A Multiplicative Up-Propagation Algorithm

Jong-Hoon Ahn JONGHUN@POSTECH.AC.KR
Department of Physics, POSTECH, Korea

Seungjin Choi SEUNGJIN@POSTECH.AC.KR
Department of Computer Science, POSTECH, Korea

Jong-Hoon Oh JHOH@POSTECH.AC.KR
Department of Physics, POSTECH, Korea

Abstract

We present a generalization of the nonnegative matrix factorization (NMF), where a multilayer generative network with nonnegative weights is used to approximate the observed nonnegative data. The multilayer generative network with nonnegativity constraints, is learned by a multiplicative up-propagation algorithm, where the weights in each layer are updated in a multiplicative fashion while the mismatch ratio is propagated from the bottom to the top layer. The monotonic convergence of the multiplicative up-propagation algorithm is shown. In contrast to NMF, the multiplicative up-propagation is an algorithm that can learn hierarchical representations, where complex higher-level representations are defined in terms of less complex lower-level representations. The interesting behavior of our algorithm is demonstrated with face image data.

1. Introduction

Nonnegative matrix factorization (NMF) which was published in Nature a few years ago (Lee & Seung, 1999), drew extensive attraction in machine learning and pattern recognition communities. It is one of the efforts to realize a parts-based representation which is a way of understanding the perception in the brain and certain computational theories rely on such a representation. We also encounter into a variety of practical applications where parts-based representations of nonnegative data are useful. These include face images for computer vision (Li et al., 2001), medical image analysis (Lee et al., 2001), spectrograms of sound data (Cho et al., 2003), text data, and so on (Saul et al., 2003).

NMF is a simple linear coding algorithm for a single-layer generative network with nonnegativity constraints. Although it is an efficient linear coding method for nonnegative data, the linear single-layer generative network leads to several limitations in a certain general problem such as:

- when it learns data which lie on or near a nonlinear manifold;
- when it learns syntactic relationships of the given data;
- when it learns hierarchically generated data.

NMF is obviously not suitable for such cases. For instance, when NMF is applied to some sequences of images which contain facial expressions, it does not show a parts-based representation clearly. For images of objects viewed from extremely different viewpoints, it requires much more data and additional degree of freedom. NMF assumes that the hidden variables are nonnegative, but makes no further assumptions about their statistical dependency. Accordingly, NMF cannot learn anything about the syntactic relationships between parts. For instance, given two basis vectors of eyes and eyes+eyebrows computed by NMF, we cannot know the similarity or dependency between them. Let us consider a problem of finding a set of initial basis vectors, given a set of hierarchically generated data, which is described as follows: 1. Assume that we have a set of p_1 nonnegative unit vectors; 2. Try to produce p_2 composites by mixing and transforming them; 3. Try to repeat the step 2 and produce another

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.

p_{i+1} composites from the ith composites; 4. When we see only the final composites, could we find the initial p_1 nonnegative vectors;

Any modifications of the NMF algorithm in a single-layer network do not yield a solution to this problem. Therefore, we propose a generalization of NMF, where we employ a multilayer generative network with nonnegativity constraints, which still preserve a useful property of NMF such as parts-based representations. A generalization of NMF with employing a multilayer generative network, is an nonlinear extension since squashing functions should be incorporated with the multilayer network.

A conventional multilayer perceptron (MLP) that belong to a recognition model, can be efficiently trained by an error back-propagation algorithm. It was also shown that the multilayer generative network could be trained by an error up-propagation algorithm (Oh & Seung, 1997). A main difference between these two networks is that the former is a supervised network and the latter is an unsupervised network, whereas both of them are trained by gradient-based additive algorithms. The multilayer network that we consider in this paper, is a generative network with all of synaptic weights being nonnegative. Therefore, the error up-propagation algorithm (Oh & Seung, 1997) cannot be directly applied for learning our multilayer generative network. However, this suggests us to use an idea of error up-propagation. Motivated from an idea of multiplicative update used in NMF, we develop a new learning algorithm, *multiplicative up-propagation* for training the multilayer generative network with nonnegativity constraints. In our multiplicative up-propagation algorithm, synaptic weights for each layer are trained in a multiplicative fashion (in order to preserve nonnegativity constraints), while the mismatch (or the error) ratio is propagated from the bottom to the top layer.

Two exemplary NMF algorithms result from objective functions based on I-divergence and squared error. We show that NMF could be a general optimizing rule for various objective functions. As in gradient descent rules used in back-prop and up-prop algorithms for a given error function, our proposed multiplicative algorithm can also be used for general objective functions with nonnegativity constraints.

The rest of this paper is organized as follows. Next section briefly reviews the original up-propagation algorithm (Oh & Seung, 1997) and NMF (Lee & Seung, 1999). In Sec. 3, we illustrate an detailed derivation of the proposed multiplicative up-propagation algorithm. Its monotonic convergence is also shown in Sec. 3. Then the algorithm is applied to face image data and its interesting behavior is shown in Sec. 4. Finally conclusion is drawn in Sec. 5.

2. Previous Work

2.1. Up-propagation

The up-propagation is an algorithm for inverting and learning a multi-layer generative model, which generalizes PCA and its variants such as conic coding (Oh & Seung, 1997). The generative model is a network of L layers of neurons with input layer at bottom, as depicted in fig. 2. The matrices $\boldsymbol{H}^{(l)}$, $l = 1, \cdots, L$, are the activations of the layers. The pattern $\boldsymbol{H}^{(0)}$ is generated from the hidden variables $\boldsymbol{H}^{(L)}$ by a top-down pass through the network,

$$\boldsymbol{H}^{(l-1)} = \mathsf{g}(\boldsymbol{W}^{(l)}\boldsymbol{H}^{(l)}), \quad l = L, \cdots, 1.$$

The nonlinear function g acts on matrices component by component. The matrix $\boldsymbol{W}^{(l)}$ contains the synaptic connections from the neurons in layer l to the neurons in layer $l-1$. A bias term $\boldsymbol{B}^{(l)}$ can be added to the argument of g, but is omitted here.

Given a sensory input $\boldsymbol{V}$, the top-down generative model can be inverted by finding hidden variables $\boldsymbol{H}^{(L)}$ that generate a pattern $\boldsymbol{H}^{(0)}$ matching $\boldsymbol{V}$. If some of the hidden variables represent the identity of the pattern, the inversion operation is called recognition. Alternatively, the hidden variables may just be a more compact representation of the pattern, in which case the operation is called analysis or encoding. The inversion is done iteratively, as described below.

In the following, the operator $\odot$ denotes the Hadamard product (elementwise multiplication), that is, $\boldsymbol{Z} = \boldsymbol{X} \odot \boldsymbol{Y}$ means $z_{ij} = x_{ij}y_{ij}$. The bottom-up pass starts with the mismatch between the sensory data $\boldsymbol{V}$ and the generated pattern $\boldsymbol{H}^{(0)}$,

$$\boldsymbol{\Delta}^{(1)} = \mathsf{g}'(\boldsymbol{W}^{(1)}\boldsymbol{H}^{(1)}) \odot (\boldsymbol{V} - \boldsymbol{H}^{(0)}), \tag{1}$$

which is propagated upwards by

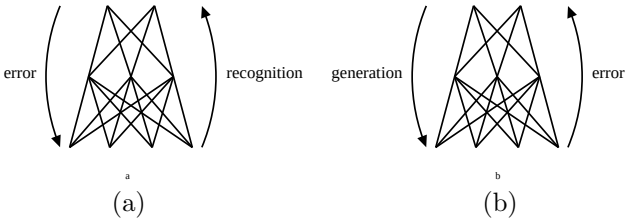
$$\boldsymbol{\Delta}^{(l+1)} = \mathsf{g}'(\boldsymbol{W}^{(l+1)}\boldsymbol{H}^{(l+1)}) \odot ((\boldsymbol{W}^{(l)})^T\boldsymbol{\Delta}^{(l)}). \tag{2}$$

When the error signal reaches the top of the network, it is used to update the hidden variables $\boldsymbol{H}^{(L)}$,

$$\Delta\boldsymbol{H}^{(L)} = -\eta(\boldsymbol{W}^{(L)})^T\boldsymbol{\Delta}^{(L)}. \tag{3}$$

This update closes the negative feedback loop. Then a new pattern $\boldsymbol{H}^{(0)}$ is generated by a top-down pass, and the process starts over again.

This iterative inversion process performs gradient descent on the cost function $\frac{1}{2} \parallel \boldsymbol{V} - \boldsymbol{H}^{(0)} \parallel^2$, subject



(a) (b)

Figure 1. Bottom-up and top-down processing in neural network: (a) back-prop network; (b) up-prop network

to the constraints. This can be proved using the chain rule, as in the traditional derivation of the back-prop algorithm. Another method of proof is to add the equations as constraints, using Lagrange multipliers,

$$\frac{1}{2} \parallel \boldsymbol{V} - \boldsymbol{H}^{(0)} \parallel^2 + \sum_{l=1}^{L} (\boldsymbol{\Delta}^{(l-1)})^T (\boldsymbol{H}^{(l-1)} - \mathrm{g}(\boldsymbol{W}^{(l)} \boldsymbol{H}^{(l)})). \tag{4}$$

This derivation has the advantage that the bottom-up activations $\boldsymbol{\Delta}^{(l)}$ have an interpretation as Lagrange multipliers.

Inverting the generative model by negative feedback can be interpreted as a process of sequential hypothesis testing. The top-down connections generate a hypothesis about the sensory data. The bottom-up connections propagate an error signal that is the disagreement between the hypothesis and data. When the error signal reaches the top, it is used to generate a revised hypothesis, and the generate-test-revise cycle starts all over again. Perception is the convergence of this feedback loop to the hypothesis that is most consistent with the data.

The synaptic weights $\boldsymbol{W}^{(l)}$ determine the types of patterns that the network is able to generate. To learn from examples, the weights are adjusted to improve the network's generation ability. A suitable cost function for learning is the reconstruction error $\frac{1}{2} \parallel \boldsymbol{V} - \boldsymbol{H}^{(0)} \parallel^2$ averaged over an ensemble of examples. Online gradient descent with respect to the synaptic weights is performed by a learning rule of the form

$$\Delta \boldsymbol{W}^{(l)} = \eta \boldsymbol{\Delta}^{(l-1)} (\boldsymbol{H}^{(l)})^T. \tag{5}$$

The same error signal $\boldsymbol{\Delta}^{(l)}$ that was used to invert the generative model is also used to learn it.

2.2. Nonnegative Matrix Factorization

Let a set of data be given as an $p \times N$ matrix $\boldsymbol{V}$, with each column consisting of the p non-negative pixel values of an image. Denote a set of basis images as a $p \times q$ matrix $\boldsymbol{W}$. Each image can be approximated as a linear combination of the basis images using coefficients.

$$\boldsymbol{V} \approx \boldsymbol{W}\boldsymbol{H} \tag{6}$$

where $\boldsymbol{H} \in \boldsymbol{R}^{q \times N}$ is the matrix which contains the coefficients.

PCA requires that the columns of $\boldsymbol{W}$ be orthonormal and the rows of $\boldsymbol{H}$ be mutually orthogonal. It imposes no other constraints than the orthogonality and hence allows the entries of $\boldsymbol{W}$ and $\boldsymbol{H}$ to be of arbitrary sign. Many basis images, for instance eigenfaces, lack intuitive meaning, and a linear combination of the basis images generally involves complex cancellations between positive and negative values. The NMF representations permit only nonnegative coefficients and thus non-subtractive combinations (Lee & Seung, 1999; Lee & Seung, 2001).

On the other hand NMF imposes the non-negativity constraints instead of the orthogonal basis and decorrelated hidden variables. The elements of $\boldsymbol{W}$ and $\boldsymbol{H}$ are all non-negative, and hence only non-subtractive combinations are allowed. This is believed to be compatible to the intuitive notion of combining parts to form a whole, and is how NMF learns the parts-based representation. It is also consistent with the physiological facts that the firing rates are non-negative and the signs of synapses do not change.

NMF uses the I-divergence of $\boldsymbol{V}$ from $\boldsymbol{W}\boldsymbol{H}$, defined as

$$\begin{aligned} & D(\boldsymbol{V} || \boldsymbol{W}\boldsymbol{H}) \\ = & \sum_{i,j} \left(\boldsymbol{V}_{ij} \log \frac{\boldsymbol{V}_{ij}}{(\boldsymbol{W}\boldsymbol{H})_{ij}} - \boldsymbol{V}_{ij} + (\boldsymbol{W}\boldsymbol{H})_{ij} \right) \end{aligned} \tag{7}$$

as the measure of fitness for factorizing $\boldsymbol{V}$ into $\boldsymbol{W}\boldsymbol{H}$. The NMF factorization is defined as

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{H}} \quad & D(\boldsymbol{V} || \boldsymbol{W}\boldsymbol{H}), \\ s.t \quad & \boldsymbol{W}, \boldsymbol{H} \geq 0. \end{aligned} \tag{8}$$

$D(\boldsymbol{V} || \boldsymbol{W}\boldsymbol{H})$ reduces to Kullback-Leibler divergence when $\sum_{i,j} \boldsymbol{V}_{ij} = \sum_{i,j} (\boldsymbol{W}\boldsymbol{H})_{ij} = 1$. The above optimization can be done by using multiplicative update rules.

$$\begin{aligned} \boldsymbol{H}_{a\mu} & \leftarrow \boldsymbol{H}_{a\mu} \frac{\sum_i \boldsymbol{W}_{ia} \boldsymbol{V}_{i\mu} / (\boldsymbol{W}\boldsymbol{H})_{i\mu}}{\sum_k \boldsymbol{W}_{ka}}, \\ \boldsymbol{W}_{ia} & \leftarrow \boldsymbol{W}_{ia} \frac{\sum_\mu \boldsymbol{H}_{a\mu} \boldsymbol{V}_{i\mu} / (\boldsymbol{W}\boldsymbol{H})_{i\mu}}{\sum_\nu \boldsymbol{H}_{a\nu}}. \end{aligned} \tag{9}$$

The algorithm performs both learning and inference simultaneously. That is, it learns a set of basis images and infers values for the hidden variables from

the visible variables. Although the generative model is linear, the inference computation is nonlinear due to the non-negativity constraints. The computation is a type of generalized EM algorithm.

Although NMF is successful in learning facial parts and semantic topics, this success does not imply that the method can learn parts from any database, such as images of objects viewed from extremely different viewpoints, or highly articulated objects. Learning parts for these complex cases is likely to require fully hierarchical models with multiple levels of hidden variables, instead of the single level in NMF (Lee & Seung, 1999).

3. The Multiplicative Up-Propagation Algorithm

There are a large number of neurons in inferior temporal cortex of monkeys which seem to encode an overall shape of biologically important objects - not specific features or parts. The finding agrees with hierarchical theories of object perception. According to these theories, cells in the cortical areas code elementary features such as line orientation and color. The outputs from these cells are then combined by detectors sensitive to higher-order features such as corners or intersections, an idea consistent with the findings of Hubel and Wiesel. The process is continued as each successive stage codes more complex combinations. At the top of the chain are IT neurons, selective for complex shapes like hands or faces. A huge number of hierarchical models for object recognition have been proposed over the years. Some of them were inspired by the desire to build intelligent machines, others by the desire to describe human recognition processes (Gazzaniga et al., 2001).

In this paper, we will try to invert the hierarchical recognition processes and view the visual perception as a hypothesis testing process. Helmholtz, in his doctrine of unconscious inference, argued that perceptions are formed by the interaction of bottom-up sensory data with top-down expectations. According to one interpretation of this doctrine, perception is a procedure of sequential hypothesis testing. We propose a new algorithm, called multiplicative up-propagation, that realizes this interpretation in layered networks. It uses top-down connections to generate hypotheses, and bottom-up connections to revise them.

How can we build such a hierarchical structure of special neurons which are responsible only for local sensory features by multi-layer generative model? We have already known that NMF can give local features and sparse codes from a given non-negative data set. It is just a single-layer generative network and gives local features and sparse codes. Then what about multilayer network with nonnegative weights and hidden variables? It is what we will introduce in this section. The previous notations of $\boldsymbol{W}$ and $\boldsymbol{H}$ in a single-layer network are changed into $\boldsymbol{W}^{(1)}$ and $\boldsymbol{H}^{(1)}$. We start by assuming further factorizations:

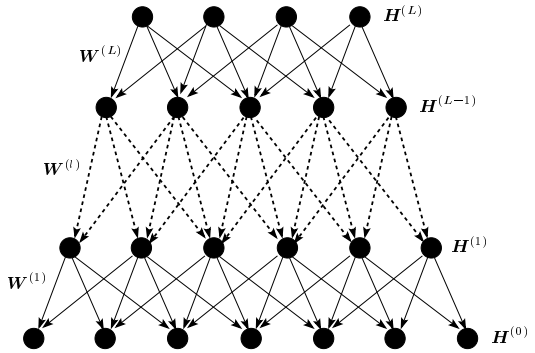


Figure 2. Multilayer generative network model: $\boldsymbol{V} \approx \boldsymbol{H}^{(0)} = \mathsf{g}\left(\boldsymbol{W}^{(1)}\mathsf{g}\left(\boldsymbol{W}^{(2)}\mathsf{g}\left(\cdots \mathsf{g}\left(\boldsymbol{W}^{(L)}\boldsymbol{H}^{(L)}\right)\right)\right)\right)$ and $\boldsymbol{H}^{(l)} = \mathsf{g}(\boldsymbol{W}^{(l+1)}\boldsymbol{H}^{(l+1)})$. Note that it will be fully connected.

$$\boldsymbol{H}^{(l)}_{a\mu} = \mathsf{g}\left((\boldsymbol{W}^{(l+1)}\boldsymbol{H}^{(l+1)})_{a\mu}\right), \tag{10}$$

where $l = 0, 1, \cdots, L-1$. The nonlinear function g, which must output nonnegative values and increase monotonically, is applied elementwise to the matrix. Then we can construct a L-layer nonnegative networks as shown in Fig. 2. It is basically a generative model and constructs the nonnegative input data at its bottom layer. Before we show how it finds a new representation from the nonnegative data, we should think of the algorithm which we can find the optimized values of weights and hidden variables with. In order to extend NMF to multilayer case, let us introduce two matrices $\boldsymbol{R}$ and $\boldsymbol{N}$ to the update rules of equation (9):

$$\begin{aligned} \boldsymbol{H}_{a\mu} &\leftarrow \boldsymbol{H}_{a\mu}\frac{\sum_i \boldsymbol{W}_{ia}\dfrac{\boldsymbol{V}_{i\mu}}{(\boldsymbol{W}\boldsymbol{H})_{i\mu}}}{\sum_k \boldsymbol{W}_{ka}} \\ &= \boldsymbol{H}_{a\mu}\frac{\sum_i \boldsymbol{W}_{ia}\boldsymbol{R}_{i\mu}}{\sum_k \boldsymbol{W}_{ka}\boldsymbol{N}_{k\mu}} \\ &= \boldsymbol{H}_{a\mu}\frac{(\boldsymbol{W}^T\boldsymbol{R})_{a\mu}}{(\boldsymbol{W}^T\boldsymbol{N})_{a\mu}}, \end{aligned} \tag{11}$$

Table 1. Cost functions and initializations of $\boldsymbol{N}^{(l)}$ and $\boldsymbol{R}^{(l)}$: If we use other cost functions, the initializations should be changed

F	$\boldsymbol{N}^{(1)}$	$\boldsymbol{R}^{(1)}$
$\parallel \boldsymbol{V} - \boldsymbol{H}^{(0)} \parallel^R$	$\boldsymbol{N}_{i\mu}^{(1)} = (\boldsymbol{H}_{i\mu}^{(0)})^{R-1}\mathsf{g}'((\boldsymbol{W}^{(1)}\boldsymbol{H}^{(1)})_{i\mu})$	$\boldsymbol{R}_{i\mu}^{(1)} = (\boldsymbol{V}_{i\mu}^{(0)})^{R-1}\mathsf{g}'((\boldsymbol{W}^{(1)}\boldsymbol{H}^{(1)})_{i\mu})$
$\sum_{i\mu} \boldsymbol{V}_{i\mu}\log(\boldsymbol{H}_{i\mu}^{(0)}) - \boldsymbol{H}_{i\mu}^{(0)}$	$\boldsymbol{N}_{i\mu}^{(1)} = \mathsf{g}'((\boldsymbol{W}^{(1)}\boldsymbol{H}^{(1)})_{i\mu})$	$\boldsymbol{R}_{i\mu}^{(1)} = \mathsf{g}'((\boldsymbol{W}^{(1)}\boldsymbol{H}^{(1)})_{i\mu})\boldsymbol{V}_{i\mu}/\boldsymbol{H}_{i\mu}^{(0)}$
$\sum_{i\mu} \boldsymbol{V}_{i\mu}/\boldsymbol{H}_{i\mu}^{(0)} + \log \boldsymbol{H}_{i\mu}^{(0)}$	$\boldsymbol{N}_{i\mu}^{(1)} = \mathsf{g}'((\boldsymbol{W}^{(1)}\boldsymbol{H}^{(1)})_{i\mu})/\boldsymbol{H}_{i\mu}^{(0)}$	$\boldsymbol{R}_{i\mu}^{(1)} = \mathsf{g}'((\boldsymbol{W}^{(1)}\boldsymbol{H}^{(1)})_{i\mu})\boldsymbol{V}_{i\mu}/(\boldsymbol{H}_{i\mu}^{(0)})^2$

and

$$\begin{aligned} \boldsymbol{W}_{ia} &\leftarrow \boldsymbol{W}_{ia}\frac{\sum_\mu \frac{\boldsymbol{V}_{i\mu}}{(\boldsymbol{W}\boldsymbol{H})_{i\mu}}\boldsymbol{H}_{a\mu}}{\sum_\nu \boldsymbol{H}_{a\nu}} \\ &= \boldsymbol{W}_{ia}\frac{\sum_\mu \boldsymbol{R}_{i\mu}\boldsymbol{H}_{a\mu}}{\sum_\nu \boldsymbol{N}_{i\nu}\boldsymbol{H}_{a\nu}} \\ &= \boldsymbol{W}_{ia}\frac{(\boldsymbol{R}\boldsymbol{H}^T)_{ia}}{(\boldsymbol{N}\boldsymbol{H}^T)_{ia}} \end{aligned} \tag{12}$$

where $\boldsymbol{R} = \boldsymbol{R}^{(1)}$ is a matrix of ratios of $\boldsymbol{V}$ to $\boldsymbol{W}\boldsymbol{H}$ and $\boldsymbol{N} = \boldsymbol{N}^{(1)}$ is a matrix of normalizing factors with all elements 1. If we put $\boldsymbol{R} = \boldsymbol{V}$ and $\boldsymbol{N} = \boldsymbol{W}\boldsymbol{H}$, the above rules will be for least squared error under nonnegativity. In fact such formulations can be applied to all cost functions F that are defined as follows:

Definition 1 F *is a cost function that quantify the quality of the approximation between* $\boldsymbol{V}$ *and* $\boldsymbol{H}^{(0)} = \mathsf{g}(\boldsymbol{W}^{(1)}\mathsf{g}(\boldsymbol{W}^{(2)}\cdots))$*, if the conditions*

$$\begin{aligned} \frac{\partial \mathsf{F}}{\partial \boldsymbol{W}^{(l)}} &= (\boldsymbol{N}^{(l)} - \boldsymbol{R}^{(l)})(\boldsymbol{H}^{(l)})^T \\ \frac{\partial \mathsf{F}}{\partial \boldsymbol{H}^{(L)}} &= (\boldsymbol{W}^{(L)})^T(\boldsymbol{N}^{(L)} - \boldsymbol{R}^{(L)}) \end{aligned} \tag{13}$$

are satisfied, where $\boldsymbol{R}^{(l)}$ *and* $\boldsymbol{N}^{(l)}$ *should be nonnegative matrices for arbitrary nonnegative matrices* $\boldsymbol{W}^{(l)}$ *and* $\boldsymbol{H}^{(l)}$*.*

If we obtain the matrix $\boldsymbol{R}^{(l)}$ and $\boldsymbol{N}^{(l)}$ at lth layer of the networks, we can optimize the multi-layer networks by using the similar update rules at all layers:

$$\boldsymbol{W}_{ia}^{(l)} \leftarrow \boldsymbol{W}_{ia}^{(l)}\left[\frac{\left(\boldsymbol{R}^{(l)}\boldsymbol{H}^{(l)T}\right)_{ia}}{\left(\boldsymbol{N}^{(l)}\boldsymbol{H}^{(l)T}\right)_{ia}}\right]^\eta \tag{14}$$

and

$$\boldsymbol{H}_{a\mu}^{(L)} \leftarrow \boldsymbol{H}_{a\mu}^{(L)}\left[\frac{\left(\boldsymbol{W}^{(L)T}\boldsymbol{R}^{(L)}\right)_{a\mu}}{\left(\boldsymbol{W}^{(L)T}\boldsymbol{N}^{(L)}\right)_{a\mu}}\right]^\eta, \tag{15}$$

where the learning rate $0 < \eta \ll 1$ should be considered in nonlinear multilayer networks. Fortunately, the $\boldsymbol{R}^{(l)}$ and $\boldsymbol{N}^{(l)}$ matrices are calculated by the following up-propagation rules:

$$\begin{aligned} \boldsymbol{R}_{i\mu}^{(l+1)} &= \left(\boldsymbol{W}^{(l)T}\boldsymbol{R}^{(l)}\right)_{i\mu}\mathsf{g}'\left(\boldsymbol{W}^{(l)}\boldsymbol{H}^{(l)}\right)_{i\mu}, \\ \boldsymbol{N}_{i\mu}^{(l+1)} &= \left(\boldsymbol{W}^{(l)T}\boldsymbol{N}^{(l)}\right)_{i\mu}\mathsf{g}'\left(\boldsymbol{W}^{(l)}\boldsymbol{H}^{(l)}\right)_{i\mu}, \end{aligned} \tag{16}$$

where $l = 1, 2, \cdots, L-1$ and

$$\begin{aligned} \boldsymbol{R}_{i\mu}^{(1)} &= \frac{\boldsymbol{V}_{i\mu}}{(\boldsymbol{W}^{(1)}\boldsymbol{H}^{(1)})_{i\mu}}\mathsf{g}'\left((\boldsymbol{W}^{(1)}\boldsymbol{H}^{(1)})_{i\mu}\right), \\ \boldsymbol{N}_{i\mu}^{(1)} &= \mathsf{g}'\left((\boldsymbol{W}^{(1)}\boldsymbol{H}^{(1)})_{i\mu}\right). \end{aligned} \tag{17}$$

Another form of initializations is given in Table 1.

Theorem 1 F *is nonincreasing under the update rules of Eqs. (14) and (15). The cost is invariant under these updates if and only if* $\boldsymbol{W}^{(l)}$ *and* $\boldsymbol{H}^{(L)}$ *are at a stationary point of the cost function.*

To prove Theorem 1, we will make use of the auxiliary function that was used in (Lee & Seung, 2001).

Definition 2 $\mathsf{G}(\boldsymbol{W}^{(l)}, \boldsymbol{W}_t^{(l)})$ *is an auxiliary function for* $\mathsf{F}(\boldsymbol{W}^{(l)})$ *if the conditions*

$$\mathsf{G}(\boldsymbol{W}^{(l)}, \boldsymbol{W}_t^{(l)}) \geq \mathsf{F}(\boldsymbol{W}^{(l)}), \quad \mathsf{G}(\boldsymbol{W}_t^{(l)}, \boldsymbol{W}_t^{(l)}) = \mathsf{F}(\boldsymbol{W}_t^{(l)})$$

are satisfied. t *is a discrete time index.*

The auxiliary function is a useful concept because of the following lemma.

Lemma 1 *If* G *is an auxiliary function, then* F *is nonincreasing under the update*

$$\boldsymbol{W}_{t+1}^{(l)} = \arg\min_{W^{(l)}} \mathsf{G}(\boldsymbol{W}^{(l)}, \boldsymbol{W}_t^{(l)})$$

Proof: $\mathsf{F}(\boldsymbol{W}_{t+1}^{(l)}) \leq \mathsf{G}(\boldsymbol{W}_{t+1}^{(l)}, \boldsymbol{W}_t^{(l)}) \leq \mathsf{G}(\boldsymbol{W}_t^{(l)}, \boldsymbol{W}_t^{(l)}) = \mathsf{F}(\boldsymbol{W}_t^{(l)})$. ■

Lemma 2 *If the derivatives of* $\mathsf{G}(\boldsymbol{W}^{(l)}, \boldsymbol{W}_t^{(l)})$ *are*

$$\begin{aligned}&\frac{\partial \mathsf{G}(\boldsymbol{W}^{(l)}, \boldsymbol{W}_t^{(l)})}{\partial (\boldsymbol{W}^{(l)})_{ia}} \\ =\ & \frac{\left(\boldsymbol{W}^{(l)}\right)_{ia}^{1/\eta}}{\left(\boldsymbol{W}_t^{(l)}\right)_{ia}^{1/\eta}} \left(\boldsymbol{N}^{(l)}(\boldsymbol{H}^{(l)})^T\right)_{ia} - \left(\boldsymbol{R}^{(l)}(\boldsymbol{H}^{(l)})^T\right)_{ia},\end{aligned}$$

there exists an auxiliary function $\mathsf{G}(\boldsymbol{W}^{(l)}, \boldsymbol{W}_t^{(l)})$ *for* $\mathsf{F}(\boldsymbol{W}_t^{(l)})$.

Proof: Let E be the difference of G from F, $\mathsf{G} - \mathsf{F}$. Then the derivatives of $\mathsf{E}(\boldsymbol{W}^{(l)}, \boldsymbol{W}_t^{(l)})$ are

$$\begin{aligned}&\frac{\partial \mathsf{E}(\boldsymbol{W}^{(l)}, \boldsymbol{W}_t^{(l)})}{\partial (\boldsymbol{W}^{(l)})_{ia}} \\ =\ & \frac{\left(\boldsymbol{W}^{(l)}\right)_{ia}^{1/\eta} - \left(\boldsymbol{W}_t^{(l)}\right)_{ia}^{1/\eta}}{\left(\boldsymbol{W}_t^{(l)}\right)_{ia}^{1/\eta}} \{\boldsymbol{N}^{(l)}(\boldsymbol{H}^{(l)})^T\}_{ia}.\end{aligned}$$

Since the derivatives are nonnegative for $(\boldsymbol{W}^{(l)})_{ia} \geq (\boldsymbol{W}_t^{(l)})_{ia}$ and negative for $(\boldsymbol{W}^{(l)})_{ia} < (\boldsymbol{W}_t^{(l)})_{ia}$, there exists the auxiliary function $\mathsf{E}(\boldsymbol{W}^{(l)}, \boldsymbol{W}_t^{(l)}) \geq 0$ by adding a constant so that E can be zero at $\boldsymbol{W}^{(l)} = \boldsymbol{W}_t^{(l)}$. ■

Proof of Theorem 1 By lemma 1, putting the derivatives zero in lemma 2 results in the update rule:

$$\begin{aligned}(\boldsymbol{W}_{t+1}^{(l)})_{ia} &= (\boldsymbol{W}_t^{(l)})_{ia} \frac{\left\{\boldsymbol{R}_{t+1}^{(l)}(\boldsymbol{H}^{(l)})^T\right\}_{ia}^{\eta}}{\left\{\boldsymbol{N}_{t+1}^{(l)}(\boldsymbol{H}^{(l)})^T\right\}_{ia}^{\eta}} \\ &\approx (\boldsymbol{W}_t^{(l)})_{ia} \frac{\left\{\boldsymbol{R}_t^{(l)}(\boldsymbol{H}^{(l)})^T\right\}_{ia}^{\eta}}{\left\{\boldsymbol{N}_t^{(l)}(\boldsymbol{H}^{(l)})^T\right\}_{ia}^{\eta}}, \quad (18)\end{aligned}$$

from $\boldsymbol{R}_{t+1}^{(l)} \approx \boldsymbol{R}_t^{(l)}$ and $\boldsymbol{N}_{t+1}^{(l)} \approx \boldsymbol{N}_t^{(l)}$. Eq. (15) can be shown similarly.
On the other hand, we can derive the up-propagation rules from the definition 1. By considering

$$\frac{\partial \mathsf{F}}{\partial (\boldsymbol{W}^{(l)}\boldsymbol{H}^{(l)})} = \boldsymbol{N}^{(l)} - \boldsymbol{R}^{(l)}$$

from the chain rule $\frac{\partial \mathsf{F}}{\partial \boldsymbol{W}^{(l)}} = \frac{\partial \mathsf{F}}{\partial (\boldsymbol{W}^{(l)}\boldsymbol{H}^{(l)})}(\boldsymbol{H}^{(l)})^T$ and the eqn. (13), we get

$$\begin{aligned}&\boldsymbol{N}^{(l+1)} - \boldsymbol{R}^{(l+1)} \\ =\ & \frac{\partial \mathsf{F}}{\partial (\boldsymbol{W}^{(l+1)}\boldsymbol{H}^{(l+1)})} \\ =\ & \frac{\partial \mathsf{F}}{\partial \mathsf{g}(\boldsymbol{W}^{(l+1)}\boldsymbol{H}^{(l+1)})} \odot \mathsf{g}'(\boldsymbol{W}^{(l+1)}\boldsymbol{H}^{(l+1)}) \\ =\ & \frac{\partial \mathsf{F}}{\partial \boldsymbol{H}^{(l)}} \odot \mathsf{g}'(\boldsymbol{W}^{(l+1)}\boldsymbol{H}^{(l+1)}) \\ =\ & \left[(\boldsymbol{W}^{(l)})^T \frac{\partial \mathsf{F}}{\partial (\boldsymbol{W}^{(l)}\boldsymbol{H}^{(l)})}\right] \odot \mathsf{g}'(\boldsymbol{W}^{(l+1)}\boldsymbol{H}^{(l+1)}) \\ =\ & \left[(\boldsymbol{W}^{(l)})^T(\boldsymbol{N}^{(l)} - \boldsymbol{R}^{(l)})\right] \odot \mathsf{g}'(\boldsymbol{W}^{(l+1)}\boldsymbol{H}^{(l+1)}),\end{aligned}$$

where the 3rd and 5th equations are expanded by using the chain rules.

Thus, $\boldsymbol{N}^{(l+1)} - \boldsymbol{R}^{(l+1)} = \left[(\boldsymbol{W}^{(l)})^T(\boldsymbol{N}^{(l)} - \boldsymbol{R}^{(l)})\right] \odot \mathsf{g}'(\boldsymbol{W}^{(l+1)}\boldsymbol{H}^{(l+1)})$ or equation (16) can be obtained. ■

The proposed algorithm is very similar with the error up-propagation algorithm for multi-layer neural network model. We can find the rule by which error up-prop algorithm is modified into multiplicative up-prop algorithm: Elementwise addition(or subtraction) is changed into elementwise multiplication(or division) and the multiplication of the learning rate η is modified into the power of η. Both of them also use the same type of up-prop rules (2) and (16). But they are also different in a couple of aspects. The error up-propagation algorithm propagates only error, whereas the proposed algorithm should separately propagate both normalizing factors and ratios of inputs to reconstructed values.

4. Experiments

We applied the three-layer network model to a set of facial images and trained it by using the multiplicative up-propagation rules. As with all gradient-based algorithms, the multiplicative up-propagation rules are susceptible to local optima. Crucial to the success of our experiments is a pertinent network growing algorithm.

4.1. Network Growing

We begin the training of the networks from a single-layer networks. In the first stage, the given data matrix $\boldsymbol{V}$ are factorized into $\mathsf{g}(\boldsymbol{W}^{(1)}\boldsymbol{H}^{(1)})$. Then we add one layer on that and train the second layer so that the $\boldsymbol{H}^{(1)}$ can be factorized into $\mathsf{g}(\boldsymbol{W}^{(2)}\boldsymbol{H}^{(2)})$. The two-layer networks are then trained by using the mul-

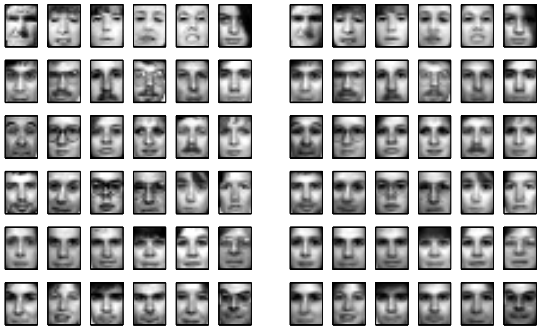


Figure 3. (a) original face images; (b) reconstructed face images.

tiplicative up-prop rules for the input matrix $\boldsymbol{V}$ to be approximated by the matrix $\boldsymbol{H}^{(0)}$. The process to grow the networks from the two-layer to three-layer is the same as the previous steps. However, the method does not assure the globally optimal solution to us. It only helps avoid some worst solutions. Also note that applying NMF recursively to the hidden values $\boldsymbol{H}^{(l)}$ is not the same as our multiplicative up-propagation rules.

4.2. Hierarchical Feature Analysis

We obtain a hierarchical set of features of facial images through the three-layer generative networks. By applying the model and learning algorithm to a set of size-normalized and eye-centered 1608 facial images with size 19×14 (see Fig. 3), we obtained the visually hierarchical features. The transformed weights of the first layer represent the highly localized features of Fig. 4c, whereas the features of Fig. 4a correspond to the nodes of top layer. The features of Fig. 4b are similar to those of NMF. The complex features in a layer are defined by the less complex features in the subordinate layers, which is the reason why we name the application a 'hierarchical feature analysis'.

This property is specially helpful in the case that we should find the hierarchical relations or dependencies of the features. We've already made an application to the case and obtained some successful results (Ahn et al., 2004). We can develope the algorithm and perform the hierarchical clustering of a set of nonnegative data by constraining the algorithm to a condition such that the summation of column vectors is one.

4.3. Representational Learning in IT Cortex

We are already familiar with the fact that mammalian visual system is hierarchically constructed, through

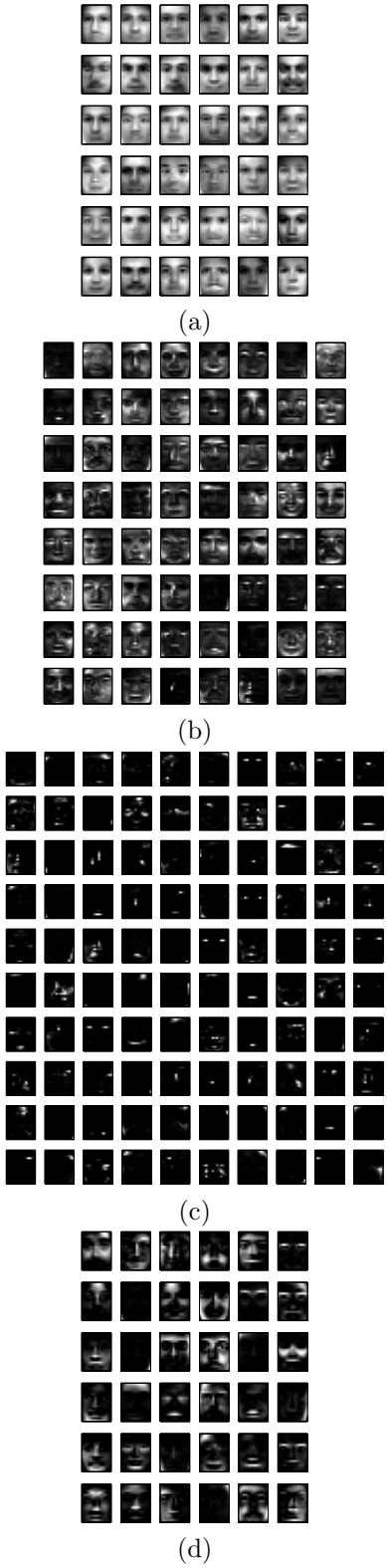


Figure 4. Three sets of facial features in a hierarchical relationship: (a) column vectors of $\mathsf{g}(\boldsymbol{W}^{(1)}\mathsf{g}(\boldsymbol{W}^{(2)}\mathsf{g}(\boldsymbol{W}^{(3)})))$; (b) column vectors of $\mathsf{g}(\boldsymbol{W}^{(1)}\mathsf{g}(\boldsymbol{W}^{(2)}))$; (c) column vectors of $\mathsf{g}(\boldsymbol{W}^{(1)})$; (d) features obtained by NMF

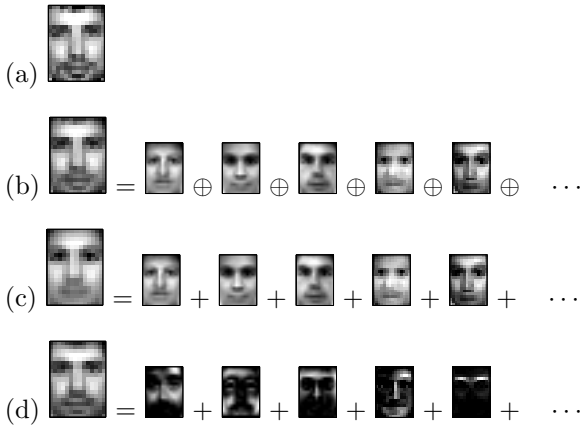


Figure 5. Whole-based representation vs. Parts-based representation: (a) original face; (b) our method (nonlinear summation, $\sum_k \mathrm{g}(\boldsymbol{W}^{(1)}\mathrm{g}(\boldsymbol{W}^{(2)}\mathrm{g}(\boldsymbol{W}^{(3)}h_k^{(3)})))$; (c) our method (linear summation, $\sum_k \mathrm{g}(\boldsymbol{W}^{(1)}\mathrm{g}(\boldsymbol{W}^{(2)}\mathrm{g}(\boldsymbol{W}^{(3)})))h_k^{(3)}$); (d) NMF.

which it acquires the flexible and invariant recognition ability. What we are interested in here is IT cortex, and its representational learning. The areas receive their inputs via a number of cortico-cortical stages from the primary visual cortex, striate cortex, through pre-striate visual areas. Neurons that are found in certain area of IT cortex preferentially or selectively to faces. When they were found in the pioneer days, some researchers conceived that they could respond to the whole face of a particular individual, that is 'Grandmother Cell' hypothesis. But it resulted in a few of problems such that generalization, noise robustness and excess of number of neurons. Instead, ensemble coding theory was popularized for representational learning of IT cortex. So to speaking, when a face is considered, a special set of parts is required from a pool of various types of parts such that eyes, nose and mouth. NMF showed a principle for the parts-based representation and it was noticed. But what is a role of the neurons that respond only to faces? Here we argue that there could exist a whole-based representation of faces in Fig. 5. We guess together that representations of facial information would be different according to the level in hierarchy, and the top layer of IT cortex could use the whole-based representations.

5. Conclusion

We proposed a new algorithm, multiplicative up-propagation, for training the multilayer generative network model with the nonnegativity. It could be considered as a nonlinear and multilayer extension of NMF. We applied the three-layer generative network model to the eye-aligned facial images and showed details of parts and composites in hierarchical relations. We argue that these results better explain the representational learning of IT cortex.

6. Acknowledgment

This work was supported by Korea Ministry of Science and Technology under Brain Science and Engineering Research Program, KOSEF 2000-2-20500-009-5, and Brain Korea 21 in POSTECH.

References

Ahn, J. H., Kim, S., Oh, J. H., & Choi, S. (2004). Multiple nonnegative-matrix factorization of dynamic PET images. *Proc. Asian Conf. Computer Vision.*

Cho, Y. C., Choi, S., & Bang, S. Y. (2003). Non-negative component parts of sound for classification. *Proc. IEEE ISSPIT.* Darmstadt, Germany.

Gazzaniga, M. S., Ivry, R. B., & Mangum, G. R. (2001). *Cognitive neuroscience: The biology of the mind.* New York: W. W. Norton & Company.

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, *401*, 788–791.

Lee, D. D., & Seung, H. S. (2001). Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems.*

Lee, J. S., Lee, D. D., Choi, S., & Lee, D. S. (2001). Application of non-negative matrix factorization to dynamic positron emission tomography. *Proc. ICA* (pp. 629–632). San Diego, California.

Li, S. Z., Hou, X. W., Zhang, H. J., & Cheng, Q. S. (2001). Learning spatially localized parts-based representation. *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (pp. 207–212). Kauai, Hawaii.

Oh, J. H., & Seung, H. S. (1997). Learning generative models with the up-propagation algorithm. *Advances in Neural Information Processing Systems* (pp. 605–611). MIT.

Saul, L. K., Sha, F., & Lee, D. D. (2003). Statistical signal processing with nonnegativity constraints. *Proc. EUROSPEECH* (pp. 1001–1004). Geneva, Switzerland.