases or decreases). So, there are finally 100 neurons for 100 components, but in early phases we have extra resource (e.g., in phase one, we have 100 neurons for 70 components). Results are averaged over 50 runs with different rotation matrices for each run. They are shown in Fig. 6 and discussed in the caption. LCA outperforms the other two variants—it is better at adaptation and suffers a more graceful forgetting of data that is not currently observed. We note that the "relearning" in the last phase does not match the previously observed performance. This is due to two reasons: the lessening of plasticity for larger neuron ages and the increasing of the manifold of the data while retaining only a fixed representation resource. The superior performance is due to the dual optimality of LCA.

B. Comparison With ICA

ICA has been proposed as a computational model for neuronal feature development. LCA is based on biologically inspired neuronal inhibitory and excitatory connections, biological in-place neuronal Hebbian learning, the optimality in spatial representation, and the optimality in the temporal course of learning. In contrast, ICA is mainly based on a mathematical

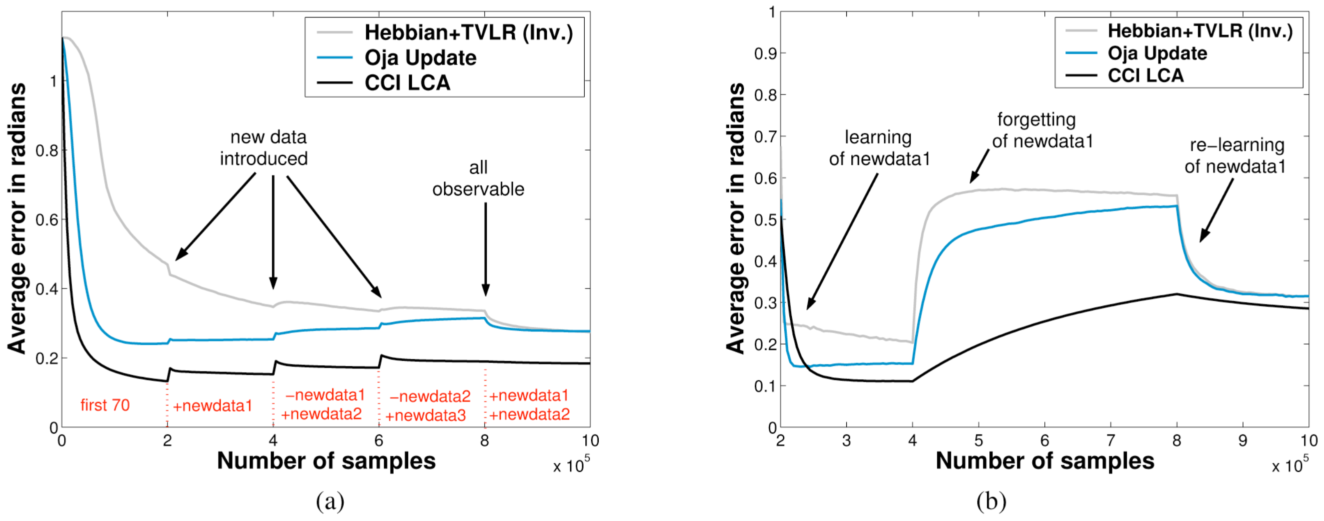


Fig. 6. Comparison of LCA with two other Hebbian learning variants for a time-varying distribution. (a) shows average error for all available components. There are 70 available until time 200 000, 80 until 400 000, 90 until 600 000 and 100 until 1 000 000. We expect a slight degradation in overall performance when new data are introduced due to the limited resource always available (100 neurons). The first jump of LCA at $t = 200\,000$ is a loss of 3.7% of the distance it had traveled to that point. (b) shows how well the neurons adapt to the ten components added at time 200 000 (called newdata1) and then how well they remember them (they are observed in only the second and fifth phases). Initially, these new data are learned well. At time 400 000, newdata2 begins to be observed, and newdata1 will *not* be observed until time 800 000. Note the "forgetting" of the non-LCA methods in comparison to the more graceful degradation of LCA. The plots focusing on newdata2 and newdata3 are similar.

assumption that responses from different neurons are statistically independent. The representation of each lobe component is *local* in the input space, realized through lateral inhibition among neurons in the same layer. The global representation of LCA arises from the firing pattern of many lobe components. In contrast, the representation of ICA is *global* because of the use of higher order statistics of the entire input space.

1) Introduction to ICA: ICA was shown to extract localized orientation features from natural images [24]. In many experiments, the (global) ICA gives superior features compared to global PCA. As is well known, statistical independence used by ICA is a much stronger condition than uncorrelatedness used by PCA. But due to this condition, ICA algorithms are complex. They are not in-place algorithms.

The original linear data model used in ICA is as follows. There is an unknown d-dimensional random signal source $\mathbf{s}$, whose components are mutually statistically independent. For every time instance $t = 1, 2, \ldots$, an unknown random sample $\mathbf{s}(t)$ is generated from the signal source. There is an unknown $d \times d$ constant, full-rank mixing matrix A, which transforms each column vector $\mathbf{s}(t)$ into an observable vector $\mathbf{x}(t)$

$$\mathbf{x}(t) = A\mathbf{s}(t) = \sum_{i=1}^{d} \mathbf{a}_i s_i(t) \tag{29}$$

where $\mathbf{a}_i$ is the ith column of A and $s_i(t)$ is the ith component of $\mathbf{a}(t)$. The goal of ICA is to estimate the matrix A. However, A cannot be determined completely. $\mathbf{s}(t)$ is generally assumed to have zero mean and unit covariance matrix, which implies that the matrix A can be determined up to a permutation of its columns and their signs.

For many other applications, however, it is not necessarily true that the signal source is driven by a linear combination of independent components. For example, there is no guarantee that video images of natural environments contain any truly independent components. The natural scene observed by a camera is the projection of multiple dynamic (moving) objects, which is much more complex than the pure linear model in (29). For example, if there are d independent objects in the scene where $s_i(t)$ independently controls its appearance, then $\mathbf{a}_i$ typically is not static (caused by, e.g., motions, lighting changes, viewing geometry changes, deformation, etc.). Therefore, the A matrix is not a constant matrix.

In ICA, a "demixing" matrix B is applied to $\mathbf{x}(t)$ so that the new components of $\mathbf{y} = B\mathbf{x}$ are mutually independent. However, because of the above model problem in (29), such a demixing matrix might not exist. In practice, ICA computes B so that the components are mutually independent as much as possible, regardless of whether the model in (29) is valid or not.

2) Experimental Comparison With ICA: LCA is ICA for super-Gaussian components. Components that have a super-Gaussian distribution roughly correspond to lobe components we defined here. Each linear combination of k super-Gaussian independent components corresponds to symmetric lobes, illustrated in Fig. 2(a). Therefore, if components in $s(t)$ are all super-Gaussian, finding lobe components by CCI LCA is roughly equivalent to finding independent components, but with different theory and much lower computational complexity.

The optimal statistical efficiency appears to drastically improve the capacity to deal with high-dimensional data. We selected two state-of-the-art incremental ICA algorithms, Type-2 Extended Bell–Sejnowski (ExtBS) [25], and Type-3 (NPCA-LS) [46], [47] for performance comparison with the proposed Type-5 CCI LCA algorithm. The choice of these two algorithms is due to their superior performance in the comparison results of [48]. We used the downloaded code from the authors for the ExtBS algorithm.

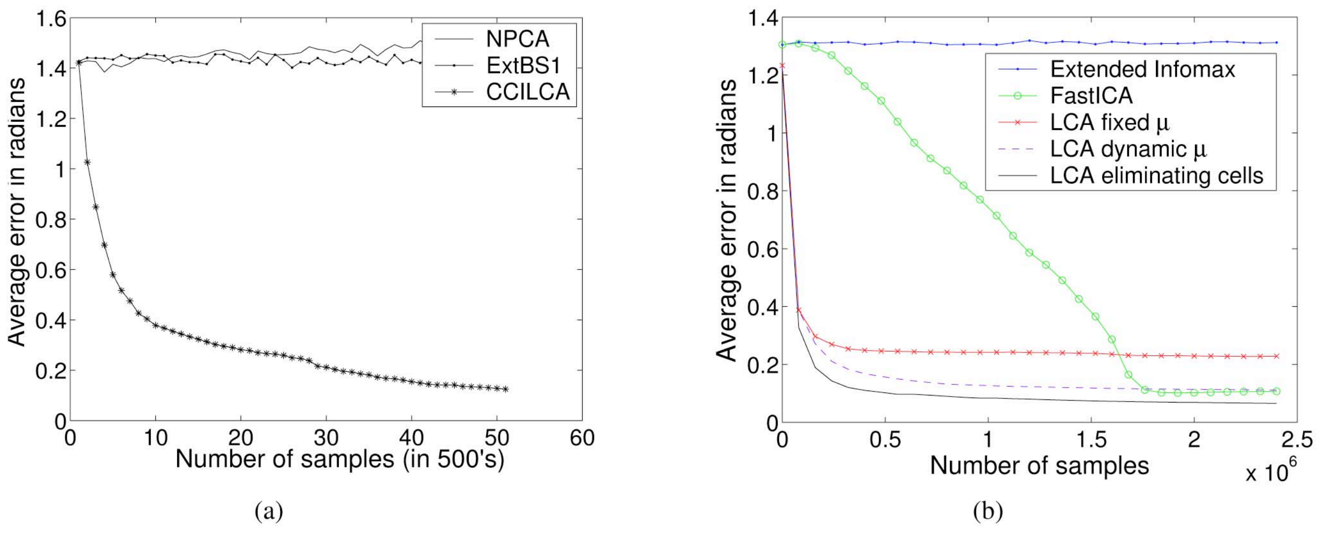


Fig. 7. (a) Comparison of ICA results among (Type-3) NPCA, (Type-2.5) ExtBS1, and (Type-5) CCI LCA for super-Gaussian sources in 25 dimensions. (b) Comparison of ICA results among (Type-2) Extended Infomax, (Type-1) FastICA, and three variants of the proposed Type-5 CCI LCA algorithm in 100 dimensions.

The NPCA algorithm used was proposed in [46] with $\beta = 0.998$ and g as tanh. The ExtBS algorithm was run with the following set of parameters: blocksize $= 1$, learning rate $= 0.001$, learning factor $= 0.985$, momentum constant $= 0.1$, number of iterations $= 1$, step $= 5$, and block size for kurtosis estimation is 1000. This version is called ExtBS1, the number 1 indicating the block size for updating. Thus, ExtBS 1 is a partial sequential algorithm, sequential for independent component update, but computation for kurtosis is block-incremental.

As in the earlier experiment section, each independent source is a Laplacian component. The mixing (rotation) matrix was chosen randomly and was nondegenerate. The error between the direction of the true and the estimated independent components is measured as the angle between them in radians. All results were averaged over 50 runs.

As indicated by the Fig. 7(a), both the Type-3 algorithm NPCA and Type-2.5 ExtBS1 did not converge for a moderate dimension of $d = 25$, although ExtBS1 did fine when $d = 2$. The proposed CCI LCA did well.

Next, we compared our CCI LCA algorithm with Type-2 Extended Bell–Sejnowski (or extended infomax) [25] with block size 1000 and Type-1 batch algorithm FastICA [22], [23]. Convergence with respect to the number of samples used in training is a good evaluation of the efficiency of ICA algorithms. This is not a fair comparison since a Type-5 algorithm (like CCI LCA) should not be expected to outperform a Type-1 or Type-2 algorithm. We compared them anyway to understand the limit when CCI LCA is compared with two state-of-the-art Type-1 and Type-2 algorithms.

It is well known that ICA algorithms require a significant amount of data for convergence. Typically, even for a low-dimension simulation task (e.g., $d = 2$), ICA algorithms need thousands of samples to approach the independent components. The number increases with the number of components as well. Some ICA algorithms may not converge at a high dimension with many components for many thousands of samples.

For a higher dimension, we synthetically generated random observations from an i.i.d. Laplacian random vector with dimension of 100. The results are shown in Fig. 7(b), where the x-axis marks the number of samples and the y-axis indicates the average error in radians. In order to show more detailed aspects of CCI LCA, three variations have been tested. "LCA with fixed μ" and "LCA with dynamic μ" are original LCA methods with a fixed $\mu(n_j)$ and varying $\mu(n_j)$, as defined in (15), respectively. The "LCA eliminating cells" algorithm dynamically eliminates cells whose hitting rate is smaller than 3/4 of the average hitting rate, since sometimes two vectors share a single lobe (which is rare and does not significantly affect the purpose of density estimation by lobe components) but does affect our error measure. As shown in Fig. 7(b), all the three LCA algorithms converged very fast—faster than the Type-2 algorithm Extended infomax and even the Type-1 FastICA. The batch Extended Infomax algorithm needs more samples at this high dimension, and it did not converge in these tests.

It was somewhat surprising that the proposed CCI LCA algorithm, operating under the most restrictive condition, outperforms the state-of-the-art Type-3, Type-2, and Type-1 algorithms by a remarkably wide margin (about 20 times faster to reach 0.4 average error in Fig. 7(b). This is due to the new lobe component concept and the optimal property of the statistical efficiency.

C. Blind Source Separation

The goal of blind source separation (BSS) [49] is to find $\mathbf{s}$ up to a scale factor from $\mathbf{x}$ in (29). The BSS problem traditionally uses ICA. From the definition of LCA, we can see that independent components, which are along the major axes in $\mathbf{s}$ (also lobe components) are, after linear mixing in (29), still lobe components in $\mathbf{x}$ because the space is rotated, skewed, and scaled by the transformation matrix in (29).

We have tested the LCA algorithm on a simulation of the cocktail party problem. Nine sound sources are mixed by a randomly chosen full rank matrix. Each sound source is 6.25 s long

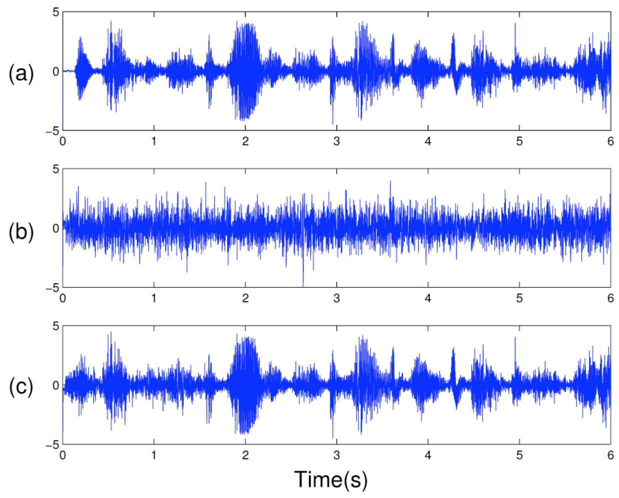


Fig. 8. Cocktail party problem. (a) A music sound clip in its original form. It is one of the nine sound sources. (b) One of the nine mixed sound signals. (c) The recovered music sound wave. Compared to (a), the sound signal can be considered recovered after approximately 1.5 s.

and the sampling rate is 8.0 KHz in 8 bits mono format. Therefore, each sound source contains 50 000 values.[8]

Fig. 8(a) shows one of the nine original source signals. Fig. 8(b) displays one of the nine mixed sound signals. The mixed signals are first whitened; then we applied the proposed algorithm to the mixed sound signals. It is worth noting that the proposed algorithm is an incremental method. Therefore, unlike other batch ICA methods that require iterations over the data set, we have used the data only once and then discarded it. Results are shown in Fig. 8(c). The independent components quickly converge to the true ones, with a good approximation as early as 1.5 s.

D. *Visual Filter Development*

Inspired by results where ICA algorithms were shown to extract orientation filters from natural images, we conducted an experiment using CCI LCA on natural image patches. Five hundred thousand of 16×16-pixel image patches were randomly taken from 13 natural images.[9] The CCI LCA algorithm was applied to the prewhitened image patches $k = 16 \times 16 = 256$ to update the lobe component matrix V. The matrix $(B^\top)^{-1}$ was then computed. Each column of the matrix is shown in Fig. 9(a) by a 16×16 patch as the features of the natural scenes. A total of 256 256-dimensional vectors are displayed in the figure. They all look smooth, and most of them are localized (only a small patch are nonzero, or gray), as expected. The entire process took less than 46 min (i.e., 181 frames/s) on a Pentium III 700-MHz PC with 512 MB memory compared to over 10 h of learning time for the FastICA algorithm using 24% of the 500 000 samples (disk thrashing is also a factor).

Fig. 9(b) shows how many times each lobe component was the top "winner." Most components have roughly a similar rate of hits, except relatively few leftmost (top) ones and rightmost (tailing) ones. Although it is not exactly true that each lobe component is equally likely to be hit, nearly equal hits for the majority is a desirable property for high-dimensional density estimation due to the criteria of maximum mutual information we explained in Section IV-D.

Fig. 10 displays the filters of a simple topographic variant of the LCA algorithm, where the winning neurons' neighbors will also update at a reduced rate (3×3 neighborhood updating kernel). Filters show iso-orientation preference in a neighborhood.

VI. Conclusion

The CCI LCA theory here provides a theoretical basis of the biological Hebbian incremental direction and further predicts a firing-age-dependent plasticity schedule for a biological neuron.

The in-depth theoretical discussion of the CCI LCA framework here explains the dual optimality of LCA in terms of its spatial and temporal optimality. This dual optimality led to the demonstrated drastic advantages in the speed and success of component extraction, as shown in comparisons with other incremental neuronal updating methods, which use a single learning rate instead of LCA's optimally tuned learning rate and retention rate. The CCI LCA was shown to outperform other Hebbian learning methods, which are not based on statistical efficiency, as well as several ICA algorithms, including Type-1 FastICA.

For the future of AMD, there is a critical need for Type-5 algorithms. Due to its simple structure, lowest possible order of time and space complexities, optimal statistical efficiency, and the Type-5 nature, we expect that this class of algorithms will be widely used.[10]

Appendix
Sigmoidal for Robust Neuronal Processing

A sigmoidal function is effective to suppress outliers in the sense of robust statistics [51]. The inner product $\mathbf{x}(t) \cdot \mathbf{v}_i$ can dominate when the projection to some neurons is very large. We might change to a more robust, biologically plausible response, which is rescaled by the monotone sigmoidal function g_i to give $z_i = g_i(\mathbf{x} \cdot \mathbf{v}_i)$, where $g_i(x)$ is the sigmoidal function of neuron represented by $\mathbf{v}_i$. The sigmoidal function $g_i(x)$ has a unit derivative at the mean of x since it replaced a scale factor 1 in the original belongingness $\mathbf{x}(t) \cdot \mathbf{v}_i$.

In order to give a single, "gene-specified" sigmoidal function $g(x)$ that is applicable to all the neurons, we divide the input to $g_i(x)$ by the incremental average of $x, \rho_i = \sqrt{E(x^2)}$, which results in a same sigmoidal function for all the neurons

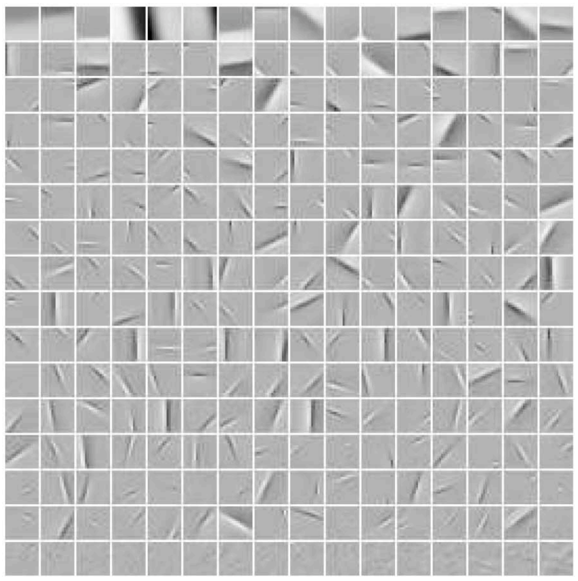
$$g_i(\mathbf{x}(t) \cdot \mathbf{v}_i) = g(\mathbf{x} \cdot \mathbf{v}_i / \rho_i). \tag{30}$$

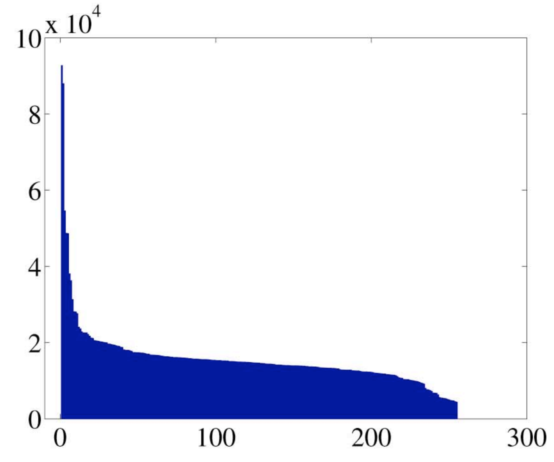
This effect of forcing (30) has the effect of compressing along $\mathbf{v}_i$ so that the standard deviation is the same, before computing

[8]We downloaded the sound clips from http://www.cis.hut.fi/projects/ica/cocktail/cocktail_en.cgi, where they used these sound clips to test the FastICA algorithm [50]

[9]Available from http://www.cis.hut.fi/projects/ica/imageica/.

[10]Code for the LCA algorithm is freely available at www.cse.msu.edu/ei/software.htm.


(a)



(b)

Fig. 9. Lobe components from natural images (not factorizable). (a) LCA derived features from natural images, ordered by the number of hits in decreasing order. (b) The numbers of hits of the corresponding lobe components in (a).

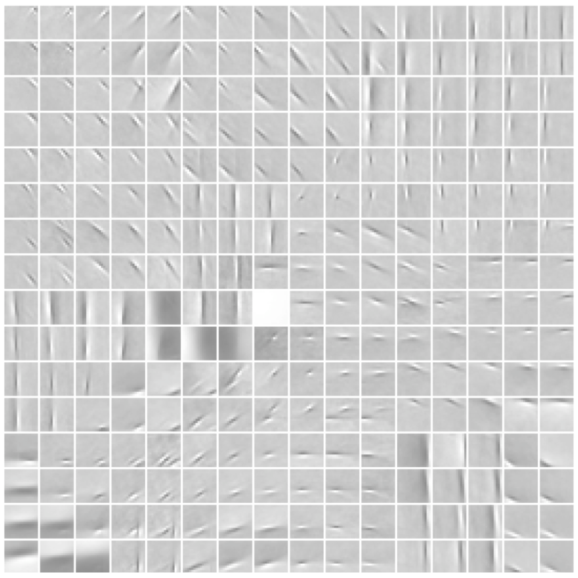


Fig. 10. Topographically ordered basis functions developed by LCA from natural image patch data.

the equal-angle boundary. This effect can be seen in Fig. 2(c), where the boundary of regions are closer to the diagonal lines.

Biologically, the average "power" of the projections would have to be recorded in-place by the neuron to become the neuron specific sigmoidal function g_i. This procedure is in-place as the information is available in-place. Each neuron can keep an internal variable to incrementally estimate the power of its own pre-action potential. This leads to a new definition of belongingness.

Definition 2: Belongingness of $\mathbf{x}$ to R_i is defined as the response $y_i = g(\mathbf{x} \cdot \mathbf{v}_i \|\mathbf{v}_i\|^{-2})$, where $\mathbf{v}_i$ is the candid lobe component vector representing region R_i.

Belongingness, as defined above, uses a standard sigmoidal function $g(x)$ for all the neurons in the cortex. The factor $\|\mathbf{v}_i\|^{-2}$ contains two factors of $\|\mathbf{v}_i\|^{-1}$: one for normalizing the length of $\mathbf{v}_i$ and the other for normalizing the project of $\mathbf{y}$ onto the direction of $\mathbf{v}_i$. In other words, $\rho_i \approx \|\mathbf{v}_i\|^{-1}$.

Then, the new response, which is re-scaled by the monotone sigmoidal function g_i, is

$$\mathbf{v}_i = \frac{1}{n}\sum_{t=1}^{n} g_i\left(\frac{\mathbf{x}(t)\cdot\mathbf{v}_i}{\|\mathbf{v}_i\|}\right)\mathbf{x}(t) \tag{31}$$

where $\mathbf{v}_i$ has a length $\lambda_{i,1}$.

The sigmoidal function $g_i(x)$ in (31) has a unit derivative at the mean of x since it replaced a scale factor 1 in (10). We can see that the mean of x is the average projection of $\mathbf{x}$ on the unit $\mathbf{v}_i$ or the currently estimated $\|\mathbf{v}_i\|$.

ACKNOWLEDGMENT

The authors would like to acknowledge N. Zhang, who generated the results shown in Figs. 7–10.

REFERENCES

[1] J. L. Elman, E. A. Bates, M. H. Johnson, A. Karmiloff-Smith, D. Parisi, and K. Plunkett, *Rethinking Innateness: A Connectionist Perspective on Development*. Cambridge, MA: MIT Press, 1997.
[2] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen, "Autonomous mental development by robots and animals," *Science*, vol. 291, no. 5504, pp. 599–600, 2001.
[3] J. Weng and I. Stockman, "Autonomous mental development: Workshop on development and learning," *AI Mag.*, vol. 23, no. 2, pp. 95–98, 2002.
[4] R. Miikkulainen, J. A. Bednar, Y. Choe, and J. Sirosh, *Computational Maps in the Visual Cortex*. Berlin, Germany: Springer, 2005.
[5] G. J. McLachlan, *The EM Algorithm and Extensions*. New York: Wiley, 1997.
[6] J. Weng and N. Zhang, "In-place learning and the lobe component analysis," in *Proc. IEEE World Congr. Comput. Intell.*, Vancouver, BC, Canada, Jul. 16–21, 2006.

[7] J. Weng, T. Luwang, H. Lu, and X. Xue, "A multilayer in-place learning network for development of general invariances," *Int. J. Human. Robot.*, vol. 4, no. 2, pp. 281–320, 2007.

[8] M. D. Luciw and J. Weng, "Topographic class grouping and its applications to 3d object recognition," in *Proc. IEEE/INNS Int. Joint Conf. Neural Netw.*, Hong Kong SAR, China, 2008.

[9] J. Weng, T. Luwang, H. Lu, and X. Xue, "Multilayer in-place learning networks for modeling functional layers in the laminar cortex," *Neural Netw.*, vol. 21, pp. 150–159, 2008.

[10] D. J. Felleman and D. C. Van Essen, "Distributed hierarchical processing in the primate cerebral cortex," *Cerebral Cortex*, vol. 1, pp. 1–47, 1991.

[11] E. M. Callaway, "Local circuits in primary visual cortex of the macaque monkey," *Annu. Rev. Neurosci.*, vol. 21, pp. 47–74, 1998.

[12] A. K. Wiser and E. M. Callaway, "Contributions of individual layer 6 pyramidal neurons to local circuitry in macaque primary visual cortex," *J. Neurosci.*, vol. 16, pp. 2724–2739, 1996.

[13] S. Grossberg and R. Raizada, "Contrast-sensitive perceptual grouping and object-based attention in the laminar circuits of primary visual cortex," *Vision Res.*, vol. 40, pp. 1413–1432, 2000.

[14] S. Grossberg, "Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors," *Biol. Cybern.*, vol. 23, pp. 121–131, 1976.

[15] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Comput. Vision, Graph., Image Process.*, vol. 37, pp. 54–115, 1987.

[16] , E. R. Kandel, J. H. Schwartz, and T. M. Jessell, Eds., *Principles of Neural Science*, 4th ed. New York: McGraw-Hill, 2000.

[17] M. Kirby and L. Sirovich, "Application of the Karhunen-Loéve procedure for the characterization of human faces," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 103–108, Jan. 1990.

[18] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cogn. Neurosci.*, vol. 3, no. 1, pp. 71–86, 1991.

[19] K. Etemad and R. Chellappa, "Discriminant analysis for recognition of human face images," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, Atlanta, GA, May 1994, pp. 2148–2151.

[20] D. L. Swets and J. Weng, "Using discriminant eigenfeatures for image retrieval," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, no. 8, pp. 831–836, 1996.

[21] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs fisherfaces: Recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 711–720, Jul. 1997.

[22] A. Hyvarinen and E. Oja, "A fast fixed-point algorithm for independent component analysis," *Neural Comput.*, vol. 9, no. 7, pp. 1483–1492, 1997.

[23] A. Hyvarinen and E. Oja, "Independent component analysis: Algorithms and applications," *Neural Netw.*, vol. 13, no. 4–5, pp. 411–430, 2000.

[24] A. J. Bell and T. J. Sejnowski, "The 'independent components' of natural scenes are edge filters," *Vision Res.*, vol. 37, no. 23, pp. 3327–3338, 1997.

[25] T. W. Lee, M. Girolami, and T. J. Sejnowski, "Independent component analysis using an extended infomax algorithm for mixed sub-Gaussian and super-Gaussian sources," *Neural Comput.*, vol. 11, no. 2, pp. 417–441, 1999.

[26] J. Karhunen and P. Pajunen, "Blind source separation using least-squares type adaptive algorithms," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Munich, Germany, 1997, pp. 3048–3051.

[27] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Netw.*, vol. 4, pp. 759–771, 1991.

[28] J. Weng, Y. Zhang, and W. Hwang, "Candid covariance-free incremental principal component analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 8, pp. 1034–1040, 2003.

[29] M. B. Feller, D. P. Wellis, D. Stellwagen, F. S. Werblin, and C. J. Shatz, "Requirement for cholinergic synaptic transmission in the propagation of spontaneous retinal waves," *Science*, vol. 272, no. 5265, pp. 1182–1187, 1996.

[30] J. C. Crowley and L. C. Katz, "Development of cortical circuits: Lessons from ocular dominance columns," *Nature Rev. Neurosci.*, vol. 3, pp. 34–42, 2002.

[31] C. W. Cotman and M. Nieto-Sampedro, "Cell biology of synaptic plasticity," *Science*, vol. 225, pp. 1287–1294, 1984.

[32] W. K. Purves, D. Sadava, G. H. Orians, and H. C. Heller, *Life: The Science of Biology*, 7th ed. Sunderland, MA: Sinauer, 2004.

[33] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. London, U.K.: Chapman and Hall, 1986.

[34] I. T. Jolliffe, *Principal Component Analysis*. New York: Springer-Verlag, 1986.

[35] Y. Tang, J. R. Nyengaard, D. M. De Groot, and H. J. Gundersen, "Total regional and global number of synapses in the human brain neocortex," *Synapse*, vol. 41, no. 3, pp. 258–273, 2001.

[36] E. L. Lehmann, *Theory of Point Estimation*. New York: Wiley, 1983.

[37] J. Weng, T. S. Huang, and N. Ahuja, *Motion and Structure From Image Sequences*. New York: Springer-Verlag, 1993.

[38] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 2nd ed. New York: McGraw-Hill, 1976.

[39] P. Dayan and L. F. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. Cambridge, MA: MIT Press, 2001.

[40] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. New York: Addison-Wesley, 1991.

[41] E. Oja, "A simplified neuron model as a principal component analyzer," *J. Math Biol.*, vol. 15, pp. 267–273, 1982.

[42] T. Kohonen, *Self-Organizing Maps*, 3rd ed. Berlin, Germany: Springer-Verlag, 2001.

[43] E. Simoncelli and B. Olshausen, "Natural image statistics and neural representation," *Annu. Rev. Neurosci.*, vol. 24, pp. 1193–1216, 2001.

[44] E. Alhoniemi, J. Vesanto, J. Himberg, and J. Parhankangas, Som toolbox for Matlab 5 Helsinki Univ. of Technol., Finland, Tech. Rep. A57, 2000.

[45] M. D. Luciw, J. Weng, and S. Zeng, "Motor initiated expectation through top-down connections as abstract context in a physical world," in *Proc. 7th Int. Conf. Develop. Learn. (ICDL'08)*, Monterey, CA, 2008.

[46] P. Pajunen and J. Karhunen, "Least-squares methods for blind source separation based on nonlinear PCA," *Int. J. Neural Syst.*, vol. 8, no. 5–6, pp. 601–612, 1998.

[47] J. Karhunen, P. Pajunen, and E. Oja, "The nonlinear PCA criterion in blind source separation: Relations with other approaches," *Neurocomputing*, vol. 22, pp. 5–20, 1998.

[48] X. Giannakopoulos, J. Karhunen, and E. Oja, "Experimental comparison of neural ICA algorithms," in *Proc. Int. Conf. Artif. Neural Netw. (ICANN'98)*, Skövde, Sweden, 1998, pp. 651–656.

[49] J.-F. Cardoso, "Blind signal separation: Statistical principles," *Proc. IEEE*, vol. 86, no. 10, pp. 2009–2025, 1998.

[50] A. Hyvarinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. New York: Wiley, 2001.

[51] P. J. Huber, *Robust Statistics*. New York: Wiley, 1981.

Juyang Weng (S'85–M'88–SM'05–F'09) received the B.S. degree from Fudan University, China, and the M.S. and Ph.D. degrees from the University of Illinois at Urbana-Champaign, all in computer science.

He is now a Professor at the Department of Computer Science and Engineering, Michigan State University, East Lansing. He is also a Faculty Member of the Cognitive Science Program and the Neuroscience Program at Michigan State. Since the work of Cresceptron (ICCV 1993), he has expanded his research interests to biologically inspired systems, especially the autonomous development of a variety of mental capabilities by robots and animals, including perception, cognition, behaviors, motivation, and abstract reasoning skills. He has published more than 200 research articles on related subjects, including task muddiness, intelligence metrics, mental architectures, vision, audition, touch, attention, recognition, autonomous navigation, and other emergent behaviors. He is Editor-in-Chief of the *International Journal of Humanoid Robotics*. He was a Member of the Executive Board of the International Neural Network Society (2006–2008), Program Chairman of the NSF/DARPA-funded Workshop on Development and Learning 2000 (1st ICDL), Program Chairman of the Second ICDL (2002), Chairman of the Governing Board of the ICDL (2005–2007), and General Chairman of the Seventh ICDL (2008) and Eighth ICDL (2009). He and his coworkers developed SAIL and Dav robots as research platforms for autonomous development.

Dr. Weng is an Associate Editor of the IEEE TRANSACTIONS ON AUTONOMOUS MENTAL DEVELOPMENT. He was Chairman of the Autonomous Mental Development Technical Committee of the IEEE Computational Intelligence Society (2004–2005) and an Associate Editor of IEEE TRANSACTIONS ON PATTERN RECOGNITION AND MACHINE INTELLIGENCE and IEEE TRANSACTIONS ON IMAGE PROCESSING.

Matthew Luciw (S'06) received the B.S. and M.S. degrees from Michigan State University (MSU), East Lansing, in 2003 and 2006, respectively, both in computer science. He is currently pursuing the doctoral degree at MSU.

He is a member of the Embodied Intelligence Laboratory, MSU. His research involves the study of biologically inspired algorithms for autonomous development of mental capabilities—especially for visual attention and recognition.

Mr. Luciw is a student member of the Society for Neuroscience and the IEEE Computational Intelligence Society.