Optimal In-Place Learning and the Lobe Component Analysis

Juyang Weng, *Senior Member, IEEE* and Nan Zhang

***Abstract*— It is difficult to map many existing learning algorithms onto biological networks because the former require a separate learning network. The computational basis of biological cortical learning is still poorly understood. This paper rigorously introduces a concept called *in-place* learning. With in-place learning, every networked neuron *in-place* is responsible for the learning of its signal processing characteristics (e.g., efficacies of synapses) within its connected network environment. There is no need for a separate learning network. With this *in-place* hypothesis, consequently, each neuron does not have extra space to compute and store the second and higher order statistics (e.g., correlations) of its input fibers. This work first provides a classification of learning algorithms. Then, it shows that the two well-known *in-place* biological mechanisms, the Hebbian rule and lateral inhibition, are sufficient to develop orientation selective cells, similar to those found in V1, from inputs of natural images. Many other cells that have not been fully understood have emerged as well. The presented computational study of these two in-place learning mechanisms leads to a new concept — these cells correspond to what are called lobe components, which are high concentrations in the probability of the neuronal input space. Further analysis explains how every neuron can learn efficiently (i.e., near-optimal efficiency) by scheduling its plasticity while interacting with other connected neurons. A simple, in-place (Type-5) learning algorithm is presented. The experimental results showed that this simple biologically inspired algorithm is superior to some well-known state-of-the-art ICA algorithms, thanks to its near-optimal efficiency.**

I. INTRODUCTION

A classical study by Blakemore & Cooper 1970 [6] reported that edge detectors are far from being totally innate. If kittens were raised in an environment with only vertical edges, only neurons that respond to vertical or nearly vertical edges were found in the primary visual cortex. Recently, experimental studies have shown how the cortex develops through input-driven self-organization, in a dynamic equilibrium with internal and external inputs (e.g., Merzenich et al. 1983 [24], Gilbert & Wiesel 1992 [10]). Such dynamic development and adaption occurs from the prenatal stage and continues throughout infancy, childhood, and adulthood (e.g., Wang & Merzenich 1995 [25], Drafoi & Sur 2004 [9], Weng et al. [27]). Recently, Hosoya & Meister 2005 [12] reported that the spatio-temporal receptive fields and the response of retinal ganglion cells change after a few seconds in a new environment. The changes are adaptive in that the new receptive field improves predictive coding under the new image statistics.

Orientation selective cells in the cortical area V1 are well known [13], but the underlying learning mechanisms that govern their emergence (i.e., development) are still elusive. Furthermore, complex cells and other cells in V1 that do not exhibit a clear orientation are still poorly understood in terms of their functions and their underlying learning mechanisms. Understanding cortical filter development algorithms is important to address these open problems.

Juyang Weng and Nan Zhang are with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA (email: weng@cse.msu.edu).

The work for filter development reported here is fundamentally different from hand-designed filters such as Gabor filters, because the lobe component analysis (LCA) filters presented here are automatically generated (i.e., developed) from signals. Unlike hand-designed filters (e.g., Gabor filters) which wastefully tile the entire input space which is only sparsely sensed, LCA only develops filters to optimally cover all the subparts of input space that have been sensed. Because of this "spectrum finding" property and its use of biological mechanisms found in various cortices, LCA can be potentially used to develop filters for any cortical areas, e.g., any sensory modalities (vision or audition) for their early, intermediate, or later processing.

In order to understand what the biological cortex detects, how it represents internally and how it develops feature detectors, this paper rigorously proposes a classification of learning algorithms, and introduces the concept of in-place learning. Then, it introduces a biologically inspired new concept, called lobe components. Next, it presents a quasi-optimally efficient in-place learning algorithm that learns the lobe components. Finally, experimental results are presented in comparison with state-of-the-art ICA algorithms.

II. TYPES OF LEARNING ALGORITHMS

Consider a neuron which takes n inputs, $\mathbf{x} = (x_1, x_2, ..., x_n)$. The synaptic weight for x_i is w_i, $i = 1, 2, ..., n$, or write $\mathbf{w} = (w_1, w_2, ..., w_n)$. The response l of a neuron has been modeled by $l = g(\mathbf{w} \cdot \mathbf{x})$, where g is a monotone, nonlinear, sigmoid function. The learning (adaptation) of the neuron is characterized by the modification of g and $\mathbf{w}$ using input vectors $\mathbf{x}_t$, $t = 1, 2, ...$

To facilitate understanding of the nature of learning algorithms, we define five types of learning algorithms:

Type-1 batch: A batch learning algorithm L_1 computes g and $\mathbf{w}$ using a batch of vector inputs $B = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_b\}$, where b is the batch size.

$$(g, \mathbf{w}) = L_1(B), \tag{1}$$

where the argument B on the right is the input to the learner, L_1, and the right side is its output. This learning algorithm needs to store an entire batch of input vectors B before learning can take place. Since L_1 requires the additional storage B, L_1 must be realized by a separate network L_1

and thus, the learning of the (learning) network L_1 is an open problem.

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.

Type-3 incremental: Each input vector must be used immediately for updating the learner's memory (which must not store all the input vectors) and then the input must be discarded before receiving the next input. Type-3 is the extreme case of Type-2 in the sense that block size $b = 1$. A type-2 algorithm, such as Infomax, becomes a Type-3 algorithm by setting $b = 1$, but the performance will further suffer (see Infomax performance with $b = 1000$ in Sec. XI).

Type-4 covariance-free incremental: A Type-4 learning algorithm L_4 is a Type-3 algorithm, but furthermore, it is not allowed to compute the 2nd or higher order statistics of the input $\mathbf{x}$. In other words, the learner's memory $M^{(t)}$ cannot contain the second order (e.g., correlation or covariance) or higher order statistics of $\mathbf{x}$. The CCI PCA algorithm [28] is a covariance-free incremental learning algorithm for computing principal components as the weight vectors of neurons.

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 N itself. A term "local learning" used by some researchers, does not imply in-place.

For example, for a neuron N, its signal processor model has two parts, the synaptic weight $\mathbf{w}^{(t)}$ and the sigmodal function $g^{(t)}$, both updated up to t. A type-5 learning algorithm L_5 must update them using the previously updated weight $\mathbf{w}^{(t-1)}$ and the sigmoidal function $g^{(t-1)}$, using the current input $\mathbf{x}_t$ while keeping its maturity indicated by t:

$$(\mathbf{w}^{(t)}, g^{(t)}, t) = L_5(\mathbf{w}^{(t-1)}, g^{(t-1)}, t-1, \mathbf{x}_t). \qquad (2)$$

After the adaptation, the computation of the response is realized by the neuron N:

$$y_t = g^{(t)}(\mathbf{w}^{(t)} \cdot \mathbf{x}_t). \qquad (3)$$

An in-place learning algorithm must realize L_5 and the computation above by the same neuron N, for $t = 1, 2, ...$

III. Why In-place Learning

Principal components computed from natural images can be used as the weight vectors of neurons. Some of the principal components have a clear orientation, but their orientations are crude (e.g., exhibiting few directions) and their support is not localized (i.e., most synapses have significant magnitudes).

Higher order statistics (e.g., higher-order moments, kurtosis, etc) have been used in the Independent Component Analysis (ICA) [7], [21], [15]. The ICA algorithms search, from higher order statistics of inputs, a set of filters whose responses are statistically as independent as possible. However,

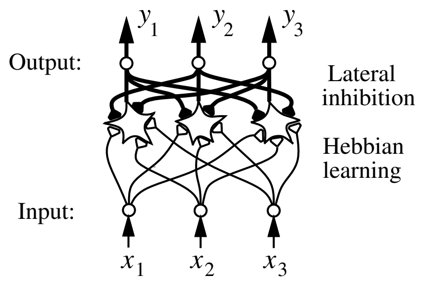


Fig. 1. Neurons with lateral inhibitions. Hollow triangles indicate excitatory connections, and the solid triangles indicate inhibitory connections.

the published ICA algorithms (e.g., the state-of-the-art ones [16], [4], [22], [19]) are Type-1 and Type-2 algorithms. By these algorithms, every learning neuron requires a dedicated, separate neural network to handle its learning. However, the existence and learning of this separate learning network has largely been left unaddressed. This overlooked obvious problem needs to be resolved not only for any neural learning algorithm to be biologically plausible, but more importantly, for understanding the tightly associated signal processing network.

The concept of in-place learning is based on the following hypothesis. Each neuron with n synaptic weights does not have extra space in itself to store correlation values and other higher-order moments of its input. For example, the covariance matrix of input $\mathbf{x}$ requires $(n+1)n/2$ additional storage units. A typical neuron in the brain has about $n = 1000$ synapses. The correlations of these n input fibres requires $(n+1)n/2 = 500,500$ storage units. A Purkinje cell typically has about $n = 150,000$ synapses. The correlation of its input vector requires as many as $(n+1)n/2 \approx 1.125 \times 10^{10}$ storage units. It is unlikely that a neuron has so many storage units within its cell body. For example, there is no evidence that the genes, residing in the cell nucleus, record and recall all correlations of cells input fibres.

Surprisingly, by forcing the learning algorithm to be in-place, we arrive at a deeper understanding of not only how they learn, but also their functions.

IV. Lateral Inhibition

Fig. 1 illustrates how neighboring neurons share the same vector of inputs $\mathbf{x}$ and how they are connected through lateral inhibition from the output of neighboring neurons.

According to the above model, the input to the i-th neuron in a layer consists of two parts, the excitatory part $\mathbf{x}$ and the inhibitory part $\mathbf{z}$:

$$z_i = g(\mathbf{w} \cdot \mathbf{x} - \mathbf{h} \cdot \mathbf{z}) \qquad (4)$$

where $\mathbf{w}$ consists of nonnegative weights for excitatory input $\mathbf{x}$, while $\mathbf{h}$ consists of nonnegative weights for nonnegative inhibitory input $\mathbf{z}$ but its i-th component is zero so that the right side does not require z_i. For biological plausibility, we assume that all the components in $\mathbf{x}$ and $\mathbf{z}$ are nonnegative.

The source of $\mathbf{z}$ is the response of neighboring neurons in the same layer.

The nature of inhibition is indicated by the minus sign before $\mathbf{h} \cdot \mathbf{z}$ in Eq. (4). In other words, the higher the response from neighboring neurons, the weaker the response is from this neuron. Vice versa, the higher the response from this neuron, the more effectively this neuron inhibits other neighboring neurons. Given an input vector $\mathbf{x}_t$, such a process of lateral inhibition reaches an equilibrium quickly.

Therefore, the net effect of lateral inhibition is the sparsity of the cortical response from each given input $\mathbf{x}$. Only relatively few winners fire. Many more neurons do not fire compared to the case without lateral inhibition.

In the computation, we simulate the effect of lateral inhibition to allow only top-k winners to fire, where k is a positive number.

V. The Hebbian Rule

The Hebbian rule [8], [11] is well known and it has many forms. The basic form of the Hebbian rule is expressed as

$$\tau_w \frac{d\mathbf{w}}{dt} = E\{v\mathbf{x}\}, \tag{5}$$

where τ_w is a time constant (or step size) that controls the rate at which the weight $\mathbf{w}$ changes [8], and the operator E denotes expectation (i.e., average) over observations $\mathbf{x}$ and $v = \mathbf{w} \cdot \mathbf{x}$.

> *In general, the Hebbian rule means that the learning rate of a neuron is closely related to the product of its current response and the current input, although the relation is typically nonlinear.*

Plug $v = \mathbf{w} \cdot \mathbf{x}$ into Eq. (5), we get

$$\tau_w \frac{d\mathbf{w}}{dt} = E\{(\mathbf{w} \cdot \mathbf{x})\mathbf{x}\}.$$

The above expression gives rise to the need to compute the correlation matrix of $\mathbf{x}$ in many learning algorithms, resulting in Type-1 and Type-2 algorithms in unsupervised learning.

In supervised learning, different forms of the Hebbian rule can be derived from different error functions $f(\mathbf{w})$, using the gradient of the objective function with respect to the weight vector $\mathbf{w}$. Consider the error function as the height of a natural terrain. Gradient at position $\mathbf{w}$ is the direction along which the objective function increases the quickest (i.e., the uphill direction). To reduce the objective function, the weight vector $\mathbf{w}$ should be modified along the opposite direction of the gradient. Methods of this type are simple and mathematically intuitive. However, gradient is a local characteristic of the error function. Although the current position $\mathbf{w}$ during a gradient-based search is a consequence of previous iterative search steps, the gradient does not consider the history of the past search. Because of this limitation, it does not provide any information about the optimal plasticity of $\mathbf{w}$. For example, it does not tell what the learning rate for $\mathbf{w}$ should be.

Although the method can be extended to supervised learning, this paper deals with unsupervised learning only.

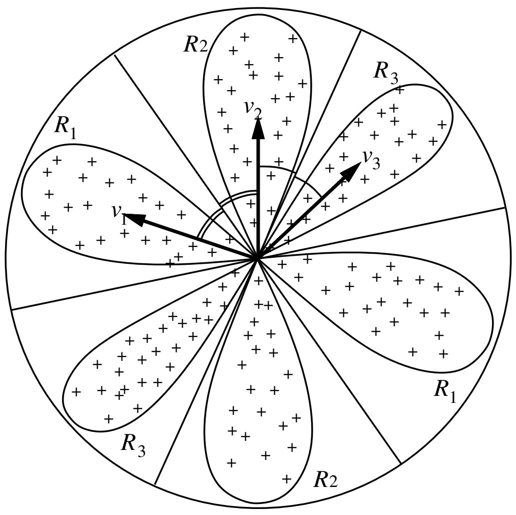


Fig. 2. The sample space of a zero-mean white random vector $\mathbf{y}$ in 2-D space can be illustrated by a circle. Each mark $+$ indicates a random sample of $\mathbf{y}$. The distribution is partitioned into $c = 3$ (symmetric) lobe regions R_i, $i = 1, 2, 3$, where R_i is represented by the lobe component (vector) $\mathbf{v}_i$.

VI. Lobe Components

Atick and coworkers [1], [2] proposed that early sensory processing decorrelates inputs. Weng et al. [28] proposed an in-place algorithm that develops a network that whitens the input. A whitened output vector has a unit variance (i.e., "energy") for every line and all zero inter-line correlations. Therefore, we can assume that prior processing has been done so that its output vector $\mathbf{y}$ is white. From now on, we assume that the input is $\mathbf{y}$. In the visual pathway, the early cortical processing does not totally whiten the signals, but they are weakly correlated. The sample space of a k-dimensional white (i.e., components have unit variance and they are pairwise uncorrelated) input random vector $\mathbf{y}$ can be illustrated by a k-dimensional hypersphere, as shown in Fig. 2.

A concentration of the probability density of the input space is called a lobe, which may have its own finer 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. If we assume that $\mathbf{x}$ and $-\mathbf{x}$ are equally likely (e.g., a patch of skin will have the equal chance of feeling the onset and the offset of a press), the distribution is then symmetric about the origin.[1]

Given a limited cortical resource, c cells fully connected to input $\mathbf{y}$, the developing cells divide the sample space $\mathcal{Y}$ into c mutually nonoverlapping regions, called *lobe regions*:

$$\mathcal{Y} = R_1 \cup R_2 \cup ... \cup R_c, \tag{6}$$

(where $\cup$ denotes the union of two spaces) as illustrated in Fig. 2. Each region R_i is represented by a single unit feature vector $\mathbf{v}_i$, called the *lobe component*. Given an input

[1]But this is not necessarily true in general and, consequently, nonsymmetric lobes can be defined.

$\mathbf{y}$, many cells, not only $\mathbf{v}_i$, will respond whose pattern forms a population representation of $\mathbf{y}$. This partition idea is similar to Kohonen's Self Organizing Maps (SOM) [20], or more generally, vector quantization.

The next issue is how neurons compute the lobe regions without a supervisor (i.e., by in-place mechanisms). Suppose that a unit vector (neuron) $\mathbf{v}_i$ represents a lobe region R_i. If $\mathbf{y}$ belongs to R_i, $\mathbf{y}$ can be approximated by $\mathbf{v}_i$ as the projection onto $\mathbf{v}_i$: $\mathbf{y} \approx \hat{\mathbf{y}} = (\mathbf{y} \cdot \mathbf{v}_i)\mathbf{v}_i$. Suppose the neuron $\mathbf{v}_i$ minimizes the mean square error $E\|\mathbf{y} - \hat{\mathbf{y}}\|^2$ of this representation when $\mathbf{y}$ belongs to R_i.

From the theory of principal component analysis (PCA) (e.g., see [17]), we know that the best solution of column vector $\mathbf{v}_i$ is the principal component of the conditional covariance matrix $\Sigma_{y,i}$, conditioned on $\mathbf{y}$ belonging to R_i. That is $\mathbf{v}_i$ satisfies $\lambda_{i,1}\mathbf{v}_i = \Sigma_{y,i}\mathbf{v}_i$.

Replacing $\Sigma_{y,i}$ by the estimated sample covariance matrix of column vector y, we have

$$\lambda_{i,1}\mathbf{v}_i \approx \frac{1}{n}\sum_{t=1}^{n}\mathbf{y}(t)\mathbf{y}(t)^\top\mathbf{v}_i = \frac{1}{n}\sum_{t=1}^{n}(\mathbf{y}(t)\cdot\mathbf{v}_i)\mathbf{y}(t). \quad (7)$$

We can see that the best lobe component vector $\mathbf{v}_i$, scaled by "energy estimate" eigenvalue $\lambda_{i,1}$, can be estimated by the *average* of the input vector $\mathbf{y}(t)$ weighted by the linearized (without g) response $\mathbf{y}(t) \cdot \mathbf{v}_i$ whenever $\mathbf{y}(t)$ 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 below.

In order to minimize the chance of false extrema and to optimize the speed to reach the desired feature solution, we use a concept in statistics called *efficiency*. Suppose that there are two estimators Γ_1 and Γ_2, for vector parameter (i.e., synapses or feature vector) $\theta = (\theta_1, ..., \theta_k)$, which are based on the same set of observations $S = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$. 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. Given the same observations, among all possible estimators, the optimally efficient estimator has the smallest possible error.

For in-place development, each neuron does not have extra space to store all the training samples $\mathbf{y}(t)$. Instead, it uses its physiological mechanisms to update synapses incrementally. If the i-th neuron $\mathbf{v}_i(t-1)$ at time $t-1$ has already been computed using previous $t-1$ inputs $\mathbf{y}(1), \mathbf{y}(2), ..., \mathbf{y}(t-1)$, the neuron can be updated into $\mathbf{v}_i(t)$ using the current sample defined from $\mathbf{y}(t)$ as:

$$\mathbf{x}_t = \frac{\mathbf{y}(t)\cdot\mathbf{v}_i(t-1)}{\|\mathbf{v}_i(t-1)\|}\mathbf{y}(t). \quad (8)$$

Then Eq. (7) states that the lobe component vector is estimated by the average:

$$\lambda_{i,1}\mathbf{v}_i \approx \frac{1}{n}\sum_{t=1}^{n}\mathbf{x}_t. \quad (9)$$

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 Theorem 4.1, p. 429-430 of Lehmann [23], which states that under some regularity conditions satisfied by most distributions (such as Gaussian and exponential distributions), the maximum likelihood estimator (MLE) $\hat{\theta}$ of the parameter vector θ 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:

$$(\hat{\theta} - \theta) \xrightarrow{p} N\{0, I(\theta)^{-1}/n\} \quad (10)$$

in which the Fisher information matrix $I(\theta)$ is the covariance matrix of the score vector $\left\{\frac{\partial f(\mathbf{x},\theta)}{\partial\theta_1}, ..., \frac{\partial f(\mathbf{x},\theta)}{\partial\theta_k}\right\}$, and $f(\mathbf{x}, \theta)$ is the probability density of random vector $\mathbf{x}$ if the true parameter value is θ (see, e.g., Lehmann [23], p. 428). The matrix $I(\theta)^{-1}$ is called information bound since under some regularity constraints, any unbiased estimator $\widetilde{\theta}$ of the parameter vector θ satisfies $\mathrm{cov}(\widetilde{\theta} - \theta) \geq I(\theta)^{-1}/n$ (see, e.g., Lehmann [23], p.428 or Weng et al. [26], pp. 203-204)[2].

Since in many cases the MLE of the population mean is the sample mean, we estimate the mean θ of vector $\mathbf{x}$ by the sample mean. Thus, we estimate an independent vector $\lambda_1\mathbf{v}_i$ by the sample mean in Eq. (9), where $\mathbf{x}_t$ is a "random observation".

VII. For Nonstationary Processes

Although the conditions on the distribution of $\mathbf{x}_t$ are mild, $\mathbf{x}_t$ depends on the currently estimated $\mathbf{v}_i$. That is, the observations of $\mathbf{x}_t$ are from a nonstationary process. In general, the environments of a developing system change over time. Therefore, the sensory input process is a nonstationary process too. We use the amnesic mean technique below which gradually "forgets" old "observations" (which use bad $\mathbf{x}_t$ when t is small) while keeping the estimator quasi-optimally efficient.

The mean in Eq. (9) is a batch method. For incremental estimation, we use what is called an amnesic mean [28].

$$\bar{x}^{(t)} = \frac{t-1-\mu(t)}{t}\bar{x}^{(t-1)} + \frac{1+\mu(t)}{t}x_t \quad (11)$$

where $\mu(t)$ is the amnesic function depending on t. If $\mu \equiv 0$, the above gives the straight incremental mean. The way to compute a mean incrementally is not new but the way to use the amnesic function of n is new for computing a mean incrementally.

$$\mu(t) = \begin{cases} 0 & \text{if } t \leq t_1, \\ c(t-t_1)/(t_2-t_1) & \text{if } t_1 < t \leq t_2, \\ c + (t-t_2)/r & \text{if } t_2 < t, \end{cases} \quad (12)$$

in which, e.g., $c = 2, r = 10000$. As can be seen above, $\mu(t)$ has three intervals. When t is small, straight incremental average is computed. Then, $\mu(t)$ changes from 0 to 2 linearly in the second interval. Finally, t enters the third section where

[2]For two real symmetric matrices A and B of the same size, $A \geq B$ means that $A - B$ is nonnegative definite.

$\mu(t)$ increases at a rate of about $1/r$, meaning the second weight $(1+\mu(t))/t$ in Eq. (11) approaches a constant $1/r$, to slowly trace the slowly changing distribution.

VIII. Efficiency of the Amnesic Mean

We consider whether the amnesic mean is an unbiased estimator. From the recursive definition of the amnesic mean in Eq. (11), we can see that the amnesic mean $\bar{x}(n)$ is a weighted sum of the involved data x_t:

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

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

$$w_t(n) = \frac{1+\mu(t)}{t} \prod_{j=t+1}^{n} \frac{j-1-\mu(j)}{j}. \qquad (13)$$

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

$$\sum_{t=1}^{n} w_t(n) = 1. \qquad (14)$$

(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, the amnesic mean is an unbiased estimator of Ex:

$$\begin{aligned} E\bar{x}(n) &= E\sum_{t=1}^{n} w_t(n) x_t = \sum_{t=1}^{n} w_t(n) E x_t \\ &= \sum_{t=1}^{n} w_t(n) Ex = Ex. \end{aligned}$$

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

$$\text{cov}(\bar{x}(n)) = \left(\sum_{t=1}^{n} w_t^2(n) \right) \text{cov}(x) = c(n)\text{cov}(x), \qquad (15)$$

where we defined the *error coefficient*:

$$c(n) = \sum_{t=1}^{n} w_t^2(n).$$

When $\mu(t) = 0$ for all n, the error coefficient becomes $c(n) = 1/n$ and Eq. (15) returns to the expected square error of the regular sample mean:

$$\text{cov}(\bar{x}(n)) = \frac{1}{n}\text{cov}(x). \qquad (16)$$

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. 3 shows the error coefficient $c(n)$ for three different amnesic functions $\mu(t) \equiv 2$, $\mu(t) \equiv 1$ and $\mu(t) \equiv 0$. The smaller the error

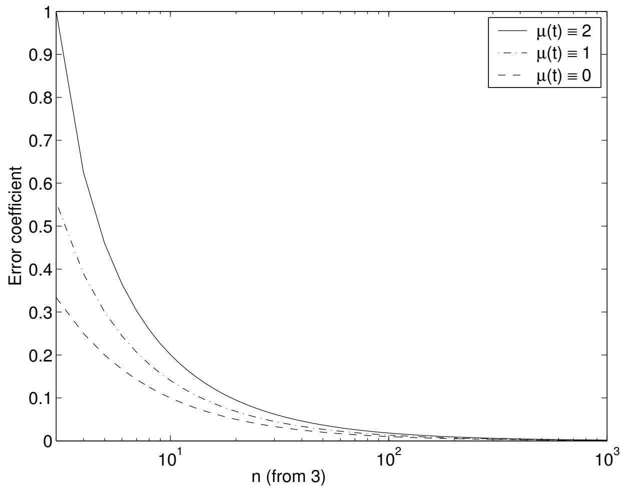


Fig. 3. The error coefficients $c(n)$ for amnesic means with different amnesic functions $\mu(t)$.

coefficient, the smaller the expected square error, but also there is less capability to adapt to a changing distribution. The three cases shown in Fig. 3 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. 3 we can see that a constant positive $\mu(t)$ is not best when t is small. The multi-sectional amnesic function $\mu(t)$ in Eq. (12) performs straight average for small t to reduce the error coefficient for earlier estimates, and when t is very large, the amnesic function changes with t to track the slowly changing distribution. Therefore, the multi-sectional amnesic function $\mu(t)$ is more suited for practical signals with unknown nonstationary statistics. 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 reality.

Therefore, each neuron can use the amnesic mean technique (i.e., time-varying plasticity), which gradually "forgets" old "observations" (which use "bad" x_t when t is small) while keeping the estimator reasonably efficient (quasi-optimally efficient) for a nonstationary process, as shown in Eq. 11. The amnesic function $\mu(t)$ represents the "critical window" of neuronal plasticity imbedded in the genetic and physiological mechanism of a neuron.

IX. In-place Learning Algorithm

We model the development (adaptation) of an area of cortical cells (e.g., a cortical column) connected by a common input column vector $\mathbf{y}$ by the following Candid Covariance-free Incremental Lobe Component Analysis (CCI LCA) (Type-5) algorithm, which incrementally updates c such cells (neurons) represented by the column vectors $\mathbf{v}_1^{(t)}, \mathbf{v}_2^{(t)}, ..., \mathbf{v}_c^{(t)}$ from input samples $\mathbf{y}(1), \mathbf{y}(2), ...$ of dimension k without computing the $k \times k$ covariance matrix of $\mathbf{y}$. The length of the estimated $\mathbf{v}_i$, its eigenvalue, is the variance of projections of the vectors $\mathbf{y}(t)$ onto $\mathbf{v}_i$. The quasi-optimally efficient, in-place CCI LCA algorithm is as follows:

1) Sequentially initialize c cells using first c observations: $\mathbf{v}_t^{(c)} = \mathbf{y}(t)$ and set cell-update age $n(t) = 1$, for $t = 1, 2, ..., c$.
2) For $t = c+1, c+2, ...$, do
   a) Compute the (linear) response for all neurons: For all i with $1 \leq i \leq c$:
   $$z_i = \frac{\mathbf{y}(t) \cdot \mathbf{v}_i^{(t-1)}}{\|\mathbf{v}_i^{(t-1)}\|}.$$
   Note that we have also used a nonlinear sigmoidal function but the result is similar.
   b) Simulating lateral inhibition, decide the winner: $j = \arg\max_{1 \leq i \leq c}\{|z_i|\}$, using $|z_i|$ as the belongingness of $\mathbf{y}(t)$ to R_i.
   c) Update only the winner neuron v_j using its temporally scheduled plasticity:
   $$\mathbf{v}_j^{(t)} = w_1 \mathbf{v}_j^{(t-1)} + w_2 z_j \mathbf{y}(t),$$
   where the scheduled plasticity is determined by its two age-dependent weights:
   $$w_1 = \frac{n(j) - 1 - \mu(n(j))}{n(j)}, w_2 = \frac{1 + \mu(n(j))}{n(j)},$$
   with $w_1 + w_2 \equiv 1$. Update the number of hits (cell age) $n(j)$ only for the winner: $n(j) \leftarrow n(j) + 1$.
   d) All other neurons keep their ages and weight unchanged: For all $1 \leq i \leq c$, $i \neq j$, $\mathbf{v}_i^{(t)} = \mathbf{v}_i^{(t-1)}$.

The neuron winning mechanism corresponds to the well known mechanism called lateral inhibition (see, e.g., Kandel et al. [18] p. 4623). The winner updating rule is a computer simulation of the Hebbian rule (see, e.g., Kandel et al. [18] p.1262). Assuming the plasticity scheduling by w_1 and w_2 are realized by the genetic and physiologic mechanisms of the cell, this algorithm is in-place.

Instead of updating only the top-1 neuron, using biologically more accurate "soft winners" where multiple top (e.g., top-k) winners update, we have obtained similar results.

This in-place algorithm is simple. Given each k-dimensional input $\mathbf{y}$, the time complexity for updating c cells and computing all the responses for each $\mathbf{y}$ is merely $O(ck)$. With t inputs, the total time complexity is $O(ckt)$. The space complexity is only $O(ck)$.

Further, these space and time complexities are also the **lowest possible** ones, since c cells need to be computed and each vector has k synapses.

From the discussed meaning of lobe components, we can infer that LCA is ICA for super-Gaussians.[3] Natural images are mostly super-Gaussian.

[3]Super-Gaussian has a positive kurtosis (see, e.g., [5], p. 388).

X. Topographic Maps

In the above CCI LCA algorithm, we assume that neurons in the same layer are mutually related only through their synaptic weights. However, biological neurons are located physically along a two-dimensional sheet (cortical sheet). An important concept of the biological cortex, which is also of great importance to artificial networks, is self-organized *modularity*: neurons that represent similar concepts (e.g., human faces) are located nearby within the cortex. The modularity property enables certain conceptual generalizations through nearby connections since nearby neurons represent conceptually similar concepts.

In the above CCI LCA algorithm, neurons are related only by their synaptic weights. Each neuron can be physically located in any place. Next, we assume additionally that the neurons reside in a 2-D surface (e.g., the 2-D cortex), to enable self-organization within the 2-D surface resulting in what is called a topographic map. In a topographic map, neurons that represent similar features are near. The CCI LCA algorithm can be slightly modified so that not only the winner updates its synaptic weights, but also its 3×3 neighbors, which results in the topographic LCA map to be shown in the experimental section. Although SOM and its published variants does develop feature maps, they do not have the concept of lobe components in high-dimensional distribution, quasi-invariance and quasi-optimal efficiency.

XI. Experimental Results

We first present comparisons of the CCI LCA algorithm with other ICA algorithms for a high-dimensional ICA problem, which shows the power of the statistical near-optimal efficiency of the CCI LCA algorithm. Then, we show what kinds of features the CCI LCA algorithm developed from natural images.

A. Comparison with other ICA algorithms

The 100-D data was synthetically generated from an identically independently distributed Laplacian random number generator. The mixing matrix was chosen randomly and was non-degenerate. This is an ICA problem with super-Gaussian distribution. As explained, LCA gives independent components for super-Gaussian when c is equal to the number of independent components.

We selected two state-of-the-art ICA algorithms: Type-2 Extended Bell-Sejnowski (extended Infomax) [22] with block size 1000 and the Type-1 FastICA [16]. Although this is *not a fair* comparison, since normally our Type-5 algorithm (like CCI LCA) should not be expected to out-perform a Type-1 or Type-2 algorithm, we compared them anyway to understand the power.

In order to show more detailed aspects of CCI LCA, three variations have been tested. "LCA with fixed m" (m stands for the amnesic function $\mu(t)$) and "LCA with dynamic m" use a fixed and time-varying $\mu(t)$, respectively. "LCA eliminating cells" dynamically deletes cells whose hitting rate is smaller than 3/4 of the average hitting rate. As shown

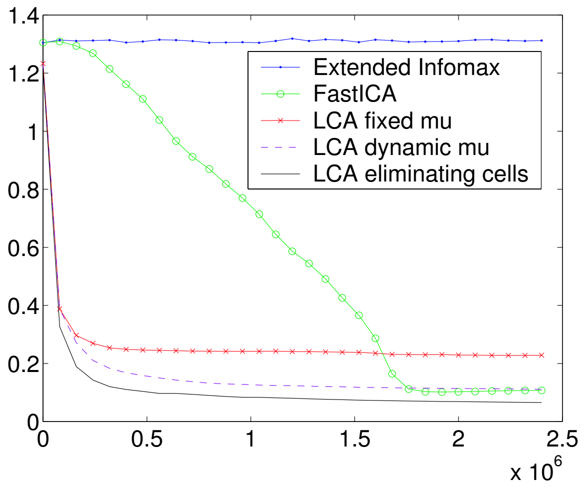


Fig. 4. Comparison among (Type-2) Extended Infomax, (Type-1) FastICA and three variants of the proposed Type-5 CCI LCA algorithms from 100-D identically independently distributed Laplacian source.

in Fig. 4, all of the three Type-5 LCA algorithms converged very quickly, faster than even the Type-1 FastICA. The Type-2 Extended Infomax algorithm needs many more samples, therefore, it did not get close to the desired results.

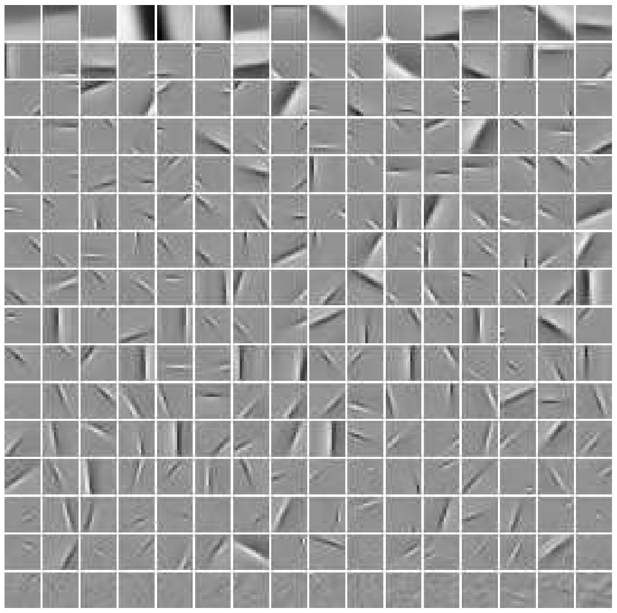
It was somewhat **surprising** that the new Type-5 CCI LCA algorithm, operating under the most restrictive condition, outperforms the state-of-the-art Type-2 and Type-1 ICA algorithms by a remarkably wide margin for high-dimensional ICA problems. Conceptually, this is mainly due to the biologically motivated conceptualization of lobe components for estimation of distribution and the avoidance of the much restrictive requirement of independence. Further, mathematically, the estimation of lobe components avoids directly solving computationally expensive eigen-system equations and has near-optimal statistical efficiency.

B. Features and receptive fields from natural images

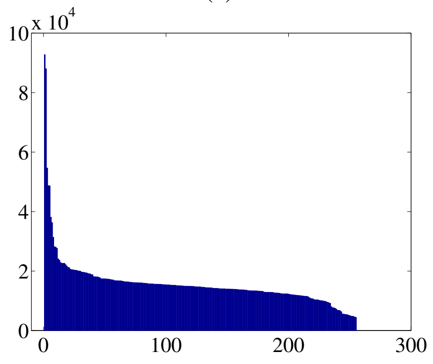
In the computer simulation, the CCI LCA algorithm received 500,000 of 16×16-pixel image patches that were randomly taken from thirteen natural images (available from http://www.cis.hut.fi/projects/ica/imageica/). The input is whitened (as required by ICA) to generate whitened input vector $\mathbf{y}$ to the CCI LCA algorithm. The developed feature detectors, combining whitening and lobe components, are shown in Fig. 5.

An important result is apparent when we examined the synaptic weights of each neuron shown as image patches in Fig. 5(a). They clearly showed localized patterns, many of which resemble the orientation selection cells reported by Hubel & Wiesel and others. This occurred while the CCI LCA developmental program did not incorporate any specific mechanism for orientation selection. This result prompted us to make a **hypothesis: A major principle of cortical cells is to approximate statistical distribution.**

Another important result is that the CCI LCA algorithm not only learned synaptic weights to develop orientation



(a)



(b)

Fig. 5. Lobe components from natural images. (a) The lobe components developed by in-place CCI LCA from natural images, ordered by the number of hits in decreasing order. (b) The numbers of hits of the corresponding lobe components in (a).

selective cells but also many other cells that **cannot be explained** by orientation sensitivity. This shows that the algorithm has a potential to be applied to later cortical processing to deal with more complex features.

Further, it also **learned receptive fields** automatically: the receptive field of each neuron, one that corresponds to non-zero weights, is automatically narrowed (localized) from initially over-grown synaptic connections. This is consistent to a well known developmental process in neuroscience called synapse elimination [3].

Fig. 5(b) shows how many times each lobe component was the "winner." Most components have roughly a similar rate of hits, except relatively few of the leftmost (top) ones and rightmost (tailing) ones.

The results correspond to in a deep biological implication: The two well-known in-place neural mechanisms develop cortical cells that represent high concentrations of input probability density, which should be the lobe components

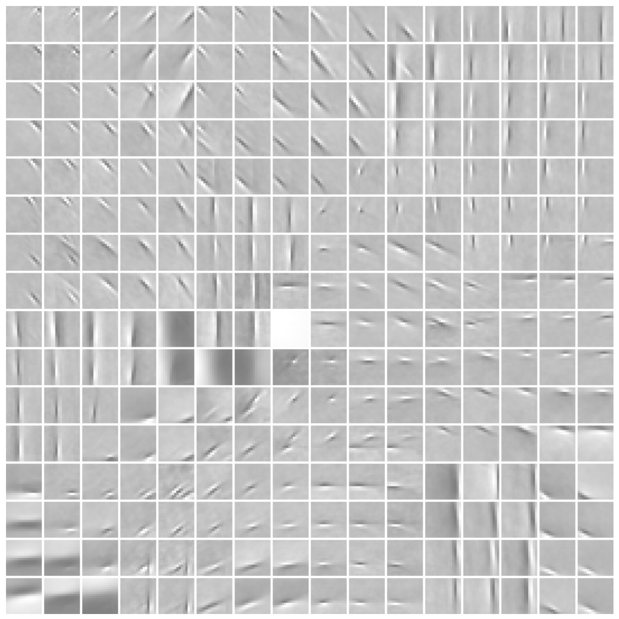


Fig. 6. Topographic CCI LCA map developed from natural images, which shows the self-organization of feature detectors along the 2-D cortical surface.

discovered here.

C. Topographic maps

The use of 3×3 updates generated the topographic map in the previous experiment shown in Fig. 6. As shown, the feature detectors that are developed are similar, but not exactly the same as those in Fig. 5(a). However, nearby cells detect similar features, resulting in a map that resembles the “spinwheel” structure found in V1 [14], [9].

XII. Conclusions

Considering the wide-spread presence of the two in-place mechanisms (Hebbian learning and lateral inhibition) in early and later cortical regions, the implications of the result here have a potentially high impact. It suggests an array of future studies for cortical regions based on the in-place learning framework: Do the cerebral cortices develop to represent the statistical distribution of their input signals as the result here suggests? Is it really true that cortical cells detect independent components as some models suggested? The result here states no. From the engineering point of view, it is of great importance that the two well-known biological in-place mechanisms, Hebbian learning and lateral inhibition, enable cells to develop quasi-optimally efficient lobe components, as resource-limited internal representation of concentration points in high dimensional probability density of sensory inputs. The surprisingly fast convergence of CCI LCA compared to the state-of-the-art ICA algorithms demonstrated the high power of this quasi-optimality.

Acknowledgment

The authors thank Shuqing Zeng for plotting Fig. 3.

References

[1] J. J. Atick and A. N. Redlich. Towards a theory of early visual processing. *Neural Computation*, 2:308–320, 1990.

[2] J. J. Atick and A. N. Redlich. Convergent algorithm for sensory receptive field development. *Neural Computation*, 5:45–60, 1993.

[3] R. J. Balice-Gordon and J. W. Lichtman. Long-term synapse loss induced by focal blockade of postsynaptic receptors. *Nature*, 372:519–524, 1994.

[4] A. J. Bell and T. J. Sejnowski. The ‘independent components’ of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

[5] P. J. Bickel and K. A. Doksum. *Mathematical Statistics. Basic Ideas and Selected Topics*. Holden-Day, Inc., San Francisco, 1976.

[6] C. Blakemore and G. F. Cooper. Development of the brain depends on the visual environment. *Nature*, 228:477–478, Oct. 1970.

[7] P. Comon. Independent component analysis, A new concept? *Signal Processing*, 36:287–314, 1994.

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

[9] V. Dragoi and M. Sur. Plasticity of orientation processing in adult visual cortex. In L. M. Chalupa and J. S. Werner, editors, *Visual Neurosciences*, pages 1654–1664. MIT Press, Cambridge, Massachusetts, 2004.

[10] C. D. Gilbert and T. N. Wiesel. Receptive field dynamics in adult primary visual cortex. *Nature*, 356:150–152, 1992.

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

[12] T. Hosoya, S. A. Baccus, and M. Meister. Dynamic predictive coding by the retina. *Nature*, 436:71–77, 2005.

[13] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex. *Journal of Physiology*, 160(1):107–155, 1962.

[14] M. Hubener, D. Shoham, A. Grinvald, and T. Bonhoeffer. Spatial relationships among three columnar systems in cat area 17. *J. of Neuroscience*, 17:9270–9284, 1997.

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

[16] A. Hyvarinen and E. Oja. A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9(7):1483–1492, 1997.

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

[18] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, editors. *Principles of Neural Science*. McGraw-Hill, New York, 4th edition, 2000.

[19] J. Karhunen and P. Pajunen. Blind source separation using least-squares type adaptive algorithms. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pages 3048–3051, Munich, Germany, 1997.

[20] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 3rd edition, 2001.

[21] T. W. Lee. *Independent Component Analysis: Theory and Applications*. Kluwer, Boston, Massachusetts, 1998.

[22] 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 Computation*, 11(2):417–441, 1999.

[23] E. L. Lehmann. *Theory of Point Estimation*. John Wiley and Sons, Inc., New York, 1983.

[24] M. M. Merzenich, J. H. Kaas, J. T. Wall, M. Sur, R. J. Nelson, and D. J. Felleman. Progression of change following median nerve section in the cortical representation of the hand in areas 3b and 1 in adult owl and squirrel monkeys. *Neuroscience*, 10(3):639–665, 1983.

[25] X. Wang, M. M. Merzenich, K. Sameshima, and W. M. Jenkins. Remodeling of hand representation in adult cortex determined by timing of tactile stimulation. *Nature*, 378(2):13–14, 1995.

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

[27] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. Autonomous mental development by robots and animals. *Science*, 291(5504):599–600, 2001.

[28] J. Weng, Y. Zhang, and W. Hwang. Candid covariance-free incremental principal component analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(8):1034–1040, 2003.