Journal of Machine Learning Research 9 (2008) 1227-1267 Submitted 5/07; Revised 11/07; Published 6/08

Maximal Causes for Non-linear Component Extraction

Jörg Lücke LUCKE@GATSBY.UCL.AC.UK
Maneesh Sahani MANEESH@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit
University College London
17 Queen Square
London WC1N 3AR, UK

Editor: Yoshua Bengio

Abstract

We study a generative model in which hidden causes combine competitively to produce observations. Multiple active causes combine to determine the value of an observed variable through a max function, in the place where algorithms such as sparse coding, independent component analysis, or non-negative matrix factorization would use a sum. This max rule can represent a more realistic model of non-linear interaction between basic components in many settings, including acoustic and image data. While exact maximum-likelihood learning of the parameters of this model proves to be intractable, we show that efficient approximations to expectation-maximization (EM) can be found in the case of sparsely active hidden causes. One of these approximations can be formulated as a neural network model with a generalized softmax activation function and Hebbian learning. Thus, we show that learning in recent softmax-like neural networks may be interpreted as approximate maximization of a data likelihood. We use the bars benchmark test to numerically verify our analytical results and to demonstrate the competitiveness of the resulting algorithms. Finally, we show results of learning model parameters to fit acoustic and visual data sets in which max-like component combinations arise naturally.

Keywords: component extraction, maximum likelihood, approximate EM, competitive learning, neural networks

1. Introduction

In recent years, algorithms such as independent components analysis (ICA; Comon, 1994; Bell and Sejnowski, 1997), sparse coding (SC; Olshausen and Field, 1996), and non-negative matrix factorization (NMF; Lee and Seung, 1999) have been used to describe the statistics of the natural environment, and the components extracted by these methods have been linked to sensory neuronal response properties. Stated in the language of probabilistic generative models (see, e.g., Dayan and Abbott, 2001; Rao et al., 2002) these systems describe sensory data as a linear superposition of learned components. For many types of data, including images, this assumed linear cooperation between generative causes is unrealistic. Alternative, more competitive generative models have also been proposed: for instance, Saund (1995) suggests a model in which hidden causes are combined by a noisy-or rule, while Dayan and Zemel (1995) suggest a yet more competitive scheme. Here, we formulate an extreme case of competition, in which the strongest generative influence on an observed variable (e.g., an image pixel) alone determines its value. Such a rule has the property of selecting, for each observed variable, a single generative cause to determine that variable's value.

©2008 Jörg Lücke and Maneesh Sahani.

This form of combination emerges naturally in the context of spectrotemporal masking in mixed audio signals. For image data, occlusion leads to a different combination rule, but one that shares the selection property in that, under constant lighting conditions, the appearance of each observed pixel is determined by a single object.

In parallel to this development of generative approaches, a number of artificial neural network architectures have been designed to tackle the problem of non-linear component extraction, mostly in artificial data (e.g., Spratling and Johnson, 2002; Lücke and von der Malsburg, 2004; Lücke and Bouecke, 2005; Spratling, 2006), although sometimes in natural images (e.g., Harpur and Prager, 1996; Charles et al., 2002; Lücke, 2007). These models often perform quite well with respect to various benchmark tests. However, the relationship between them and the density models that are implicit or explicit in the generative approach has not, thus far, been made clear. We show here that inference and learning in a restricted form of our novel generative model correspond closely in form to the processing and plasticity rules used in such neural network approaches, thus bringing together these two disparate threads of investigation.

The organization of the remainder of this article is as follows. In Section 2 we define the novel generative model and then proceed to obtain the associated parameter update rules in Section 3. In Section 4 we derive computationally efficient approximations to these update rules, in the context of sparsely active hidden causes—that is, when a small number of hidden causes generally suffices to explain the data. In Section 5 we relate a restricted form of the generative model to neural network learning rules with Hebbian plasticity and divisive normalization. Results of numerical experiments in Section 6 show the component extraction performance of the generative schemes as well as a comparison to other algorithms. Finally, in Section 7, we discuss our analytical and numerical results.

2. A Generative Model with Maximum Non-linearity

We consider a generative model for D observed variables y_d, $(d = 1,\ldots,D)$, in which H hidden binary causes s_h, $(h = 1,\ldots,H)$, each taking the value 0 or 1, *compete* to determine the value of each observation (see Figure 1). Associated with each pair (s_h, y_d), is a weight W_{hd}. Given a set of active causes (i.e., those taking the value 1), the distribution of y_d is determined by the *largest* of the weights associated with the active causes and y_d.

Much of our discussion will apply generally to all models of this causal structure, irrespective of the details of the distributions involved. For concreteness, however, we focus on a particular choice, in which the hidden variables are drawn from a multivariate Bernoulli distribution; and the observed variables are non-negative, integer-valued and, given the causes, conditionally independent and Poisson-distributed. Thus, collecting all the causes into a single binary vector $\vec{s} \in \{0,1\}^H$, and all the observed variables into an integer vector $\vec{y} \in \mathbb{Z}_+^D$ we have:

$$p(\vec{s}\,|\,\vec{\pi}) = \prod_{h=1}^{H} p(s_h\,|\,\pi_h), \quad p(s_h\,|\,\pi_h) = \pi_h^{s_h}\,(1-\pi_h)^{1-s_h}, \tag{1}$$

$$p(\vec{y}\,|\,\vec{s},W) = \prod_{d=1}^{D} p(y_d\,|\,\overline{W}_d(\vec{s},W)), \quad p(y_d\,|\,w) = \frac{w^{y_d}}{y_d!}e^{-w}. \tag{2}$$

Here, $\vec{\pi} \in [0,1]^H$ parameterizes the prior distribution on $\vec{s}$, while the weight matrix $W \in \mathrm{R}^{H\times D}$ parameterizes the influence of the hidden causes on the distribution of $\vec{y}$. It will be convenient to

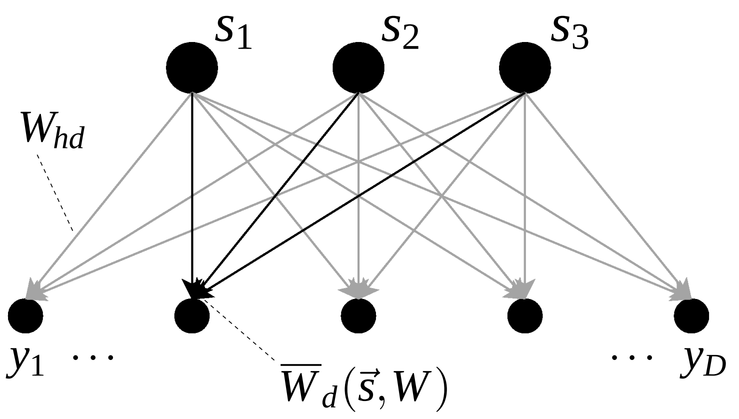

Figure 1: A generative model with $H = 3$ hidden variables and $D = 5$ observed variables. The values y_d of the observed variables are conditionally independent given the values $\vec{s}$ of the hidden variables. The value y_d is drawn from a distribution which is determined by the parameters W_{1d}, W_{2d}, and W_{3d}. For a given binary vector $\vec{s}$ these parameters combine competitively according to the function $\overline{W}_d(\vec{s}, W) = \max_h\{s_h W_{hd}\}$.

group these parameters together into $\Theta = (\vec{\pi}, W)$. The function $\overline{W}_d(\vec{s}, W)$ in (2) gives the *effective weight* on y_d, resulting from a particular pattern of causes $\vec{s}$. Thus, in the model considered here,

$$\overline{W}_d(\vec{s}, W) \quad = \quad \max_h\{s_h W_{hd}\}. \tag{3}$$

It is useful to place the model (1)–(3) in context. Models of this general type, in which the observations are conditionally independent of one another given a set of hidden causes, are widespread. They underlie algorithms such as ICA, SC, principal components analysis (PCA), factor analysis (see, e.g., Everitt, 1984), and NMF. In these five cases, and indeed in the majority of such models studied, the effective weights $\overline{W}_d(\vec{s}, W)$ are formed by a linear combination of all the weights that link hidden variables to the observation; that is, $\overline{W}_d(\vec{s}, W) = \sum_h s_h W_{hd}$. Some other models, notably those of Saund (1995) and Dayan and Zemel (1995), have implemented more competitive combination rules, where larger individual weights dominate the effective combination. The present model takes this competition to an extreme, so that only the single largest weight (amongst those associated with active hidden variables) determines the output distribution. Thus, where ICA, PCA, SC, or NMF use a sum, we use a max. We refer to this new generative model as the Maximal Causes Analysis (MCA) model.

Figure 2 illustrates the difference between linear superposition and competitive combination using (3). Let us suppose that noise-free observations are generated by causes in the form of horizontal and vertical objects with the same gray-value, on a dark (black) background (see Figure 2). If these objects occlude one-another, they may generate an observed image such as that illustrated in Figure 2B. However, if we were to use the actual causes and weights in Figure 2A, but instead combine them linearly, we would obtain the (different) input pattern of Figure 2C. In this case, competitive combination using the max-rule of Equation (3) would result in the correct pattern. This is not, of course, generally true, but for monochrome objects with small variations in their gray-values it

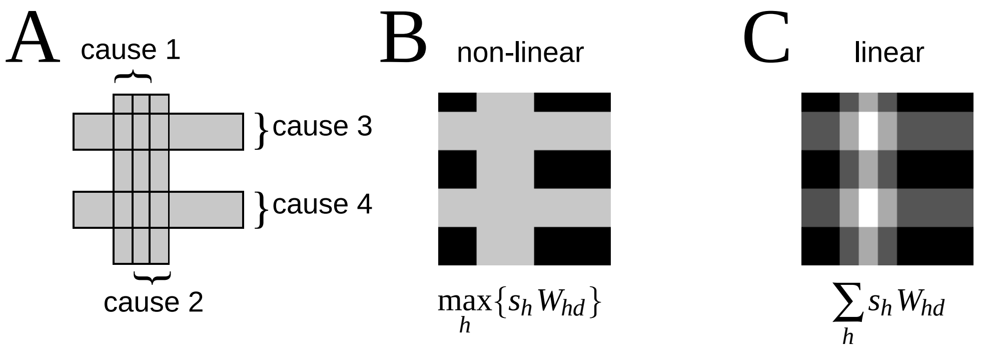

Figure 2: An illustration of non-linear versus linear combination of hidden causes. **A** Four examples of hidden causes with gray-value 200. **B** The input image that may result if sources occlude one another. In this case, the correct function $\overline{W}_d(\vec{s},W)$ (see Figure 1) to combine the hidden causes is the max-operation. **C** The input image that results if the four causes combine linearly (gray-values are scaled to fill the interval [0,255]). For **C**, the correct function $\overline{W}_d(\vec{s},W)$ is linear super-position.

holds approximately. More generally, the maximum combination rule is always closer to the result of occlusion than is the simple sum implied by models such as ICA.

As stated above, although in this paper we focus on the specific distributions given in (1) and (2), much of the analytical treatment is independent of these specific choices. Thus, update rules for learning the weights W from data will be derived in a general form, that can accommodate alternative, non-factored distributions for the binary hidden variables. This general form is also preserved if the Poisson distribution is replaced, for example, by a Gaussian. Poisson variability represents a reasonable choice for the non-negative data considered in this paper, and resembles the cost function introduced by Lee and Seung (1999) for NMF.

3. Maximum Likelihood

Given a set of observed data vectors $Y=\{\vec{y}^{(n)}\}_{n=1,\dots,N}$, taken to be generated independently from a stationary process, we seek parameter values $\Theta^* = (\vec{\pi}^*, W^*)$ that maximize the likelihood of the data under the generative model of Equations (1) to (3):

$$\Theta^* \;=\; \mathrm{argmax}_{\Theta}\{\mathcal{L}(\Theta)\} \quad \text{with} \quad \mathcal{L}(\Theta) = \log\Big(p(\vec{y}^{(1)},\dots,\vec{y}^{(N)}\,|\,\Theta)\Big)\,.$$

We use Expectation-Maximization (EM; Dempster et al. 1977; see also Neal and Hinton 1998, for the formulation that appears here) to maximize the likelihood in this latent variable model. To do so, we introduce the free-energy $\mathcal{F}(\Theta,q)$—a data-dependent function of the parameters Θ and an unknown distribution $q(\vec{s}^{(1)},\dots,\vec{s}^{(N)})$ over the hidden data or variables—that is always equal to or less than the likelihood evaluated at the same parameter values. For independently generated data vectors $\vec{y}^{(n)}$, the distribution q may be taken (without loss of generality) to factor over the hidden vectors $q(\vec{s}^{(1)},\ldots,\vec{s}^{(N)}) = \prod_n q_n(\vec{s}^{(n)})$. Then the free-energy is defined as:

$$\mathcal{F}(\Theta,q) = \sum_{n=1}^{N}\left[\sum_{\vec{s}} q_n(\vec{s}) \left[\log\left(p(\vec{y}^{(n)}\,|\,\vec{s},\Theta)\right) + \log\left(p(\vec{s}\,|\,\Theta)\right)\right]\right] + H(q) \le \mathcal{L}(\Theta), \tag{4}$$

where $H(q) = \sum_n H(q_n(\vec{s})) = -\sum_n \sum_{\vec{s}} q_n(\vec{s}) \log(q_n(\vec{s}))$ is the Shannon entropy of q. The iterations of EM alternately increase $\mathcal{F}$ with respect to the distributions q_n while holding Θ fixed (the E-step), and with respect to Θ while holding the q_n fixed (the M-step). Thus, if we consider a pair of steps beginning from parameters Θ', the E-step first finds new distributions q_n that depend on Θ' and the observations $\vec{y}^{(n)}$, which we write as $q_n(\vec{s};\Theta')$. Ideally, these distributions maximize $\mathcal{F}$ for fixed Θ', in which case it can be shown that $q_n(\vec{s};\Theta') = p(\vec{s}\,|\,\vec{y}^{(n)},\Theta')$ and $\mathcal{F}(\Theta', q_n(\vec{s};\Theta')) = \mathcal{L}(\Theta')$ (Neal and Hinton, 1998). In practice, computation of this exact posterior may be intractable, and it is often replaced by an approximation. After choosing the q_n's in the E-step, we maximize $\mathcal{F}$ with respect to Θ in the M-step while holding the q_n distributions fixed. Thus the free-energy can be re-written in terms of Θ and Θ':

$$\mathcal{F}(\Theta,\Theta') \;=\; \sum_{n=1}^{N}\left[\sum_{\vec{s}} \quad q_n(\vec{s};\Theta') \quad \left[\log\left(p(\vec{y}^{(n)}\,|\,\vec{s},\Theta)\right) + \log\left(p(\vec{s}\,|\,\Theta)\right)\right]\right] \;+\; H(\Theta')\,. \tag{5}$$

where $H(\Theta') = \sum_n H(q_n(\vec{s};\Theta'))$. A necessary condition to achieve this maximum with respect to $W_{id} \in \Theta$, is that (see Appendix A for details):

$$\frac{\partial}{\partial W_{id}}\mathcal{F}(\Theta,\Theta') = \sum_n \sum_{\vec{s}} q_n(\vec{s};\Theta') \left(\frac{\partial}{\partial W_{id}}\overline{W}_d(\vec{s},W)\right) \frac{y_d^{(n)} - \overline{W}_d(\vec{s},W)}{\overline{W}_d(\vec{s},W)} \;\overset{!}{=}\; 0\,. \tag{6}$$

Unfortunately, under the max-combination rule of Equation (3), $\overline{W}_d$ is not differentiable. Instead, we define a smooth function $\overline{W}_d^{\rho}$ that converges to $\overline{W}_d$ as ρ approaches infinity:

$$\overline{W}_d^{\rho}(\vec{s},W) \;:=\; \left(\sum_{h=1}^{H}(s_h W_{hd})^{\rho}\right)^{\frac{1}{\rho}} \;\Rightarrow\; \lim_{\rho\to\infty}\overline{W}_d^{\rho}(\vec{s},W) = \overline{W}_d(\vec{s},W), \tag{7}$$

and replace the derivative of $\overline{W}_d$ by the limiting value of the derivative of $\overline{W}_d^{\rho}$, which we write as $\mathcal{A}_{id}$ (see Appendix A for details):

$$\mathcal{A}_{id}(\vec{s},W) := \lim_{\rho\to\infty}\left(\frac{\partial}{\partial W_{id}}\overline{W}_d^{\rho}(\vec{s},W)\right) = \lim_{\rho\to\infty}\frac{s_i\,(W_{id})^{\rho}}{\sum_h s_h (W_{hd})^{\rho}}\,. \tag{8}$$

Armed with this definition, a rearrangement of the terms in (6) yields (see Appendix A):

$$W_{id} = \frac{\sum\limits_n \langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} y_d^{(n)}}{\sum\limits_n \langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n}}, \tag{9}$$

where $\langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n}$ is the expectation of $\mathcal{A}_{id}(\vec{s},W)$ under the distribution $q_n(\vec{s};\Theta')$:

$$\langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} = \sum_{\vec{s}} q_n(\vec{s};\Theta')\,\mathcal{A}_{id}(\vec{s},W)\,. \tag{10}$$

Equation (9) represents a set of non-linear equations (one for each W_{id}) that defines the necessary conditions for an optimum of $\mathcal{F}$ with respect to W. The equations do not represent straightforward update rules for W_{id} because the right-hand-side does not depend only on the old values $W' \in \Theta'$. They can, however, be used as fixed-point iteration equations, by simply evaluating the derivatives $\mathcal{A}_{id}$ at W' instead of W. Although there is no guarantee that these iterations converge, if they do converge the corresponding parameters must lie at a stationary point of the free-energy. Numerical experiments described later confirm that this fixed-point approach is, in fact, robust and convergent. Note that the denominator in (9) vanishes only if $q_n(\vec{s};\Theta')\,\mathcal{A}_{id}(\vec{s},W) = 0$ for all $\vec{s}$ and n (assuming positive weights), in which case (6) is already satisfied, and no update of W is required.

Thus far, we have not made explicit reference to the form of prior source distribution, and so the result of Equation (9) is independent of this choice. For our chosen Bernoulli distribution (1), the M-step is obtained by setting the derivative of $\mathcal{F}$ with respect to π_i to zero, giving (after rearrangement):

$$\pi_i \;=\; \frac{1}{N}\sum_n \langle s_i\rangle_{q_n}\,, \quad \text{with } \langle s_i\rangle_{q_n} = \sum_{\vec{s}} q_n(\vec{s};\Theta')\,s_i\,. \tag{11}$$

Parameter values that satisfy Equations (9) and (11), maximize the free-energy given the distributions $q_n = q_n(\vec{s};\Theta')$. As stated before, the optimum with respect to q (and therefore, exact optimization of the likelihood, since the optimal setting of q forces the free-energy bound to be tight) is obtained by setting the q_n to the posterior distributions:

$$q_n(\vec{s};\Theta') \,=\, p(\vec{s}\,|\,\vec{y}^{(n)},\Theta') \,=\, \frac{p(\vec{s},\vec{y}^{(n)}\,|\,\Theta')}{\sum_{\tilde{\vec{s}}} p(\,\tilde{\vec{s}}\,,\vec{y}^{(n)}\,|\,\Theta')}, \tag{12}$$

where $p(\vec{s},\vec{y}^{(n)}\,|\,\Theta') = p(\vec{s}\,|\,\vec{\pi}')\,p(\vec{y}^{(n)}\,|\,\vec{s},W')$, and with the latter distributions given by (1) and (2), respectively.

Equations (9) to (12) thus represent a complete set of update rules for maximizing the data likelihood under the generative model. The only approximation made to this point is to use the old values W' on the right-hand-side of the M-step equation in (9). We therefore refer to this set of updates as a pseudo-exact learning rule and call the algorithm they define MCA$_{\text{ex}}$, with the subscript for *exact*. We will see in numerical experiments that MCA$_{\text{ex}}$ does indeed maximize the likelihood.

4. E-Step Approximations

The computational cost of finding the exact sufficient statistics $\langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n}$, with q_n equal to the posterior probability (12), is intractable in general. It grows exponentially in the smaller of the number of hidden causes H, and the number of observed variables D (see Appendix B for details). A practical learning algorithm, then, must depend on finding a computationally tractable approximation to the true expectation. One approach, a form of variational method (Jordan et al., 1999), would be to optimize the q_n within a constrained class of distributions; for example, distributions that factor over the sources s_h. Unfortunately, this conventional factoring approach provides limited benefit here, as the form of $\mathcal{A}_{id}(\vec{s},W)$ resists straightforward evaluation of the expected value with respect to the individual sources. Instead, we base our approximations on an assumption of sparsity—that only a small number of active hidden sources is needed to explain any one observed data vector (note that sparsity here refers to the *number* of active hidden sources, rather than to their *proportion*). The resulting expressions relate to those that would be found by a variational optimization constrained to distributions that are sparse in the sense above, but are not identical. The relationship will be explored further in the Discussion.

To develop the sparse approximations, consider grouping the terms in the expected value of Equation (10) according to the number of active sources in the vector $\vec{s}$:

$$\begin{aligned}\langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} &= \sum_{\vec{s}} p(\vec{s}\,|\,\vec{y}^{(n)},\Theta')\,\mathcal{A}_{id}(\vec{s},W) \qquad (13)\\ &= \sum_a p(\vec{s}_a\,|\,\vec{y}^{(n)},\Theta')\,\mathcal{A}_{id}(\vec{s}_a,W) + \sum_{\substack{a,b\\a<b}} p(\vec{s}_{ab}\,|\,\vec{y}^{(n)},\Theta')\,\mathcal{A}_{id}(\vec{s}_{ab},W) + \sum_{\substack{a,b,c\\a<b<c}} \ldots,\end{aligned}$$

where $\vec{s}_a := (0,\ldots,0,1,0,\ldots,0)$ with only $s_a = 1$
$\vec{s}_{ab} := (0,\ldots,0,1,0,\ldots,0,1,0,\ldots,0)$ with only $s_a = 1$, $s_b = 1$, $a \neq b$,
and $\vec{s}_{abc}$ etc. are defined analogously.

Note that $\mathcal{A}_{id}(\vec{0},W) = 0$ because of (7) and (8). Now, each of the conditional probabilities $p(\vec{s}\,|\,\vec{y}^{(n)},\Theta')$ implicitly contains a similar sum over $\vec{s}$ for normalization:

$$p(\vec{s}\,|\,\vec{y}^{(n)},\Theta') = \frac{1}{\mathcal{Z}}\,p(\vec{s},\vec{y}^{(n)}\,|\,\Theta'), \qquad \mathcal{Z} := \sum_{\vec{s}} p(\vec{s},\vec{y}^{(n)}\,|\,\Theta'), \qquad (14)$$

and the terms of this sum may be grouped in the same way

$$\mathcal{Z} := p(\vec{0},\vec{y}^{(n)}\,|\,\Theta') + \sum_a p(\vec{s}_a,\vec{y}^{(n)}\,|\,\Theta') + \sum_{\substack{a,b\\a<b}} p(\vec{s}_{ab},\vec{y}^{(n)}\,|\,\Theta') + \sum_{\substack{a,b,c\\a<b<c}} p(\vec{s}_{abc},\vec{y}^{(n)}\,|\,\Theta') + \ldots.$$

Combining (13) and (14) yields:

$$\langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} = \qquad (15)$$
$$\frac{\sum_a p(\vec{s}_a,\vec{y}^{(n)}\,|\,\Theta')\,\mathcal{A}_{id}(\vec{s}_a,W) + \sum_{\substack{a,b\\a<b}} p(\vec{s}_{ab},\vec{y}^{(n)}\,|\,\Theta')\,\mathcal{A}_{id}(\vec{s}_{ab},W) + \ldots}{p(\vec{0},\vec{y}^{(n)}\,|\,\Theta') + \sum_a p(\vec{s}_a,\vec{y}^{(n)}\,|\,\Theta') + \sum_{\substack{a,b\\a<b}} p(\vec{s}_{ab},\vec{y}^{(n)}\,|\,\Theta') + \ldots}.$$

A similar grouping of terms is possible for the expectation $\langle s_h\rangle_{q_n}$.

If we now assume that the significant posterior probability mass will concentrate on vectors $\vec{s}$ with only a limited number of non-zero entries, the expanded sums in both numerator and denominator of (15) may be truncated without significant loss. The accuracy of the approximation depends both on the sparsity of the true generative process, and on the distance of the current model parameters (in the current EM iteration) from the true ones. In general, provided that the true process is indeed sparse, a truncated approximation will become more accurate as the estimated parameters approach their maximum likelihood values. The convergence properties and accuracy of algorithms based on this form of approximation will be tested numerically in Section 6.

Different choices of the truncation yield approximate algorithms with different properties. Two of these will be considered here.

4.1 MCA$_3$

In the first approximation, we truncate all but one of the sums that appear in the expansions of $\langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n}$ and $\langle s_i\rangle_{q_n}$ after the terms that include three active sources, while truncating the numerator of $\langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n}$ after the two-source terms (see Appendix C for details):

$$\langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} \approx \frac{\overline{\pi}_i \exp(I_i^{(n)}) + \sum_{c\,(c\neq i)} \overline{\pi}_i\overline{\pi}_c \exp(I_{ic}^{(n)})\,\mathcal{H}(W_{id}-W_{cd})}{1+\sum_h \overline{\pi}_h \exp(I_h^{(n)}) + \frac{1}{2}\sum_{\substack{a,b\\ a\neq b}} \overline{\pi}_a\overline{\pi}_b \exp(I_{ab}^{(n)}) + \frac{1}{6}\sum_{\substack{a,b,c\\ a\neq b\neq c}} \overline{\pi}_a\overline{\pi}_b\overline{\pi}_c \exp(I_{abc}^{(n)})} \tag{16}$$

$$\text{and} \qquad \langle s_i\rangle_{q_n} \approx \frac{\overline{\pi}_i \exp(I_i^{(n)}) + \sum_{c\,(c\neq i)} \overline{\pi}_i\overline{\pi}_c \exp(I_{ic}^{(n)}) + \frac{\alpha}{2}\sum_{b,c\,(b\neq c\neq i)} \overline{\pi}_i\overline{\pi}_b\overline{\pi}_c \exp(I_{ibc}^{(n)})}{1+\sum_h \overline{\pi}_h \exp(I_h^{(n)}) + \frac{1}{2}\sum_{\substack{a,b\\ a\neq b}} \overline{\pi}_a\overline{\pi}_b \exp(I_{ab}^{(n)}) + \frac{1}{6}\sum_{\substack{a,b,c\\ a\neq b\neq c}} \overline{\pi}_a\overline{\pi}_b\overline{\pi}_c \exp(I_{abc}^{(n)})}, \tag{17}$$

where

$$\begin{aligned}
\overline{\pi}_i &= \frac{\pi_i}{1-\pi_i}, & I_i^{(n)} &= \sum_d \Big(\log(W_{id})\,y_d^{(n)} - W_{id}\Big), \\
\tilde{W}_d^{ab} &= \max(W_{ad},W_{bd}), & I_{ab}^{(n)} &= \sum_d \Big(\log(\tilde{W}_d^{ab})\,y_d^{(n)} - \tilde{W}_d^{ab}\Big), \\
\tilde{W}_d^{abc} &= \max(W_{ad},W_{bd},W_{cd}), & I_{abc}^{(n)} &= \sum_d \Big(\log(\tilde{W}_d^{abc})\,y_d^{(n)} - \tilde{W}_d^{abc}\Big),
\end{aligned} \tag{18}$$

and where $\mathcal{H}(x) = 1$ for $x>0$; $\frac{1}{2}$ for $x=0$; 0 for $x<0$ is the Heaviside function. The above equations have been simplified by dividing both numerator and denominator by terms that do not depend on $\vec{s}$, for example, by $\prod_{i=1}^{H}(1-\pi_i)$ (see Appendix C). Approximations (16) and (17) are used in the fixed-point updates of Equations (9) and (11), where the parameters that appear on the right-hand-side are held at their current values. Thus all parameters that appear on the right-hand-side of the approximations take values in $\Theta' = (\vec{\pi}', W')$.

The early truncation of the numerator in (16) improves performance in experiments, partly by increasing competition between causes further, and partly by reducing the contribution of more complex data patterns that are better fit, given the current parameter settings, by three active sources than by two. By contrast, the three-source terms are kept in the numerator of (17). In this case, neglecting complex input patterns as in (16) would lead to greater errors in the estimated source activation probabilities π_i. Indeed, even while keeping these terms, π_i tend to be underestimated if the input data include many patterns with more than three active sources. To compensate, we introduce a factor of $\alpha > 1$ multiplying the three-source term in (17) (so that $\alpha = 1$ corresponds to the actual truncated sum), which is updated as described in Appendix C. This scheme yields good estimates of π_i, even if more than three sources are often active in the input data.

The M-step Equations (9) and (11) together with E-step approximations (16) and (17) represent a complete set of update equations for the MCA generative model. The computational cost of one parameter update grows polynomially in the total number of causes, with order H^3. The algorithm that is defined by these updates will therefore be referred to as MCA$_3$.

4.2 R-MCA$_2$

In the second place, we consider a restriction of the generative model in which (i) all s_h are distributed according to the same prior distribution with fixed parameter π; (ii) the weights W_{id} associated with each source variable i are constrained to sum to a constant C:

$$\forall i \in \{1,\ldots,H\}: \quad \pi_i = \pi \quad \text{and} \quad \sum_d W_{id} = C; \tag{19}$$

and (iii) on average, the influence of each hidden source is homogeneously covered by the other sources. This third restriction means that each non-zero generating weight W_{id}^{gen} associated with cause i can be covered by the same number of $W_{cd}^{\text{gen}} \geq W_{id}^{\text{gen}}$:

$$W_{id}^{\text{gen}} > 0 \quad \Rightarrow \quad \sum_{c \neq i} \mathcal{H}(W_{cd}^{\text{gen}} - W_{id}^{\text{gen}}) \approx b_i\,, \tag{20}$$

where $\mathcal{H}$ is the Heaviside function and b_i is the number of causes that can cover cause i. Figure 3 il-

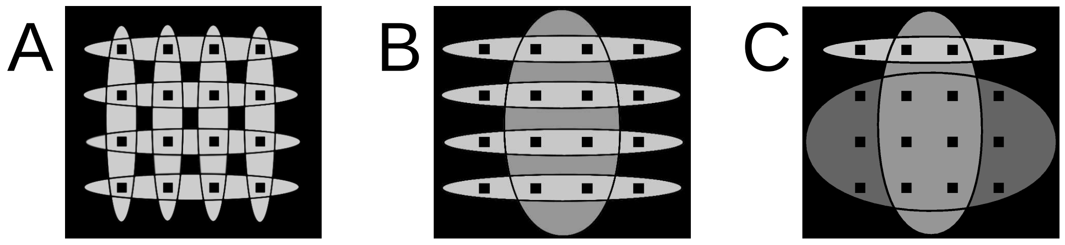

Figure 3: **A** and **B** show patterns of weights that satisfy the uniformity condition (20) whereas weights in **C** violate it. Each hidden cause is symbolized by an ellipse, with the gray-level of the ellipse representing the value W_{id} of each weight within the ellipse. Weights outside the ellipse for each cause are zero (black). The black squares indicate the 4-by-4 grid of observed pixels.

lustrates this condition. Figure 3A,B show weight patterns associated with hidden causes for which the condition is fulfilled; for instance in Figure 3B $b_i = 0$ for all causes with horizontal weight patterns, while $b_i = 1$ for the vertically oriented cause. In Figure 3C the condition is violated. Roughly, these conditions guarantee that all hidden causes have equal average effects on the generated data vectors. They make the development of a more efficient approximate learning algorithm possible but, despite their role in the derivation, the impact of these assumptions is limited in practice, in the sense that the resulting algorithm can perform well even when the input data set violates assumptions (19) and (20). This is demonstrated in a series of numerical experiments detailed below.

Update rules for the restricted generative model can again be derived by approximate expectation-maximization (see Appendix C). Using both the sum constraint of (19) and the assumption of homogeneous coverage of causes, we obtain the M-step update:

$$W_{id} \;=\; C\,\frac{\sum_n \langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} y_d^{(n)}}{\sum_{d'}\sum_n \langle \mathcal{A}_{id'}(\vec{s},W)\rangle_{q_n} y_{d'}^{(n)}}\,. \tag{21}$$

Empirically, we find that the restricted parameter space of this model means that we can approximate the sufficient statistics $\langle \mathcal{A}_{id}(\vec{s},W) \rangle_{q_n}$ by a more severe truncation than before, now keeping two-source terms in the denominator, but only single-source terms in the numerator, of the expansion (15). This approximation, combined with the fact that any zero-valued observed patterns (i.e., those with $\sum_d y_d^{(n)} = 0$) do not affect the update rule (21) and so can be neglected, yields the expression (see Appendix C):

$$\langle \mathcal{A}_{id}(\vec{s},W) \rangle_{q_n} \approx \frac{\exp(I_i^{(n)})}{\sum_h \exp(I_h^{(n)}) + \frac{\bar{\pi}}{2} \sum_{\substack{a,b \\ a \neq b}} \exp(I_{ab}^{(n)})}, \quad \bar{\pi} := \frac{\pi}{1-\pi}, \tag{22}$$

with abbreviations given in (18). Equations (21) and (22) are update rules for the MCA generative model, subject to the conditions (19) and (20). They define an algorithm that we will refer to as R-MCA$_2$ with R for *restricted* and with *2* indicating a computational cost that grows quadratically with H.

5. Relation to Neural Networks

We now relate component extraction as learned within the MCA framework to that achieved by a family of artificial neural networks. Consider the network of Figure 4 which consists of D input variables (or *units*) with values $y_1, \ldots, y_D$ and H hidden units with values $g_1, \ldots, g_H$. An observation $\vec{y}$ is represented by the values (or *activities*) of the input units, which act through *connections* parameterized by $(\mathcal{W}_{id})$ to determine the activities of the hidden units through an *activation function* $g_i = g_i(\vec{y}, \mathcal{W})$. These parameters $(\mathcal{W}_{id})$ are known as the network (or synaptic) *weights*.

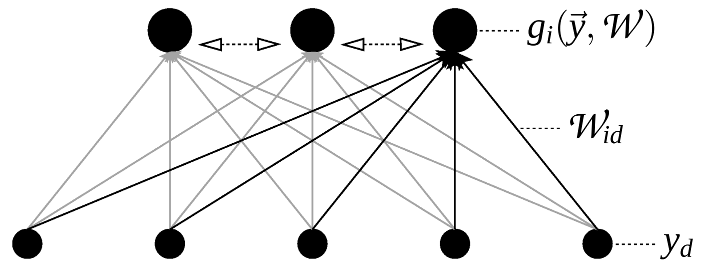

Figure 4: Architecture of a two layer neural network. Input is represented by values y_1 to y_D of D input units (small black circles). These values combine with synaptic weights $\mathcal{W}$ to determine the activities of the hidden units g_1 to g_H (big black circles). The dotted horizontal arrows symbolize lateral information exchange that may be required to compute the functions g_1 to g_H. After the g_i are computed the parameters $(\mathcal{W}_{id})$ are modified using a Δ-rule.

Learning in such a neural network involves adjusting the weights $\mathcal{W}$ in response to a series of input patterns, using a rule that is heuristically designed to extract some form of structure from these inputs. A standard choice is the Hebbian Δ-rule with divisive normalization:

$$\Delta \mathcal{W}_{id} \;=\; \varepsilon g_i(\vec{y}, \mathcal{W})\, y_d \quad \text{and} \quad \mathcal{W}_{id}^{\text{new}} = C\, \frac{\mathcal{W}_{id} + \Delta \mathcal{W}_{id}}{\sum_{d'} (\mathcal{W}_{id'} + \Delta \mathcal{W}_{id'})}\,, \tag{23}$$

The normalization step is needed to prevent weights from growing without bound, and the divisive form used here is most common. Here, the constant C defines the value at which $\sum_d \mathcal{W}_{id}$ is held constant; it will be related below to the C appearing in Equation 19. Many neural networks with the structure depicted in Figure 4, and that use a learning rule identical or similar to (23), have been shown to converge to weight values that identify clusters in, or extract useful components from, a set of input patterns (O'Reilly, 2001; Spratling and Johnson, 2002; Yuille and Geiger, 2003; Lücke and von der Malsburg, 2004; Lücke, 2004; Lücke and Bouecke, 2005; Spratling, 2006).

The update rule (23) depends on only one input pattern, and is usually applied *online*, with the weights being changed in response to each pattern in turn. If, instead, we consider the effect of presenting a group of patterns $\{\vec{y}^{(n)}\}$, the net change is approximately (see Appendix D):

$$\mathcal{W}_{id}^{\text{new}} \approx C\, \frac{\sum_n g_i(\vec{y}^{(n)}, \mathcal{W})\, y_d^{(n)}}{\sum_{d'} \sum_n g_i(\vec{y}^{(n)}, \mathcal{W})\, y_{d'}^{(n)}}\,. \tag{24}$$

Now, comparing (24) to (21), we see that if the activation function of a neural network were chosen so that $g_i(\vec{y}^{(n)}, \mathcal{W}) = \langle \mathcal{A}_{id}(\vec{s}, W) \rangle_{q_n}$, then the network would optimize the parameters of the restricted MCA generative model, with $W = \mathcal{W}$ (we drop the distinction between $\mathcal{W}$ and W from now on). Unfortunately, the expectation $\langle \mathcal{A}_{id}(\vec{s}, W) \rangle_{q_n}$ depends on d, and thus exact optimization in the general case would require a modified Hebbian rule. However, the truncated approximation of (22) is the same for all d, and so the changes in each weight depend only on the activities of the corresponding pre- and post-synaptic units. Thus, the Hebbian Δ-rule,

$$\Delta W_{id} \;=\; \varepsilon g_i\, y_d \quad \text{with} \quad g_i = \frac{\exp(I_i)}{\sum\limits_h \exp(I_h) + \frac{\overline{\pi}}{2} \sum\limits_{\substack{a,b \\ a \neq b}} \exp(I_{ab})} \tag{25}$$

(where I_h, I_{ab}, and $\overline{\pi}$ are the abbreviations introduced in Equations 18 and 22), when combined with divisive normalization, implements an online version of the R-MCA$_2$ algorithm. We refer to this online weight update rule as R-MCA$_{\text{NN}}$ (for Neural Network).

Note that the function g_i in (25) resembles the softmax function (see, e.g., Yuille and Geiger, 2003), but contains an additional term in the denominator. This added term reduces the change in weights when an input pattern results in more than one hidden unit with significant activity. That is, the system tries to explain a given input pattern using the current state of its model parameters W. If one hidden unit explains the input better than any combination of two units, that unit is modified. If the input is better explained by a combination of two units, the total learning rate is reduced.

Soft winner-take-all (WTA) activation functions, such as the softmax, are found in many networks that serve to both cluster *and* extract components from inputs, as appropriate. For clustering, the relationship between WTA-like competition and maximum-likelihood methods is well known (Nowlan, 1990). The connection drawn here offers a probabilistic account of the effectiveness of similar rules for component identification. If the probability of more than one cause being active is small (i.e., π is small), our activation rule for g_i (25) reduces to the standard softmax, suggesting that neural networks with activation and learning functions that resemble Equations (25) may perform well at both component extraction and clustering.

6. Experiments

The MCA generative model, along with the associated learning algorithms that have been introduced here, are designed to extract component features from non-linear mixtures. To study their performance, we employ numerical experiments, using artificial as well as more realistic data. The artificial data sets are based on a widely-used benchmark for non-linear component extraction, while the more realistic data are taken from acoustic recordings in one case and from natural images in the other. The goals of these experiments are (1) to establish whether the approximate algorithms do indeed increase the likelihood of the model parameters; (2) to test convergence and asymptotic accuracy of the algorithms; (3) to compare component extraction using MCA to other component-extraction algorithms; and (4) to demonstrate the applicability of the model and algorithms to more realistic data where non-linear component combinations arise naturally.

6.1 The Bars Test

The data sets used in experiments on artificial data were drawn from variants of the "bars test" introduced by Földiák (1990). Each data vector represents a grayscale image, with a non-linear combination of randomly chosen horizontal and vertical light-colored bars, each extending all the way across a black background. Most commonly, the intensity of the bars is uniform and equal, and the combination rule is such that overlapping regions remain at the same intensity. This type of data is a benchmark for the study of component extraction with non-linear interactions between hidden causes. Many component-extraction algorithms have been applied to a version of the bars test, including some with probabilistic generative semantics (Saund, 1995; Dayan and Zemel, 1995; Hinton et al., 1995; Hinton and Ghahramani, 1997), as well as many with non-generative objective functions (Harpur and Prager, 1996; Hochreiter and Schmidhuber, 1999; Lee and Seung, 2001; Hoyer, 2004) a substantial group of which have been neurally inspired (Földiák, 1990; Fyfe, 1997; O'Reilly, 2001; Charles et al., 2002; Spratling and Johnson, 2002; Lücke and von der Malsburg, 2004; Lücke and Bouecke, 2005; Spratling, 2006; Butko and Triesch, 2007).

In most of the experiments described here the input data were 25-dimensional vectors, representing a 5-by-5 grid of pixels; that is, $D = 5 \times 5$. There were b possible single bars, some of which were superimposed to create each image. On the 5-by-5 grid there are 5 possible horizontal, and 5 vertical, bar positions, so that $b = 10$. Each bar appears independently with a probability π, with areas of overlap retaining the same value as the individual bars. Figure 5A shows an example set of noisy data vectors constructed in this way.

6.2 Annealing

The likelihood surface for the MCA generative model is potentially multimodal. Thus, hill-climbing algorithms based on EM may converge to local optima in the likelihood, which may well be considerably poorer than the global optimum. This tendency to find sub-optimal fixed points can be reduced by incorporating a deterministic annealing, or relaxation, procedure (Ueda and Nakano, 1998; Sahani, 1999), whereby the entropy of the posterior distribution in the free energy (4) is artificially inflated in early iterations, with this inflation progressively reduced in later iterations, under the control of a temperature parameter. All of the experiments discussed here incorporated deterministic annealing, the details of which are given in Appendix E.

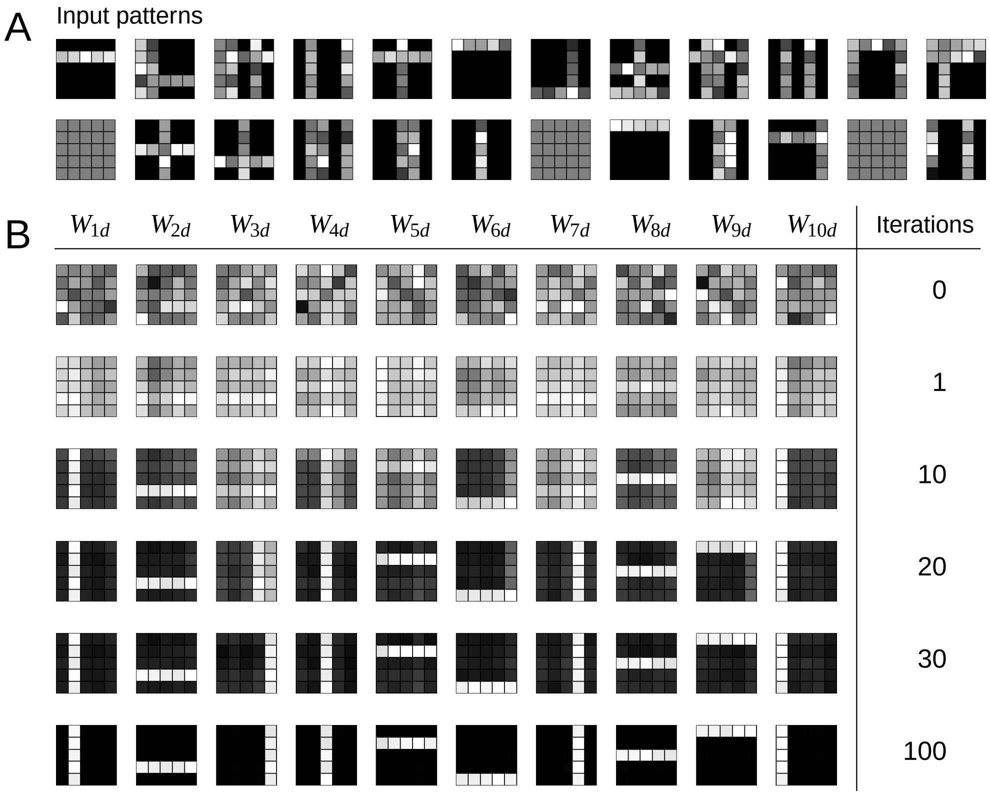

Figure 5: Bars test data with $b = 10$ bars on $D = 5 \times 5$ pixels and a bar appearance probability of $\pi = \frac{2}{10}$. **A** 24 patterns from the set of $N = 500$ input patterns that were generated according to the generative model with Poisson noise. **B** Change of the parameters W if MCA_3 is used for parameter update. Learning stopped automatically after 108 iterations in this trial (see Appendix E).

6.3 Convergence

From a theoretical standpoint, none of the four algorithms MCA_{ex}, MCA_3, R-MCA_2, or R-MCA_{NN}, can be guaranteed to maximize the likelihood of the MCA generative model. All of them update the parameters in the M-step using a fixed-point iteration, rather than either maximization or a gradient step. All but MCA_{ex} also approximate the posterior sufficient statistics (10). Thus, our first numerical experiments are designed to verify that the algorithms do, in fact, increase parameter likelihood in practice, and that they do converge. For this purpose, it is appropriate to use a version of the bars test in which observations are generated by the MCA model.

Thus, we selected MCA parameters that generated noisy bar-like images. There were 10 hidden sources in the generating model, one corresponding to each bar. The associated matrix of generating weights, W^{gen}, was 10×25, with each row representing a horizontal or vertical bar in a 5-by-5 pixel grid. The weights W_{id}^{gen} that correspond to the pixels of the bar were set to 10, the others to 0, so that $\sum_d W_{id}^{\text{gen}} = 50$. Each source was active with probability $\pi_i^{\text{gen}} = \frac{2}{10}$, leading to an average of two bars appearing in each image. We generated $N = 500$ input patterns (each with 25 elements) using Equations (1) to (3); a subset of the resulting patterns is displayed in Figure 5A.

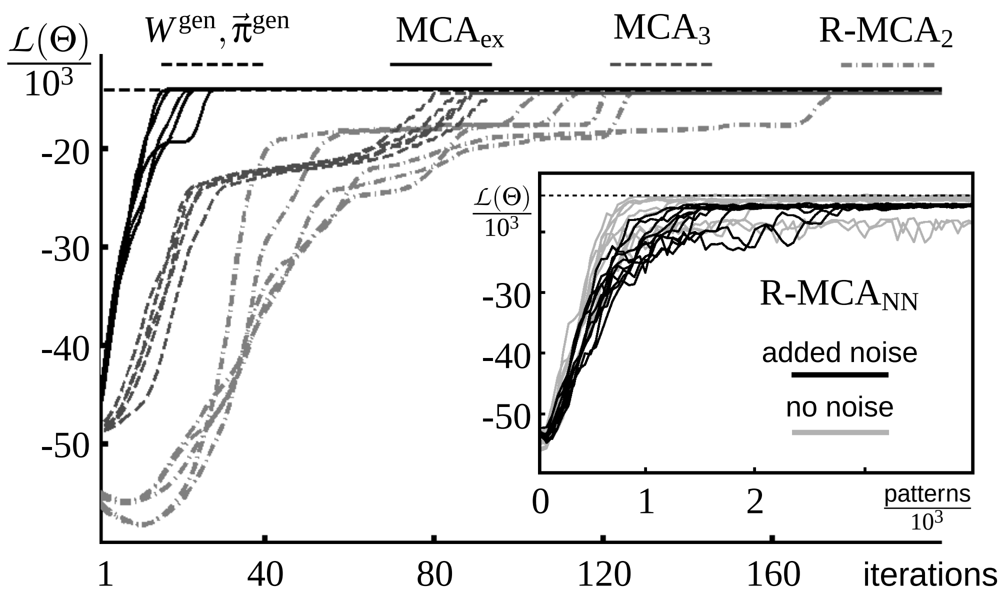

Figure 6: Change of the MCA parameter likelihood under different MCA learning algorithms. Data were generated as in Figure 5. To allow for comparison, the same set of $N = 500$ input patterns was used for all experiments shown. The likelihood of the generating parameters (W^{gen},$\vec{\pi}^{gen}$) is shown by the dotted horizontal line. The main axes show likelihood values of the batch-mode algorithms MCA$_{ex}$, MCA$_3$, and R-MCA$_2$ as a function of EM iteration. The inset axes shows likelihood values of the online algorithm R-MCA$_{NN}$ as a function of number of input pattern presentations. Patterns were randomly selected from the set of $N = 500$ inputs, and the parameters were updated for each pattern.

Figure 6 shows the evolution of parameter likelihoods, as a function of iteration, for each of the MCA algorithms, with 5 different choices of initial parameters for each. With the exception of the first few iterations of R-MCA$_2$, the likelihood of the parameters under the batch mode algorithms increased at almost every iteration. The online R-MCA$_{NN}$ showed greater fluctuations as updates based on individual data vectors inevitably perturbed the parameter estimates.

As might be expected, given the observation of increasing likelihoods and the fact that the likelihoods are bounded, each algorithm eventually converged from each initial value used in Figure 6. Furthermore, in each case, the likelihood of the solution found was close to the likelihood of the actual weights used in generation (the dashed horizontal lines). The final likelihood values for MCA$_{ex}$ were slightly higher than the likelihoods of (W^{gen}, $\vec{\pi}^{gen}$), as is expected for an exact maximum-likelihood algorithm in noisy data; whereas the values achieved by the approximations MCA$_3$ and R-MCA$_2$ were slightly lower. In fact, in 100 further experiments, the annealing and parameter initialization schemes described in Appendix E, brought the likelihood close to that of the generating weights in 98 of 100 runs using R-MCA$_2$ and in 90 of 100 runs using MCA$_3$. We did not run these more extensive tests for MCA$_{\text{ex}}$ due to its long running time (it is also omitted from similar quantitative analyses below).

The two basic observations, that likelihoods generally increased at each iteration and that the batch-mode algorithms all reliably converged, held true for all of the experiments described here and below, even where data were not generated from a version of the MCA model. Thus, we conclude that these algorithms are generally robust in practice, despite the absence of any theoretical guarantees.

6.4 Parameter Recovery

Figure 5B shows the evolution of parameters W, under the approximate MCA$_3$ algorithm, showing that the estimated W did indeed converge to values close to the generating parameters W^{gen}, as was suggested by the convergence of the likelihood to values close to that of the generative parameters. While not shown, the convergence of W under MCA$_{\text{ex}}$, R-MCA$_2$ or R-MCA$_{\text{NN}}$ was qualitatively similar to this sequence.

Clearly, if MCA$_{\text{ex}}$ finds the global optimum, we would expect the parameters found to be close to those used for generation. The same is not necessarily true of the approximate algorithms. However, both MCA$_3$ and R-MCA$_2$ did in fact find weights W that were very close to the generating values whenever an obviously poor local optimum was avoided.

In MCA$_3$ the average pixel intensity of a bar was estimated to be 10.0 ± 0.5 (standard deviation), taken across all bar pixels in 90 trials where the likelihood increased to a high value. Using R-MCA$_2$ this value was estimated to be 10.0 ± 0.8 (across all bar pixels on 98 high-likelihood trials). Note that the Poisson distribution (2) results in a considerable variance of bar pixel intensities around the mean of 10.0 (compare Figure 5A) which explains the high standard deviation around the relatively precise mean value. The background pixels (original value zero) are estimated to have an intensity of 0.05 ± 0.02 in MCA$_3$ and are all virtually zero (all are smaller than 10^{-56}) in R-MCA$_2$. MCA$_3$ also estimates the parameters $\vec{\pi}$. Because of the finite number of patterns ($N = 500$) we compared the estimates with the actual frequency of occurrence of each bar i: $\pi'_i = (\text{numb of bars } i \text{ in input})/N$. The mean absolute difference between the estimate π_i and the actual probability π'_i was 0.006 (across the 90 trials with high likelihood), which demonstrates the relative accuracy of the solutions, despite the approximation made in Equation (17).

For the neural network algorithm R-MCA$_{\text{NN}}$ given by (25) we observed virtually the same behavior as for R-MCA$_2$ when using a small learning rate (e.g., $\varepsilon = 0.1$) and the same cooling schedule in both cases (see Lücke and Sahani, 2007). The additional noise introduced by the online updates of R-MCA$_{\text{NN}}$ had only a negligible effect. For larger learning rates the situation was different, however. For later comparison to noisy neural network algorithms, we used a version of R-MCA$_{\text{NN}}$ with a relatively high learning rate of $\varepsilon = 1.0$. Furthermore, instead of a cooling schedule, we used a fixed temperature $T = 16$ and added Gaussian noise ($\sigma = 0.02$) at each parameter update: $\Delta W_{id} = \varepsilon g_i y_d + \sigma\eta$. With these learning parameters, R-MCA$_{\text{NN}}$ learned very rapidly, requiring fewer than 1000 pattern presentations in the majority of trials. Ten plots of likelihoods against number of presented patterns are shown for R-MCA$_{\text{NN}}$ in Figure 6 (inset figure, black lines) for the same $N = 500$ patterns as used for the batch-mode algorithms. Because of the additional noise in W, the final likelihood values were somewhat lower than those of the generating weights. Using R-MCA$_{\text{NN}}$ with the same parameters but without added noise ($\sigma = 0$), final likelihood values were often higher (inset axes, gray lines) but the algorithm also converged to local optima more often. In

Reliability		
Model	noisy	no noise
MCA_3	90%	81%
R-MCA_2	98%	96%
R-MCA_{NN}	>99%	>99%

Reliability		
Model	no noise	reference
noisy-or	27%	Saund, 1995
competitive	69%*	Dayan/Zemel, 1995
LOCOCODE	96%	Hochreiter/Schmidhuber, 1999

Table 1: Comparison of MCA algorithms with other systems in the standard bars test with $b = 10$ bars ($D = 5 \times 5, \pi = \frac{2}{10}, N = 500$). For the MCA algorithms reliability values are computed on the basis of 100 trials. Values for these algorithms are also given for the same bars test with Poisson noise. Reliability values for the other systems are taken from the literature. For instance, the model of Hochreiter and Schmidhuber (1999) is reported to fail to extract all bars in one of 25 trials. Two systems, back-propagation (BP) and GeneRec, that are described by O'Reilly (2001) have also been applied to this bars test. In their standard versions, BP and GeneRec achieve 10% and 60% reliability, respectively. Hochreiter and Schmidhuber (1999) report that ICA and PCA extract only subsets of all bars. *Trained without bar overlap.

contrast, R-MCA$_{\text{NN}}$ with noise avoided local optima in all 100 trials. In the following, R-MCA$_{\text{NN}}$ will therefore refer to the noisy version with $\sigma = 0.02$ unless otherwise stated.

6.5 Comparison to Other Algorithms—Noiseless Bars

To compare the component extraction results of MCA to that of other algorithms reported in the literature, we used a standard version of the bars benchmark test, in which the bars appear with no noise. The competing algorithms do not necessarily employ probabilistic semantics, and may not be explicitly generative; thus, we cannot compare performance in terms of likelihoods, nor in terms of the accuracy with which generative parameters are recovered. Instead, we adopt a commonly used measure, which asks how *reliably* all the different bars are identified (see, e.g., Hochreiter and Schmidhuber, 1999; O'Reilly, 2001; Spratling and Johnson, 2002; Lücke and von der Malsburg, 2004; Spratling, 2006). For each model, an internal variable (say the activities of the hidden units, or the posterior probabilities of each source being active) is identified as the response to an image. The responses evoked in the learned model by each of the possible single-bar images are then considered, and the most active unit or most probable source corresponding to each bar is identified. If the mapping from single-bar images to the most active internal variable is injective—that is, for each single bar a different hidden unit or source is the most active—then this instance of the model is said to have represented all of the bars. The reliability is the frequency with which each model represents all possible bars, when started from random initial conditions, and given a random set of images generated with the same parameter settings. For the MCA algorithms, the responses are defined to be the approximated posterior values for each possible source vector with only one active source, evaluated at the final parameter values after learning: $q(\vec{s}_h; \Theta) \approx p(\vec{s}_h \,|\, \vec{y}^{\text{bar}}, W)$.

The reliabilities of MCA$_3$, R-MCA$_2$, and R-MCA$_{\text{NN}}$ as well as some other published component-extraction algorithms are shown in Table 1. These experiments used a configuration of the bars test much as above ($D = 5 \times 5$, $b = 10$, and $\pi^{\text{gen}} = \frac{2}{10}$) which is perhaps the most commonly used in the literature, (e.g., Saund, 1995; Dayan and Zemel, 1995; Hochreiter and Schmidhuber, 1999; O'Reilly, 2001). The bars have a fixed and equal gray-value. We generated $N = 500$ patterns according to these settings and normalized the input patterns $\vec{y}^{(n)}$ to lie in the interval $[0,10]$ (i.e., bar pixels have a value of 10 and the background is 0). We considered both the case with Poisson noise (which has been discussed above) and the standard noiseless case. Experiments were run starting from 100 different randomly initialized parameters W. The same algorithms and the same cooling schedule were used (the same fixed T in the case of R-MCA$_{\text{NN}}$) to fit patterns with and without noise.

Without noise, MCA$_3$ with $H = 10$ hidden variables found all 10 bars in 81 of 100 experiments. R-MCA$_2$ with $H = 10$ found all bars in 96 of 100 experiments. Using the criterion of reliability, R-MCA$_{\text{NN}}$ performed best and found all bars in all 100 of 100 experiments. This seems likely to result from the fact that the added Gaussian noise, as well as noise introduced by the online updates, combined to drive the system out of shallow optima. Furthermore, R-MCA$_{\text{NN}}$ was, on average, faster than MCA$_3$ and R-MCA$_2$ in terms of required pattern presentations. It took fewer than 1000 pattern presentations to find all bars in the majority of 100 experiments,[1] although in a few trials learning did take much longer.

On the other hand, MCA$_3$ and R-MCA$_2$ achieved better likelihoods and recovered generative parameters closer to the true values. These algorithms also have the advantage of a well defined stopping criterion. MCA$_3$ learns the parameters of the prior distribution whereas R-MCA$_2$ uses a fixed value. R-MCA$_2$ does, however, remain highly reliable, even when the fixed parameter π differs significantly from the true value π^{gen}.

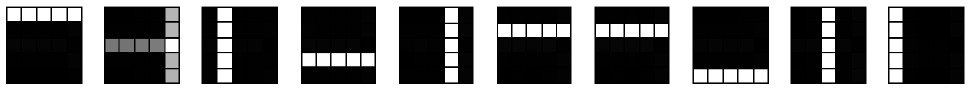

Figure 7: A common local optimum found by MCA$_3$ in the standard bars test. Two weight patterns reflect the same hidden cause, while another represents the superposition of two causes.

As was the case for the noisy bars, the R-MCA algorithms avoided local optima more often. This may well be a result of the smaller parameter space associated with the restricted model. A common local optimum for MCA$_3$ is displayed in Figure 7, where the weights associated with two sources generate the same horizontal bar, while a third source generates a weaker combination of two bars. This local solution is suboptimal, but the fact that MCA$_3$ has parameters to represent varying probabilities for each cause being present, means that it can adjust the corresponding rates to match the data. The fixed setting of π for R-MCA would introduce a further likelihood penalty for this solution.

Many component-extraction algorithms—particularly those based on artificial neural networks—use models with more hidden elements than there are distinct causes in the input data (e.g., Charles et al., 2002; Lücke and von der Malsburg, 2004; Spratling, 2006). If we use $H = 12$ hidden variables, then all the MCA-algorithms (MCA$_3$, R-MCA$_2$, and R-MCA$_{\text{NN}}$) found all of the bars in all of 100 trials.

1. Note that, according to the definition above, all bars are often already represented at intermediate likelihood values.

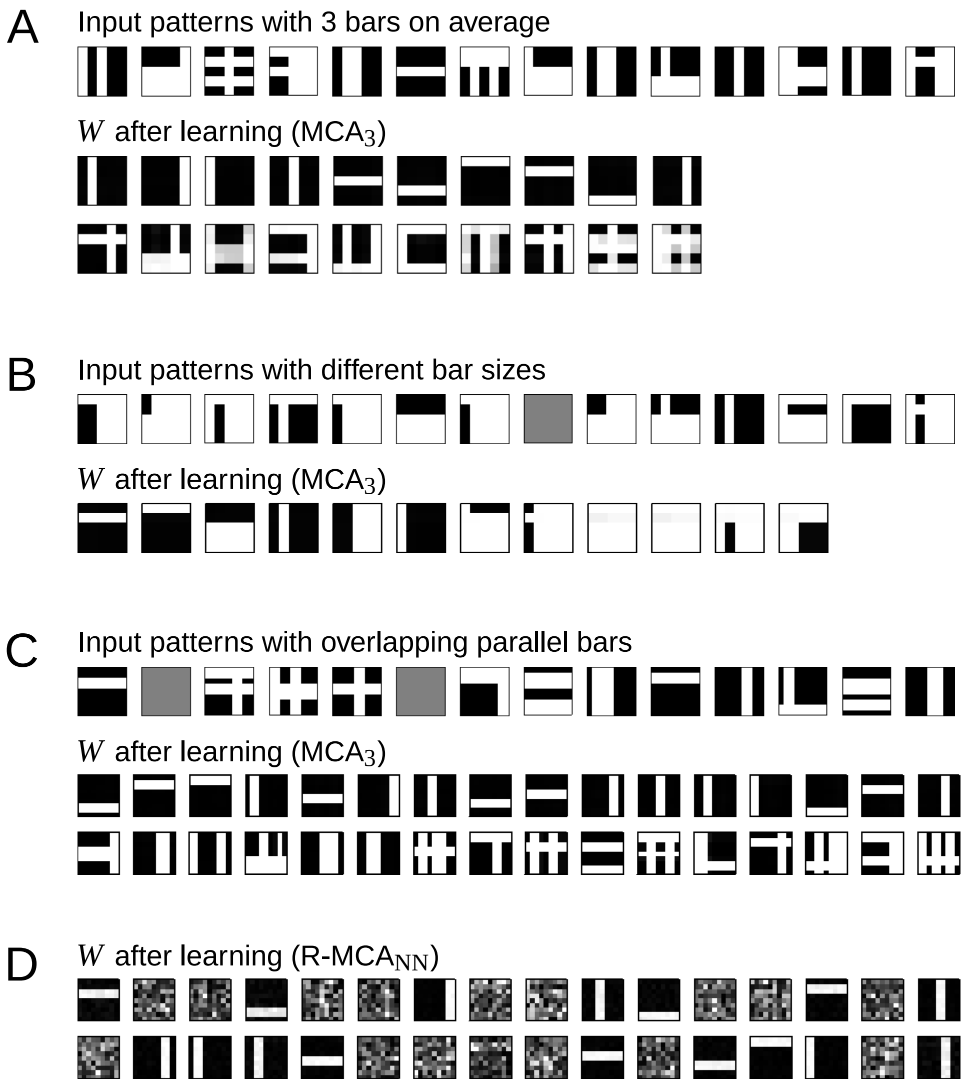
A
Input patterns with 3 bars on average
W after learning (MCA3)
B
Input patterns with different bar sizes
W after learning (MCA3)
C
Input patterns with overlapping parallel bars
W after learning (MCA3)
D
W after learning (R-MCANN)

Figure 8: Experiments with increased bar overlap. In **A** bar overlap is increased by increasing the bar appearance probability to $\pi^{\text{gen}} = \frac{3}{10}$ (an average of three bars per pattern). In **B** bar overlap is varied using different bar widths (two one-pixel-wide bars and one three-pixel-wide bar for each orientation). In the bars test in **C** there are 8 (two-pixel-wide) horizontal bars and 8 (two-pixel-wide) vertical bars on a $D = 9 \times 9$ pixel grid. Each bar appears with probability $\pi^{\text{gen}} = \frac{2}{16}$ (two bars per input pattern on average). Each experimental data set is illustrated by 14 typical input patterns. For **A** to **C** the parameters W of a typical trial are shown if MCA$_3$ is used for learning. The vectors $\vec{W}_i = (W_{i1}, \ldots, W_{iD})$ appear in order of decreasing learned appearance probability π_i. In **D** the parameters W for a typical trial using R-MCA$_{\text{NN}}$ are shown.

6.6 Comparison to Other Algorithms—Bar Overlap

For most component-extraction algorithms that have been tested against the bars benchmark, it is difficult to know how specialized they are to the form of this test. The algorithms might, for example, depend on the fact that all bars appear with the same probability, or that they have the same width. Different versions of the bars test have therefore been introduced to probe how generally the different algorithms might succeed. In particular, there has been considerable recent interest in studying robustness to varying degrees of overlap between bars (see, e.g., Lücke and von der Malsburg, 2004; Lücke, 2004; Spratling, 2006). This is because it is the non-linear combination within the regions of overlap that most distinguishes the bars test images from linear superpositions of sources. In three different experiments we varied the degree of overlap in three different ways. Following Spratling (2006), in all experiments the MCA model had twice as many possible sources as there were bars in the generative input. In all experiments we used the same algorithms, initial conditions, and cooling schedules as described above and in Appendix E. Again, each trial used a newly generated set of training patterns and a different randomly generated matrix W. In the following, reliability values are computed on the basis of 25 trials each.

The most straightforward way to increase the degree of bar overlap is to use the standard bars test with an average of three instead of two bars per image, that is, take $\pi = \frac{3}{10}$ for an otherwise unchanged bars test with $b = 10$ bars on $D = 5 \times 5$ pixels (see Figure 8A for some examples). When using $H = 20$ hidden variables, MCA$_3$ extracted all bars in 92% of 25 experiments. Thus the algorithm works well even for relatively high degrees of superposition. The values of W found in a typical trial are shown in Figure 8A. The parameters $\vec{W}_i = (W_{i1}, \ldots, W_{iD})$ that are associated with a hidden variable or unit are sorted according to the learned appearance probabilities π_i. Like MCA$_3$, both R-MCA$_2$ and R-MCA$_{\text{NN}}$ were run without changing any parameters. In the restricted case, this meant that the assumed value for the source probability ($\pi = \frac{2}{10}$) was different from the generating value ($\pi^{\text{gen}} = \frac{3}{10}$). Nevertheless, the performance of both algorithms remained better than that of MCA$_3$, with R-MCA$_2$ and R-MCA$_{\text{NN}}$ finding all 10 bars in 96% and 100% of 25 trials, respectively.

We can also choose unequal bar appearance probabilities (cf., Lücke and von der Malsburg, 2004). For example, half the bars appeared with probability $\pi_h^{\text{gen}} = (1+\gamma)\,\frac{2}{10}$ and the other half[2] appeared with probability $\pi_h^{\text{gen}} = (1-\gamma)\,\frac{2}{10}$, MCA$_3$ extracted all bars in all of 25 experiments for $\gamma = 0.5$. For $\gamma = 0.6$ (when half the bars appeared 4 times more often than the other half) all bars were extracted in 88% of 25 experiments. For $\gamma = 0.6$ R-MCA$_2$ and R-MCA$_{\text{NN}}$ found all bars in 96% and 100% of 25 experiments respectively. Reliability values for R-MCA$_{\text{NN}}$ started to decrease for $\gamma = 0.7$ (92% reliability).

As suggested by Lücke and von der Malsburg (2004), we also varied the bar overlap in a second experiment by choosing bars of different widths. For each orientation we used two one-pixel wide bars and one three-pixel-wide bar. Thus, for this data set, $b = 6$ and $D = 5 \times 5$. The bar appearance probability was $\pi = \frac{2}{6}$, so that an input contained, as usual, two bars on average. Figure 8B shows some examples. MCA$_3$ extracted all bars in 84% of 25 experiments for this test. Reliability values decreased for more extreme differences in the bar sizes. R-MCA$_2$ and R-MCA$_{\text{NN}}$ both found all bars in all 25 trials each. Thus, although the unequal bar sizes violated the assumption $\sum_d W_{id} = C$ that was made in the derivation of R-MCA$_2$ and R-MCA$_{\text{NN}}$, the algorithms' performance in terms of reliability seemed unaffected.

2. If bars are numbered $h = 1$ to 5 for the horizontal and $h = 6$ to 10 for the vertical, we chose the ones with even numbers to appear with the higher probability.

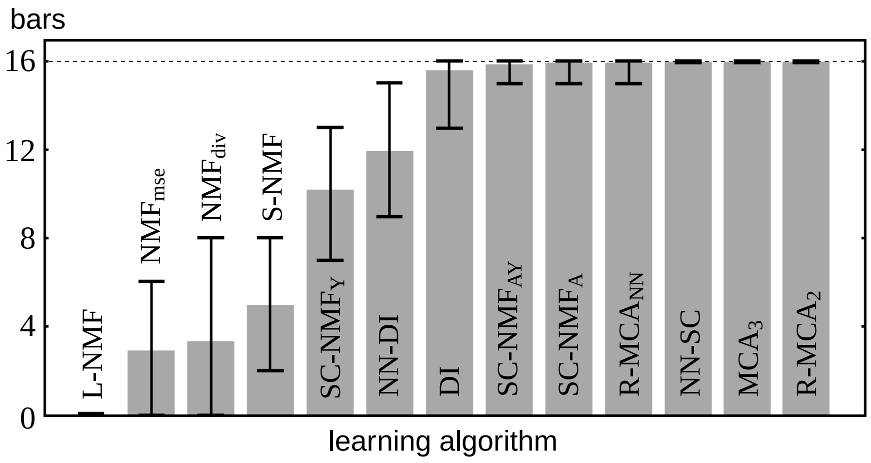

Figure 9: Comparison of MCA$_3$, R-MCA$_2$, and R-MCA$_{NN}$ with other systems in the bars test with increased occlusion (compare Figure 8C and Figure 2). Bars test parameters are $D = 9 \times 9$, $b = 16$, $\pi = \frac{2}{16}$, and $N = 400$. Data for the non-MCA algorithms are taken from Spratling (2006). The bar heights represent the average numbers of extracted bars in 25 trials. Error bars indicate the largest and the lowest number of bars found in a trial. The algorithms NN-DI and DI are feed-forward neural networks of the type depicted in Figure 4. All other (non-MCA) algorithms are versions of NMF with different objectives and constraints (see Appendix E and Spratling, 2006, for details).

In the third experiment we changed the degree of bar overlap more substantially, using a bars test that included overlapping parallel bars as introduced by Lücke (2004). We used eight horizontal and eight vertical bars, each two pixels wide, on a 9-by-9 grid. Thus, two parallel neighboring bars had substantial overlap. Figure 8C shows some typical input patterns. Note that the introductory example of Figure 2A,B is also of this type. To allow for a detailed comparison with other systems we adopted the exact settings used by Spratling (2006), that is, we considered 25 runs of a system with $H = 32$ hidden variables using bars test parameters $D = 9 \times 9$, $\pi^{\text{gen}} = \frac{2}{16}$, and $N = 400$. For these data, MCA$_3$ found all 16 bars in all of 25 experiments. The same is true for R-MCA$_2$ whereas R-MCA$_{NN}$ missed one bar in one of the 25 trials. Figure 9 shows a quantitative comparison with other algorithms that have been applied to this version of the bars test. Of the ten algorithms studied by Spratling (2006) just one, namely non-negative sparse coding (NN-SC; Hoyer, 2002, with sparseness parameter $\lambda = 1$), is as reliable as MCA$_3$ and R-MCA$_2$. The other systems, including forms of NMF both with and without a sparseness constraint, fail partly or entirely in extracting the actual hidden causes. For a typical trial using MCA$_3$ the final parameters W are displayed in Figure 8C. Again the $\vec{W}_i$'s associated with the different hidden variables are sorted according to their learned parameters π_i. A qualitatively different set of $\vec{W}_i$'s was obtained when R-MCA$_{NN}$ was used for learning. Figure 8D shows a typical outcome ($\vec{W}_i$'s are not sorted). In this case, only the actual causes are clearly represented whereas the $\vec{W}_i$'s of the supernumerary units remain unspecialized. The same feature is reported by Spratling (2006) for the algorithms NN-DI and DI used in this same test. Convergence to a representation that contains just the true hidden causes and leaves supernumerary units unspecialized can improve the interpretability of the result. When using a higher fixed temperature for R-MCA$_{NN}$ all the hidden units represented bars, with some bars represented by more than one unit. However, hidden units that represented more composite inputs, as seen for MCA$_3$, were rarely observed. On the other hand, the parameters found by MCA$_3$ provide an indication of significance of each weight pattern in the appearance probabilities π_i. Thus, in Figure 8C the appearance probabilities for the first 16 sources are much higher than for the others. The later sources may be interpreted as capturing some of the higher-order structure that results from a finite set of input patterns. In contrast to R-MCA, such higher-order representations need not adversely affect the data likelihood because the corresponding appearance probabilities can be relatively small.

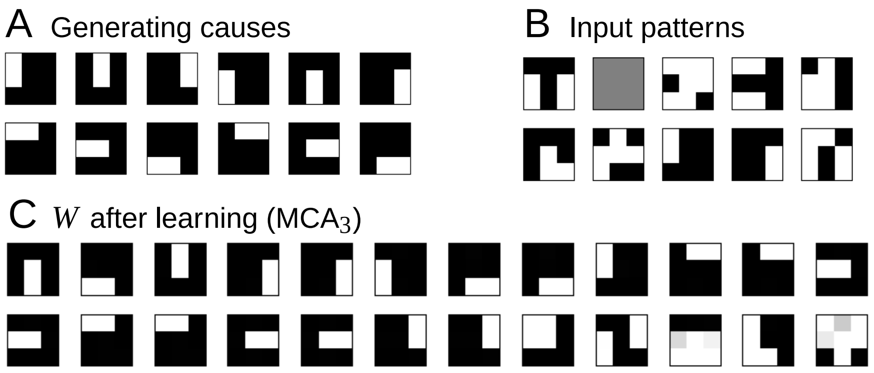

Figure 10: Experiments with more causes and hidden variables than observed variables. **A** The 12 patterns used to generate the data. Each is a 1-by-2 pixel bar on a 3-by-3 grid ($D = 9$). **B** Ten examples of the 500 input patterns generated using the causes shown in **A**. **C** Parameters W found in a typical run of MCA$_3$ with $H = 24$. The vectors $\vec{W}_i = (W_{i1}, \ldots, W_{iD})$ appear in order of decreasing learned appearance probability π_i.

6.7 More Causes than Observed Variables

In the experiments described above, the number of hidden causes was always smaller than the number of observed variables. We next briefly studied the “over-complete” case where data were generated, and models were fit, using more hidden causes than observed variables. We generated $N = 500$ patterns on a 3-by-3 grid ($D = 9$), using sparse combinations of 12 hidden causes corresponding to 6 horizontal and 6 vertical bars, each 1-by-2 pixels in size and thus extending across only a portion of the image (Figure 10A). As in the bars tests above, black was assigned to a value of 0 and white to 10. Patterns were generated without noise, with an average of two bars appearing in each ($\pi = \frac{2}{12}$). Ten such patterns are shown in Figure 10B.

Figure 10C shows the weights learned during a typical run using MCA$_3$ with the same parameter settings as above and twice as many hidden variables than observed ones ($H = 24$). Weights are sorted in order of decreasing inferred appearance probabilities π_i. All 12 causes were identified, with many represented more than once. A few hidden variables, with lower inferred probabilities of appearance, were associated with more composite patterns. MCA$_3$ extracted all causes in all of 25 trials. R-MCA$_2$ also extracted all causes in all of 25 trials, and never represented composite patterns. R-MCA$_{NN}$ only extracted all causes when run at fixed temperatures that were lower than those used for the bars tests above (e.g., $T = 3$), in which case it did so in all of 25 trials. This requirement for a lower temperature was consistent with the observation that a lower data dimension D leads to a decrease in the critical temperatures associated with the algorithms (see Appendix E). For larger values of T (e.g., $T = 16$) R-MCA$_{NN}$ did not extract single causes.

6.8 Violations of Model Assumptions

To optimize the likelihood of the data under the MCA generative model, each of the approximate learning algorithms relies on the fact that, under the Bernoulli prior (1), some number of the observed data vectors will be generated by only a small number of active sources. To highlight this point we explicitly removed such sparse data vectors from a standard bars test, thereby violating the Bernoulli prior assumption of the generative model. We used bars tests as described above, with $b = 10$ or $b = 16$ bars and $\pi = \frac{2}{b}$, generating $N = 500$ (or more) patterns, in each case by first drawing causes from the Bernoulli distribution (1) and then rejecting patterns in which fewer than m causes were active. As might be expected, when m was 3 or greater the approximate algorithms all failed to learn the weights associated with single causes. However, when only patterns with fewer than 2 bars had been removed, MCA$_3$ was still able to identify all the bars in many of the runs. More precisely, using data generated as above with $b = 10$, $m = 2$ and $N = 500$, MCA$_3$ with $H = 10$ hidden variables found all causes in 69 of 100 trials with noisy observations and in 37 of 100 trials without noise (the parameters for MCA$_3$ and the associated annealing schedule were unchanged). Note that in these experiments the average number of active causes per input vector is increased by the removal of sparse data vectors. An increase in reliability in the noisy case is consistent with our other experiments. The relatively low reliability seen for noiseless bars in this experiment may be due to the combined violation of both the assumed prior and noise distributions.

As long as the data set did contain some vectors generated by few sources, the learning algorithms could all relatively robustly identify the causes given sufficient data, even when the average observation contained many active sources. For instance, in a standard noiseless bars test with $b = 16$ bars on an 8×8 grid, and $N = 1000$ patterns with an average of four active causes in each ($\pi = \frac{4}{16}$), all three algorithms still achieved high reliability values, using twice as many hidden variables as actual bars ($H = 32$), and using the same parameters as for the standard bars test above. MCA$_3$ found all causes in 20 of 25 trials in these data (80% reliability). Reliabilities of R-MCA$_2$ and R-MCA$_{NN}$ (25 trials each) were 76% and 100%, respectively. The reliabilities of all algorithms fell when the data set contained fewer patterns, or when the average number of bars per pattern was larger.

6.9 Applications to More Realistic Data

We study two examples of component extraction in more realistic settings, applying the MCA algorithms to both acoustic and image data.

Acoustic data. Sound waveforms from multiple different sources combine linearly, and so are conventionally unmixed using algorithms such as ICA applied to simultaneous recordings from multiple microphones. The situation is different, however, for *spectrogram* representations of natural

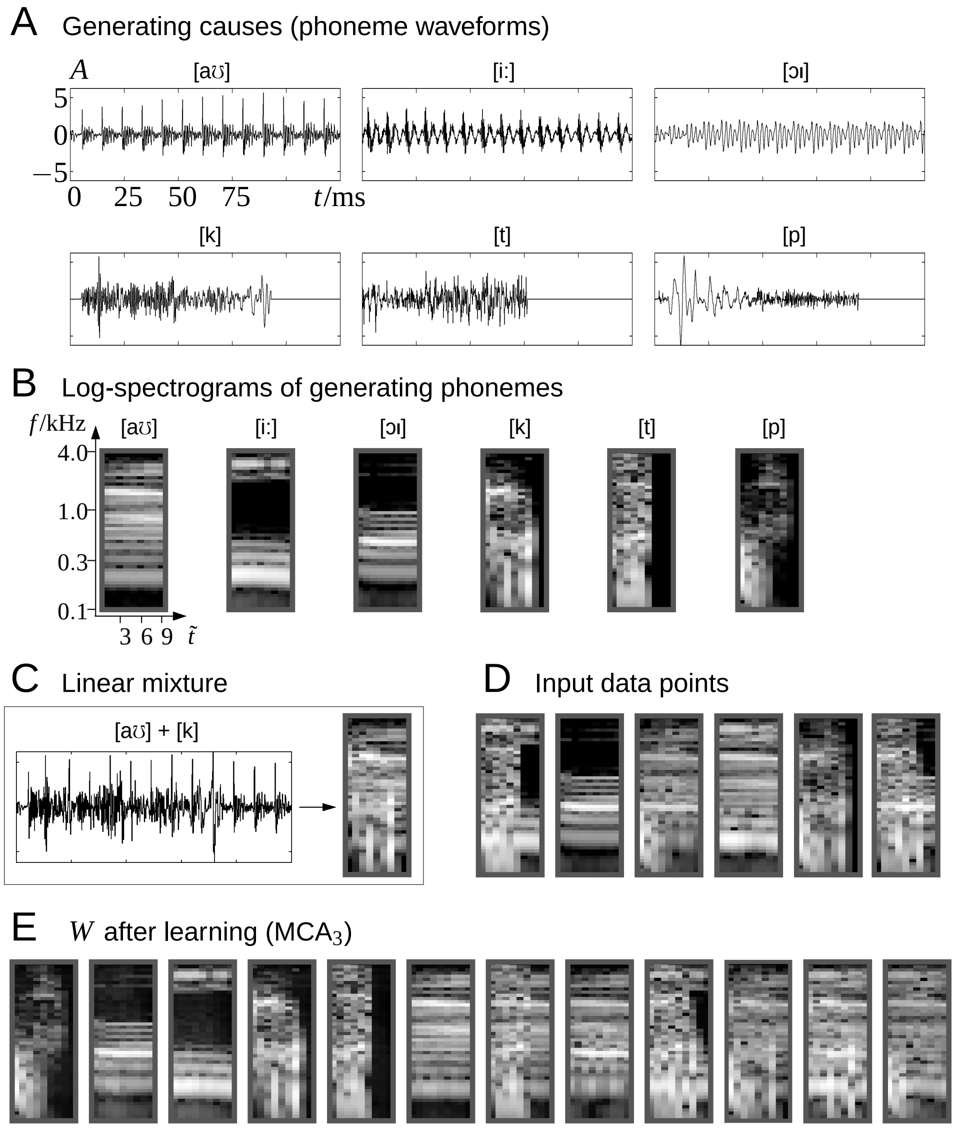

Figure 11: Application to acoustic data. **A** Pressure waveforms of six phonemes spoken by a male voice. Axes here, and for the waveform in **C**, are as shown for [aʊ] (A is a normalized amplitude). **B** The log-spectrograms of the phonemes in **A**. We use 50 frequency channels and nine time windows ($\tilde{t} = 1, \ldots, 9$). Axes of all log-spectrograms in the figure are as shown for [aʊ]. **C** Waveform of the linear mixture of phonemes [aʊ] and [k], and the log-spectrogram of this linear mixture. **D** Six examples of the $N = 500$ data vectors that were used for the experiments. Each data vector is the log-spectrogram of a linear mixture of the phoneme waveforms in **A**. The data sets for the experiments used an average of two waveforms per data vector. **E** Parameters W found by MCA$_3$ with $H = 12$, using 500 mixture log-spectrograms. The parameter vectors $\vec{W}_i = (W_{i1}, \ldots, W_{iD})$ appear in order of decreasing learned appearance probability π_i and are linearly scaled to fill the gray scale.

sound. The power of natural sounds in individual time-frequency bins varies over many orders of magnitude, and so is typically measured logarithmically and expressed in units of decibels, giving a representation that is closely aligned with the response of the cochlea to the corresponding sound. In this representation, the combination of log-spectrograms of the different sources may be well approximated by the max rule (R. K. Moore, 1983, quoted by Roweis, 2003). In particular, the logarithmic power distribution, as well as the sub-linear power summation due to phase misalignment, both lead to the total power in a time-frequency bin being dominated by the single largest contribution to that bin (see Discussion).

To study the extraction of components from mixtures of sound by MCA, we based the following experiment on six recordings of phonemes spoken by a male voice (see Figure 11A). The phoneme waveforms were mixed *linearly* to generate $N = 500$ superpositions, with each phoneme appearing in each mixture with probability $\pi = \frac{2}{6}$. Thus each mixture comprised two phonemes on average, with a combination rule that resembled the MCA max-rule in the approximate sense described above.

We applied the MCA algorithms to the log-spectrograms of these mixtures. Figure 11B shows the log-spectrograms of the individual phonemes and Figure 11C shows the log-spectrogram of an example phoneme mixture. We used 50 frequency channels and 9 time bins to construct the log-spectrograms. The resulting values were thresholded and then rescaled linearly so that power-levels across all phonemes filled the interval $[0,10]$, as in the standard bars test. For more details see Appendix E.

The MCA algorithms were used with the same parameter settings as in the bars tests above, except that annealing began at a lower initial temperature (see Appendix E). As in the bars tests with increased overlap, we used twice as many hidden variables ($H = 12$) as there were causes in the input. Figure 11E shows the parameters W learned in one run using MCA$_3$. The parameter vectors $\vec{W}_i = (W_{i1}, \ldots, W_{iD})$ are displayed in decreasing order of the corresponding learned value of π_i. As can be seen, the first six such vectors converged to spectrogram representations similar to those of the six original phonemes. The six hidden variables associated with lower values of π_i, converged to weight vectors that represented more composite spectrograms. This result is representative of those found with MCA$_3$. R-MCA$_2$ also converged to single spectrogram representations, but tended to represent those single spectrograms multiple times rather than representing more composite patterns with the additional components. Results for R-MCA$_{\mathrm{NN}}$ were very similar to those for R-MCA$_2$ when we used a high fixed temperature (see Appendix E for details). For intermediate fixed temperatures, results for R-MCA$_{\mathrm{NN}}$ were similar to those of the bars test in Figure 8D in that each cause was represented just once, with additional hidden units displaying little structure in their weights. For lower fixed temperatures (starting from $T \approx 40$) R-MCA$_{\mathrm{NN}}$ failed to represent all causes.

In general, the reliability values of all three algorithms were high. These were measured as described for the bars tests above, by checking whether, after learning, inference based on each individual phoneme log-spectrogram led to a different hidden cause being most probable. MCA$_3$ found all causes in 21 of 25 trials (84% reliability), R-MCA$_2$ found all causes in all of 25 trials; as did R-MCA$_{\mathrm{NN}}$ (with fixed $T = 70$). Reliability for MCA$_3$ improved to 96% with a slower cooling procedure ($\theta_{\Delta W} = 0.25 \times 10^{-3}$; see Appendix E).

Visual data. Finally, we consider a data set for which the exact hidden sources and their mixing rule are unknown. The data were taken from a single 250-by-250 pixel gray-level image of grass taken

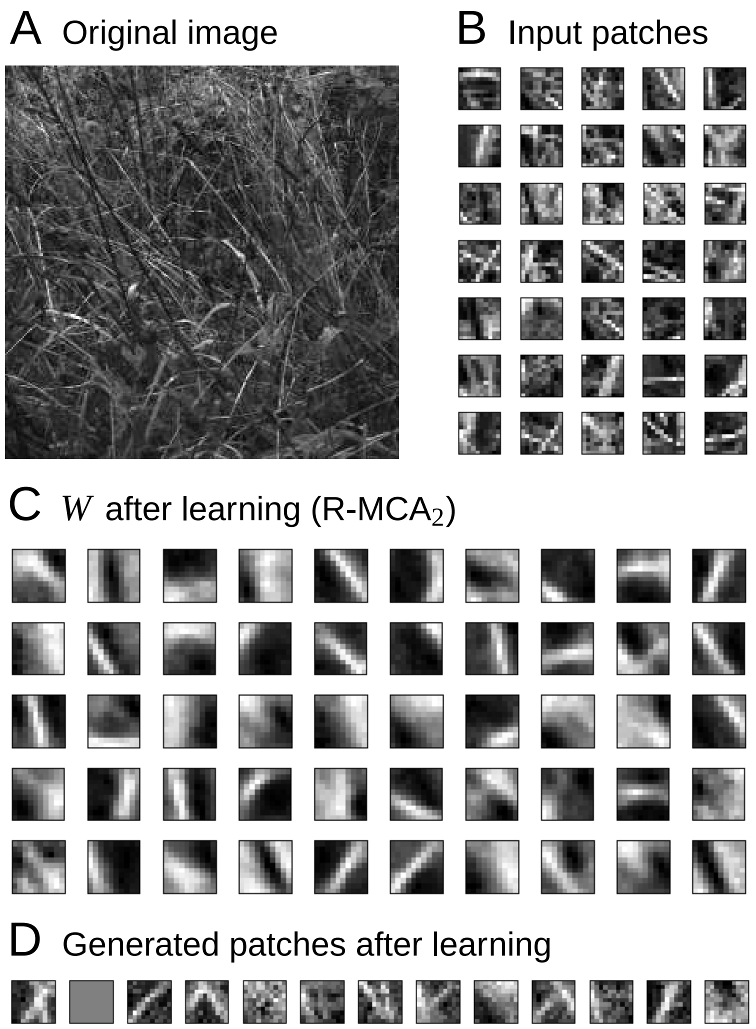
A Original image
B Input patches
C W after learning (R-MCA$_2$)
D Generated patches after learning

Figure 12: Application to visual data. **A** The 250-by-250 pixel image used as basis for the experiments. The image is taken from the van Hateren database of natural images (see Appendix E). For visualization we have brightened the image (we let values in the lower half of the light intensity range fill the range of gray values from zero to 255 and clamped values in the upper half to value 255). Without brightening, the image would appear unnaturally dark on a finite gray scale because of a small number of pixels with very high values. **B** 35 examples taken from the 5000 10-by-10 pixel patches that were used for numerical experiments. The patches represent light intensities linearly. For visualization, each patch has been scaled to fill the range of gray values. **C** Parameters W resulting from a typical run of R-MCA$_2$ with $H = 50$ hidden variables and $N = 5000$ image patches. For visualization, each parameter vector $\vec{W}_i = (W_{i1}, \ldots, W_{iD})$ has been linearly scaled to fill the range of gray values. **D** Patches generated using the restricted generative model and weights as in **C** (patches have been scaled as in **B** and **C**).

from the van Hateren database (Figure 12A) and linearly rescaled so that pixel intensities filled the interval $[0, 10]$. Each data vector was a 10-by-10 pixel patch drawn from a random position in the image (see Figure 12B for some examples).

The image comprised stems and blades of grass which occluded each other. As discussed above (see, e.g., Figure 2), the combination rule for such objects may be well approximated by the max rule of the MCA generative model (at least for the lighting conditions that appear to prevail in Figure 12A). Thus, the MCA learning algorithms may be expected to converge to parameters W that represent intensity images of 'grass'-like object parts. However, each blade of grass might appear at many different positions within the image patches, rather than at a fixed set of possible locations as in the bars test. Thus to recover these grass-like elements in the MCA causal weights requires the use of models with large numbers of hidden variables (and, correspondingly, many data vectors). For the number of patches and hidden variables required, the cubic cost of MCA$_3$ led to impractically long execution times. In experiments with smaller patch sizes and small H (e.g., $H = 10$ or $H = 20$) some weight patterns did converge to represent 'grass'-like objects, but many converged to less structured configurations.

The computational cost of R-MCA$_2$ is smaller and we evaluated trials using $H = 50$ hidden variables and $N = 5000$ 10-by-10 patches. R-MCA$_2$ was used with the same parameter setting as for the bars tests above, except for lower initial and final temperatures for annealing (see Appendix E). Figure 12C shows a typical outcome obtained when cooling from $T = 4.0$ to $T = 1.0$. A large number of weight vectors have converged to represent 'grass'-like object parts, whereas others represent more extensive causes that might be interpreted as capturing background noise. Many of the weight patterns have an orientation similar to the dominant orientation in the original image. Figure 12D shows a selection of patches generated using the learned weights. We used a higher value of C during generation than during learning (the parameter is not learned with R-MCA$_2$), thus globally rescaling the learned weights, so as to reduce the apparent noise level. In experiments where annealing was terminated at $T = 1.5$ (as in the bars test), the resulting weights were generally similar to the ones in Figure 12C, but with a larger proportion of weight vectors showing little structure. Learning with slower annealing did not result in significantly different weights. With fewer than $N = 5000$ patches for training, the weight patterns were less smooth, presumably reflecting overfitting to the subset of data used.

In experiments applying the online algorithm R-MCA$_{\text{NN}}$ to a set of 5000 10-by-10 patches as above, we found that it would converge to 'grass'-like weight patterns provided the learning rate (ε in Equation 25) was set to a much lower value than had been used in the bars tests. A lower learning rate corresponds to effectively averaging over a much larger set of input patterns. With $\varepsilon = 0.02$ (instead of 1.0 as above), and with noise on the weights (σ) scaled down by the same factor, R-MCA$_{\text{NN}}$ converged to weights similar to those shown in Figure 12C (for R-MCA$_2$), although a larger number of hidden units showed relatively uniform weight structure. For R-MCA$_{\text{NN}}$ we used a fixed temperature of $T = 2.0$.

7. Discussion

We discuss the applicability of the MCA learning algorithms, and the generality of the MCA framework, before relating the new algorithms to previous methods and neural network systems.

7.1 Applicability of the Model

The MCA generative model and associated learning algorithms are designed to extract causal components from input data in which the components combine non-linearly. More precisely, the generative model assumes that the single active cause with the strongest influence on a particular observed variable alone determines its observed value—something we have referred to here as the max-rule for combination. This stands in contrast to other feature extraction models such as PCA, ICA, NMF, or SC, in which the influences of the different causes are summed.

One context in which data with a superposition property very close to the max-rule arise naturally is the psychoacoustic combination of sounds. The perception of sound is largely driven by the logarithm of the time-varying intensity within each of a bank of narrow-band frequency channels. The narrow-band, short-time intensity of natural sounds may vary over many orders of magnitude. Further, sounds from different sources may have unrelated phases, and so intensities within each channel will generally add sub-linearly. Thus, even though sound *waveforms* from different sources combine linearly, the time- and frequency-local *intensities*, expressed logarithmically (in decibels), are dominated by the loudest of the sounds within each time-frequency bin. Indeed, even if two sounds are of equal loudness, the intensity of the sum is greater than each of them by at most 3 dB. Here, then, the max-rule is a very good approximation to the true generative combination. This observation motivated our use of acoustic data in the experiments shown in Figure 11.

In the image domain, the max-rule's relevance comes from the fact that it matches the true occlusive combination rule more closely than does the more commonly used sum. This is true both quantitatively (see Figure 2 and the discussion thereof), and also qualitatively, in the sense that both occlusion and the max-rule share a property of exclusiveness—that is, only one of the hidden causes determines the value of each pixel. Numerical experiments on raw image data (Figure 12) demonstrate that plausible generative causes are extracted using the MCA approach. The weight patterns associated with the extracted causes resemble images of the single object parts (blades and stems of grass in our example) that combine non-linearly to generate the image. The MCA approach also holds some potential for component extraction in more low-level image processing, for example, if we assume that each input pixel is generated exclusively by one edge instead of a whole object or object part. The application of MCA might, however, be less straight-forward in this case and presumably requires image preprocessing and perhaps a different noise model.

7.2 Generality of the Framework

Many of the details of the algorithms presented here, as well as many of the experiments, have been based on a specific model in which the hidden variables are drawn from a multivariate Bernoulli distribution (1), and the observations are then Poisson, conditioned on these values (2). These choices are natural ones for non-negative data generated from binary sources (cf., NMF; Lee and Seung, 1999, 2001). However, while the details have largely been omitted for brevity, it is straightforward to incorporate alternative generative distributions within the same framework, and with the same approximations.

Thus, the equations that define the M-step (9), as well as the expansion (15) used to approximate the E-step, would hold for any well-behaved prior over binary variables. In particular, the sources need not be marginally independent. This generality contrasts with the key assumption of independence that underlies many linear combination models. It suggests that an extension to a hierarchical model, in which higher-order statistics in the source distribution were captured by a further parametric model, might be straightforward.

The general formalism also remains unchanged for different noise distributions. Thus, if the conditional distribution of observations given sources, here Poisson, were instead Gaussian, all of the derivations and approximations would essentially remain unchanged, with one exception being the definitions of $I_i^{(n)}$, $I_{ab}^{(n)}$, and $I_{abc}^{(n)}$ in Equation (18). Approximate learning for the additional variance parameter of a Gaussian distribution would also be straightforward, largely following the arguments developed for the parameters W and $\vec{\pi}$. If suited to the data set under consideration, distributions other than Poisson and Gaussian may also be used within this same framework, and combined with different dependent or independent prior distributions.

7.3 Relationship to Variational Approximations

A now standard approach to approximate learning in intractable models is to replace the true posterior distribution of equation (12) by an approximate q_n that is obtained by minimizing the Kullback-Leibler divergence $\mathsf{KL}[q_n \| p(\vec{s}\,|\,\vec{y}^{(n)}, \Theta')]$ within a constrained class of functions. This provides a form of variational learning (Jordan et al., 1999), which provably increases a lower bound on the likelihood at every iteration. A common choice of a constrained family might be one which factors over the latent variables. Unfortunately, this common choice is of little benefit in the MCA generative model, as the costs of evaluation of the expected values of the derivatives $\mathcal{A}_{id}(\vec{s}, W)$, given in (8), grows exponentially even under factored distributions.

An alternative approach would be to constrain q_n to place mass only on source configurations where a limited number of causes are active. The minimum Kullback-Leibler divergence under this constraint would then be achieved when the probability of such sparse configurations under q_n was proportional to the corresponding true posterior values. Revisiting the argument leading to equation (15), it is clear that such an approximation would correspond to truncating the sums in both numerator and denominator of (15), as well as the corresponding expression for $\langle s_h \rangle_{q_n}$, at the same point. Our experience has been that the algorithms described here, in which fewer terms are kept in the numerator of (15) than in the denominator, always perform better than this strict variational approach.

7.4 The Different MCA Algorithms

The computational cost of exact expectation-maximization learning (i.e., MCA_{ex}) in the MCA generative model grows exponentially in the smaller of the number of observation dimensions and the number of hidden variables ($\min(D, H)$), and is thus generally intractable. We have introduced three approximations, all based on early truncation of the expanded sums in Equation (15). One of these, MCA_3, with cubic computational complexity, learns all the parameters of the full generative model, including the prior source probabilities. However, if the sums over source distributions are truncated further, to yield an algorithm with quadratic complexity, experimental performance of an otherwise unconstrained algorithm suffers. This difficulty is avoided in the restricted version of the generative model, in which the prior probabilities are held fixed and equal, and the weights associated with the sources satisfy a homogeneous coverage property. In this restricted model, the quadratic-cost algorithm becomes effective, and we have studied both a batch-mode algorithm, R-MCA_2, and an online version R-MCA_{NN}. Experiments showed that these restricted algorithms remained effective in terms of identifying generative weight vectors, even when the data were generated with prior source probabilities that were substantially different from that assumed by the algorithms (see, e.g., the experiments of Figure 8A or the discussion of violations of model assumptions in Experiments). Furthermore, the R-MCA algorithms were also robust to violations of the assumption of homogeneously distributed hidden causes (20). In some situations, the R-MCA algorithms succeeded in extracting the true causes where MCA_3 did not. We have observed that this is particularly true for bars with large differences in intensity. For this type of data, R-MCA_2 appears to be the most robust of the algorithms (see Lücke and Sahani, 2007). The R-MCA algorithms may also be more robust to greater differences in bar widths than those that we have studied here. Overall, in terms of the reliability with which hidden sources are recovered, R-MCA_2 and R-$\mathrm{MCA}_{\mathrm{NN}}$ may outperform MCA_3 even in experiments in which the assumptions used to derive them are violated. These results suggest that constrained optimization can improve measures such as reliability or learning time (in terms of pattern presentations), even when the constraint is not exactly valid. Approximately valid constraints may make it easier to avoid local optima, and to learn from fewer examples. However, the more severe approximation of R-MCA_2 and R-$\mathrm{MCA}_{\mathrm{NN}}$ can affect the likelihood of the parameters found. In this sense, MCA_3 is the more successful algorithm. One approach to increasing the speed of convergence might be to use R-$\mathrm{MCA}_{\mathrm{NN}}$ to provide initial values to MCA_3, thus reducing the number of cubic-complexity iterations required for final convergence. Such a hybrid algorithm would provide a learning system with relatively short learning times, high final likelihoods and high reliability.

7.5 Relationship to Previous Algorithms

It is helpful to divide the algorithms that have previously been proposed for component extraction into three groups: generative models with linear superposition, competitive generative models, and neural network models in which assumptions about the data are implicit in the network structure and learning algorithm.

7.5.1 LINEAR SUPERPOSITION MODELS

The functional difference between linear superposition and the max-rule has been discussed above. Despite the mismatch in the generative process, linear superposition models have been used within non-linear component extraction contexts, with some success. In particular, the non-negativity constraints of NMF have helped to identify constructively combined features (Lee and Seung, 1999).

For non-negative data, the specific algorithms developed here (MCA_3, R-MCA_2, and R-$\mathrm{MCA}_{\mathrm{NN}}$) can all be regarded as explicitly non-linear alternatives to the different versions of NMF. In particular, the Poisson noise distribution matches one of the cost functions often used with NMF (Lee and Seung, 1999). The basic methodology of our MCA development is, however, independent of the assumption of non-negativity.

It is worth noting that non-negativity may be better suited to finding featural sub-parts (cf., Lee and Seung, 1999; Wersing and Körner, 2003) of generative components (as was, in fact, originally proposed) than the entire components. In the bars test (with $b = 16$) Spratling (2006) showed that at least some NMF algorithms succeed in extracting all of the bars. However, if the bar overlap is increased (as in the test depicted in Figure 2 and Figure 8C), most NMF algorithms fail. For such input data, NMF only succeeds if its objective function is extended by an additional term that enforces a form of sparseness. MCA_3 and R-MCA_2 perform better on these data than all other algorithms tested (see Figure 9 for results) except for one sparse-NMF version (NN-SC; Hoyer,

2002) that performs equally well. However, for this and other sparse versions of NMF the sparseness parameter (or parameters) must be chosen either based on prior knowledge about the input, or by trial-and-error (see Spratling, 2006, for a critical discussion).

In contrast, the *generative model* underlying MCA does not assume sparseness. Instead, the notion of sparseness was introduced in the discussion of learning, to justify the tractable approximate learning algorithms MCA_3, R-MCA_2, and R-MCA_{NN}. The truncated approximations that underlie these algorithms are more accurate when the input causes are sparsely active; but this does not incorporate an explicit prior for sparsity in the way that SC, ICA, or sparse-NMF do, and does not enforce a pre-specified degree of sparseness in the learned generative model. Indeed, the MCA algorithms were found to robustly optimize the data likelihood, even for input causes that were not sparsely active on average. It is possible, however, that if data were generated by a process that was substantially different from that assumed by MCA, the approximate algorithms might well introduce a bias towards a sparser solution. These differences in approach to sparsity between MCA and models such as SC, ICA, and sparse-NMF, suggest that MCA might provide a good basis from which to study the relationship between non-linear component combinations and sparsity assumed in learning algorithms.

7.5.2 COMPETITIVE GENERATIVE MODELS

Models that use an explicitly non-linear generative combination rule include those of Saund (1995), which uses a noisy-or rule for binary observations, and of Dayan and Zemel (1995), where the combination scheme is more competitive. The MCA model may be viewed as taking this competition to an extreme, by selecting just one hidden variable to be responsible for each observed one.

Competitive generative models have proven challenging from a learning standpoint, in that published algorithms often converge to local optima. In the bars test ($b = 10$, $N = 500$) the noisy-or algorithm (Saund, 1995) finds all bars in just 27% of trials. The more competitive scheme (Dayan and Zemel, 1995) only extracts all bars if bar overlap is excluded for training, that is, if training patterns only contain parallel bars. In this simplified case the system achieves 69% reliability.

For comparison, the MCA learning algorithms MCA_3, R-MCA_2, and R-MCA_{NN} all show significantly higher values of reliability in the same bars test (see Table 1), at least when combined with an annealing procedure. The reliability can be boosted further in two ways—either by adding Poisson noise to the input (Table 1), or by adding more hidden variables or units to the model (in which case all three MCA algorithms find all 10 bars in all of our experiments).

7.5.3 NEURAL NETWORK MODELS

High reliability in component extraction in the bars test is not a feature exclusive to the new algorithms presented. Other highly reliable systems include some that optimize a non-probabilistic objective function (e.g., Charles and Fyfe, 1998; Hochreiter and Schmidhuber, 1999; Charles et al., 2002) as well as neural network models (Spratling and Johnson, 2002; Lücke and von der Malsburg, 2004; Lücke, 2004; Lücke and Bouecke, 2005; Spratling, 2006; Butko and Triesch, 2007). While the probabilistic generative approach has the advantage of a principled framework, which makes clear the assumptions being made about the data, it has been criticized (Hochreiter and Schmidhuber, 1999; Spratling and Johnson, 2002; Lücke and von der Malsburg, 2004) for not working reliably—that is, for often failing to extract the true causes.

The models and algorithms introduced here show that a generative approach can indeed be made robust. The R-MCA$_{\text{NN}}$ algorithm shows that generative and neural network approaches can come together in the form of a competitive neural network model that is both reliable and probabilistically interpretable. Using a high learning rate and additional noise, the network model R-MCA$_{\text{NN}}$ avoids local optima and needs few pattern presentations for learning (less than 1000 in the majority of trials). The same is reported for other network models (Spratling and Johnson, 2002; Lücke and von der Malsburg, 2004; Spratling, 2006) which fit into the framework of Equation (23) and Figure 4. The appropriate activation rule for a network to optimize the data likelihood under our generative model turns out to be a generalization of the softmax rule (see Equation 25). For input generated by very sparsely active causes, this generalization reduces to the usual softmax, which is commonly used for clustering (see, e.g., McLachlan and Peel, 2000). The generalized rule (25) therefore offers an explanation for why some networks (Spratling and Johnson, 2002; Lücke and von der Malsburg, 2004; Lücke, 2004) can also be successfully applied to clustering tasks. However, R-MCA$_{\text{NN}}$ and standard neural network algorithms can differ in the details of their behavior. On the one hand, for data involving substantial overlap between components (e.g., Figure 9), R-MCA$_{\text{NN}}$ seems to be more robust than the DI and NN-DI networks discussed by Spratling (2006). On the other hand, DI and the network of Lücke and von der Malsburg (2004) seem to be more robust to larger differences in component sizes.

A distinguishing feature of our model is the use of the max function. In neural modeling this function has also been used in other contexts and for other purposes. Among other models (e.g., Grzywacz and Yuille, 1990) it has been used as an activation function for hidden units in a feed-forward model for visual object recognition (Riesenhuber and Poggio, 1999). However this use in the recognition model should not be confused with our use of the max function in the generative process. Indeed, inference within the MCA model shows that the appropriate activation function of hidden units, for example, (16) or (22), is necessarily more complex. The extraction of input components, for example, in the bars test, fails if a simple max is used for inference instead. However, for input without superposition (and for recognition after learning) a max function as used by Riesenhuber and Poggio (1999) may be interpreted as a further approximation of the generalized softmax in the neural network approximation of R-MCA$_{\text{NN}}$.

7.6 Conclusion

To conclude, we have formulated a novel class of generative models that competitively combines hidden causes. In place of the linear superposition of prominent models like PCA, ICA, SC, and NMF, we use the max-operation. We have shown how a new technique for posterior approximation in such models can provide efficient parameter update rules if the input causes are sparsely active. Making specific choices for prior and noise distributions, we obtain efficient algorithms that performed well on artificial and natural non-linear mixtures, and are found to be competitive with the best current performance on standard non-linear benchmarks.

Acknowledgments

We thank Richard Turner for helpful discussions and for assistance with the acoustic data. This work was funded by the Gatsby Charitable Foundation.

Appendix A. Maximum Likelihood

To maximize $\mathcal{F}(\Theta,\Theta')$ in (5) with respect to W_{id} we require that:

$$\begin{aligned} \frac{\partial}{\partial W_{id}}\mathcal{F}(\Theta,\Theta') &\overset{!}{=} 0 \\ \Rightarrow \quad \sum_n \sum_{\vec{s}} q_n(\vec{s};\Theta') \sum_{d'} \left(\frac{\partial}{\partial W_{id}} \log\left(p(y_{d'}^{(n)} \,|\, \overline{W}_{d'}(\vec{s},W)) \right) \right) &\overset{!}{=} 0 \\ \Rightarrow \quad \sum_n \sum_{\vec{s}} q_n(\vec{s};\Theta') \left(\frac{\partial}{\partial W_{id}} \overline{W}_d(\vec{s},W) \right) f(y_d^{(n)}, \overline{W}_d(\vec{s},W)) &\overset{!}{=} 0\,, \end{aligned} \tag{26}$$

$$\text{where} \qquad f(y,w) = \frac{\partial}{\partial w}\log(p(y\,|\,w))\,, \tag{27}$$

with $p(y|w)$ given in (2). Now, for any well-behaved function g, and large ρ:

$$\mathcal{A}^{\rho}_{id}(\vec{s},W)\, g(\overline{W}_d(\vec{s},W)) \approx \mathcal{A}^{\rho}_{id}(\vec{s},W)\, g(W_{id})\,, \text{ where } \mathcal{A}^{\rho}_{id}(\vec{s},W) := \frac{\partial}{\partial W_{id}} \overline{W}^{\rho}_d(\vec{s},W). \tag{28}$$

Equation (28) holds because $\mathcal{A}^{\rho}_{id}(\vec{s},W) \approx 0$ whenever $\overline{W}_d(\vec{s},W) \neq W_{id}$. Hence it follows from (26) that:

$$\sum_n \sum_{\vec{s}} q_n(\vec{s};\Theta')\, \mathcal{A}^{\rho}_{id}(\vec{s},W)\; f(y_d^{(n)}, W_{id}) \overset{!}{\approx} 0\,, \tag{29}$$

$$\Rightarrow \quad \sum_n \sum_{\vec{s}} q_n(\vec{s};\Theta')\, \mathcal{A}^{\rho}_{id}(\vec{s},W) \left(y_d^{(n)} - W_{id} \right) \overset{!}{\approx} 0\,. \tag{30}$$

Equation (9) is obtained in the limit of large ρ. To be more precise, we might have used $\mathcal{A}^{\rho}_{id}(\vec{s},W)$ instead of $\mathcal{A}_{id}(\vec{s},W)$ in the main text. However, we abstained from doing so for the sake of readability, and because only the limit $\rho \to \infty$ is needed to derive the learning algorithms. The expression for this limit given in Equation (8) is found from (7), with the derivative of $\overline{W}^{\rho}_d$ given by:

$$\frac{\partial}{\partial W_{id}} \overline{W}^{\rho}_d(\vec{s},W) = \left(\frac{s_i\,(W_{id})^{\rho}}{\sum_h s_h (W_{hd})^{\rho}} \right) \frac{(\sum_h s_h (W_{hd})^{\rho})^{\frac{1}{\rho}}}{s_i\, W_{id}}\,. \tag{31}$$

In the limit $\rho \to \infty$ this reduces to (8) because the second factor on the right-hand-side converges to 1 whenever the first term is nonzero.

Note that (29) is true for any type of conditionally independent noise distribution. Only in the final step, from Equation (29) to (30), is the specific form of the Poisson distribution needed, where it appears in the derivative (27).

Appendix B. Intractability of Sufficient Statistics

To compute the exact sufficient statistics in (10) requires the evaluation of sums over all possible hidden states $\vec{s}$, suggesting a computational complexity of 2^H. In some cases, however, there may be multiple different source configurations, all of which result in the same effective weights $\overline{W}_d(\vec{s},W)$ for all dimensions $d \in \{1,\ldots,D\}$. In such cases, it might be possible to group these equal terms in each of the sums together, thereby reducing the complexity of the sum to scale with the number of such groups, rather than with the number of source configurations. In fact, we show below that the complexity of computing these expected values in general scales at least as fast as $2^{\min(H,D)}$.

We examine three cases individually:

$H = D$ Consider parameters W for which, corresponding to each observed node d, there is a hidden node i such that $W_{id} > W_{jd}$ for all $j \neq i$ (beyond this restriction, the entries in W may have arbitrary values). For $H = D$ this condition can be satisfied, and if it is, then for any two hidden vectors $\vec{s}$ and $\vec{s}'$ there is a d such that $\overline{W}_d(\vec{s},W) \neq \overline{W}_d(\vec{s}',W)$. In other words: any change of the hidden vector $\vec{s}$ results in a change of the pre-noise output vector $(\overline{W}_1(\vec{s},W),\ldots,\overline{W}_D(\vec{s},W))$. Hence, each summand in the partition function in (12) contributes a potentially different value, and they must be evaluated one-by-one. Thus, in this case the computational cost scales as 2^H.

$H < D$ Consider a subset of H of the D observed nodes and apply the argument above. Thus, the computational cost scales as 2^H. Note that for fixed H and random parameters W the existence of H (or approximately H) hidden nodes for which the above condition is fulfilled becomes increasingly likely with increasing D.

$H > D$ On the one hand, if we just consider D of the H hidden nodes and apply the argument above, we can infer that the computational complexity grows with at least 2^D. On the other hand, we can obtain at most $H^D + 1$ different vectors $(\overline{W}_1(\vec{s},W),\ldots,\overline{W}_D(\vec{s},W))$ and thus at most $H^D + 1$ groups to sum over. The computational complexity thus lies between 2^D and $H^D + 1$ in this case.

Appendix C. MCA$_3$ and R-MCA$_2$—Details of the Derivations

The update rules that define the algorithms MCA$_3$ and R-MCA$_2$ follow directly from (12) and (15) using the distributions (1) and (2). Note that in (15) the joint probability $p(\vec{s},\vec{y}^{(n)}\,|\,\Theta')$ can be replaced by any function F satisfying $F(\vec{s},\vec{y}^{(n)},\Theta') = \frac{p(\vec{s},\vec{y}^{(n)}\,|\,\Theta')}{A(\vec{y}^{(n)},\Theta')}$, where A is any well-behaved function not depending on $\vec{s}$. For the update rules of MCA$_3$ and R-MCA$_2$ we have used:

$$F(\vec{s},\vec{y}^{(n)},\Theta) = \left(\prod_i \overline{\pi}_i^{s_i}\right)\exp(I^{(n)}), \quad I^{(n)} = \sum_d \Big(\log(\overline{W}_d(\vec{s},W))\,y_d^{(n)} - \overline{W}_d(\vec{s},W)\Big).$$

C.1 MCA$_3$

The first term in the sum over states $\vec{s}$ in the denominator of (16) and (17) only contributes significantly if $\vec{y}^{(n)} = \vec{0}$, that is, if $F(\vec{0},\vec{y}^{(n)},\Theta) = 1$ (given Poisson noise). In all other cases its contribution is negligible. To derive the numerator of (16) we have used the property:

$$\mathcal{A}_{id}(\vec{s}_h,W) = \delta_{ih}, \qquad \mathcal{A}_{id}(\vec{s}_{ab},W) = \delta_{ia}\,\mathcal{H}(W_{id}-W_{bd}) + \delta_{ib}\,\mathcal{H}(W_{id}-W_{ad}), \tag{32}$$

where $\mathcal{H}$ is the Heaviside function. Note that instead of (32) we also could have used $\mathcal{A}_{id}(\vec{s}_h,W)$ and $\mathcal{A}_{id}(\vec{s}_{ab},W)$ directly or the corresponding expressions of $\mathcal{A}^{\rho}_{id}(\vec{s},W)$ in (31) with high ρ. By using (32) we can simplify the expression of the numerator of (16), however.

The derivation of the numerator of (17) (with $\alpha = 1$) is straightforward. For less sparse input we have to correct for neglecting input patterns which were generated by four or more hidden causes.

We do so by updating α using a consistency argument. On the one hand, using the same arguments as for the derivation of (16) and (17), we can estimate the total number of input patterns generated by less than three causes:

$$\mathcal{N}^{\leq 2}(Y,\Theta) \approx \sum_n \frac{\mathcal{H}(\frac{1}{2} - \sum_d y_d^{(n)}) + \sum_i \overline{\pi}_i \exp(I_i^{(n)}) + \frac{1}{2}\sum_{a,b\,(a\neq b)} \overline{\pi}_a \overline{\pi}_b \exp(I_{ab}^{(n)})}{1 + \sum_h \overline{\pi}_h \exp(I_h^{(n)}) + \frac{1}{2}\sum_{\substack{a,b\\ a\neq b}} \overline{\pi}_a \overline{\pi}_b \exp(I_{ab}^{(n)}) + \frac{1}{6}\sum_{\substack{a,b,c\\ a\neq b\neq c}} \overline{\pi}_a \overline{\pi}_b \overline{\pi}_c \exp(I_{abc}^{(n)})}\,. \tag{33}$$

On the other hand, the same number can be estimated using the prior distributions alone:

$$\tilde{\mathcal{N}}^{\leq 2}(\vec{\pi}) \;=\; N \left(\prod_i (1-\pi_i)\right) \left(1 + \sum_h \overline{\pi}_h + \frac{1}{2} \sum_{a,b\,(a\neq b)} \overline{\pi}_a \overline{\pi}_b \right). \tag{34}$$

If the parameters π_i are underestimated using approximation (17), the estimate (33) is smaller than the estimate (34). We change α after each EM iteration until both estimates are consistent ($\mathcal{N}^{\leq 2}(Y,\Theta) \approx \tilde{\mathcal{N}}^{\leq 2}(\vec{\pi})$):

$$\alpha \;=\; \alpha^{\mathrm{old}} + \frac{\varepsilon_\alpha}{N} \left(\tilde{\mathcal{N}}^{\leq 2}(\vec{\pi}) - \mathcal{N}^{\leq 2}(Y,\Theta)\right).$$

Note that the additional computational cost to infer α is small. Computations in (34) scale quadratically with H, and the terms in (33) have to be computed for (17) anyway. In experiments we use $\varepsilon_\alpha = 1$.

C.2 R-MCA$_2$

If we optimize (5) under the constraint $\sum_d W_{id} = C$ in (19), we obtain:

$$\sum_n \langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} \frac{y_d^{(n)} - W_{id}}{W_{id}} + \mu_i \;=\; 0\,.$$

The elimination of the Lagrange multipliers μ_i results in:

$$W_{id} \;=\; \frac{\sum_n \langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} y_d^{(n)}}{\frac{1}{C}\sum_{n,d'} \langle \mathcal{A}_{id'}(\vec{s},W)\rangle_{q_n} y_{d'}^{(n)} - \sum_n \left((\sum_{d'} \langle \mathcal{A}_{id'}(\vec{s},W)\rangle_{q_n} \frac{W_{id'}}{C}) - \langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} \right)}\,.$$

If the model parameters W fulfill condition (20), which can be expected at least close to the maximum likelihood solution, we obtain after the rearrangement of terms, the approximate M-step of Equation (21). To derive the sufficient statistics (22) note that given the update rule (21) we have:

$$W_{id} = C\, \frac{\sum_n \langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} y_d^{(n)}}{\sum_{d'}\sum_n \langle \mathcal{A}_{id'}(\vec{s},W)\rangle_{q_n} y_{d'}^{(n)}} = C\, \frac{\sum_{n\in N^{>0}} \langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} y_d^{(n)}}{\sum_{d'}\sum_{n\in N^{>0}} \langle \mathcal{A}_{id'}(\vec{s},W)\rangle_{q_n} y_{d'}^{(n)}}\,,$$

where $N^{>0}$ is the set of all non-zero input patterns. To approximate $\langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n}$ we can therefore assume input statistics that follow from Equations (1) to (3), but in which all inputs exactly equal to zero are omitted. For such modified input statistics, Equation (15) remains unchanged except for the partition function $\mathcal{Z}$ in (14) whose term $p(\vec{0},\vec{y}^{(n)}\,|\,\Theta')$ now equals zero. If we truncate the sum in (15) after terms of order two, we obtain Equation (22).

Appendix D. Neural Network Details

Here we show that the online neural network update rule (23) approaches the batch rule (24) for large data sets and small learning rates. Let $\mathcal{W}^{(n)}$ be the weight matrix at the nth update, and for convenience define the Hebbian correlation signal to be $G_{id}^{(n)} = g_i(\vec{y}^{(n)}, \mathcal{W}^{(n)})\, y_d^{(n)}$, and the weight renormalization term to be $z_i^{(n)} = C^{-1} \sum_{d'} \big(\mathcal{W}_{id'}^{(n)} + \varepsilon G_{id'}^{(n)}\big) = (1 + \frac{\varepsilon}{C} \sum_{d'} G_{id'}^{(n)})$ (where we have used the fact that $\sum_{d'} \mathcal{W}_{id'}^{(n)} = C$). Then we can rewrite (23) as

$$\mathcal{W}_{id}^{(n)} = \frac{\mathcal{W}_{id}^{(n-1)} + \varepsilon G_{id}^{(n-1)}}{z_i^{(n-1)}}\,,$$

and by applying N updates starting from initial step t find that

$$\mathcal{W}_{id}^{(t+N)} = \frac{\mathcal{W}_{id}^{(t)} + \varepsilon \sum_{n=1}^{N} G_{id}^{(t+N-n)} \prod_{k=n+1}^{N} z_i^{(t+N-k)}}{\prod_{k=1}^{N} z_i^{(t+N-k)}}\,,$$

where the empty product at $n = N$ is taken equal to 1.

We now make approximations based on the assumptions that N is large, ε is small, and that $\mathcal{W}^{(t)}$ is drawn from the equilibrium distribution over weights. First, as each $z_i^{(n)}$ is (slightly) larger than 0, the sum will be dominated by the leading terms (where n is small). The coefficients of these terms can be approximated, assuming that ε is small, by a logarithmic transform and the weak law of large numbers: $\prod_{k=n+1}^{N} z_i^{(t+N-k)} \approx \exp\big((N-n)\frac{\varepsilon}{C} \sum_{d'} \overline{G}_{id'}\big)$, where $\overline{G}_{id}$ is the expected value of $G_{id}^{(n)}$, which is taken to be stationary by the equilibrium assumption.

Inserting this expression into (35), taking the expected value of the right-hand-side, and summing the resulting geometric series to infinity, we obtain:

$$\mathcal{W}_{id}^{(t+N)} \approx e^{-N\varepsilon \sum_{d'} \overline{G}_{id'}/C}\, \mathcal{W}_{id}^{(t)} + \varepsilon \overline{G}_{id} \left(\frac{1}{1 - e^{-\varepsilon \sum_{d'} \overline{G}_{id'}/C}} - 1 \right).$$

Finally, assuming $N\varepsilon$ to be large enough for the first term to be negligible, expanding the second, and keeping only terms that do not scale with ε we obtain

$$\mathcal{W}_{id}^{(t+N)} \;\approx\; C \frac{\overline{G}_{id}}{\sum_{d'} \overline{G}_{id'}} \;\approx\; C\, \frac{\sum_{n=1}^{N} g_i(\vec{y}^{(n)}, \mathcal{W}^{(t)})\, y_d^{(n)}}{\sum_{d'} \sum_{n=1}^{N} g_i(\vec{y}^{(n)}, \mathcal{W}^{(t)})\, y_{d'}^{(n)}}\,.$$

This equivalence of the online and batch rules at equilibrium shows that the average fixed points of R-MCA$_{\text{NN}}$ equal fixed points of R-MCA$_2$. The equivalence becomes inexact away from equilibrium, although our experiments suggest that the behaviour during convergence may nonetheless be similar (Lücke and Sahani, 2007).

Appendix E. Experimental Details

This appendix gives details of the training procedures used for the MCA algorithms, provides more information about the other algorithms used for comparison, and gives particulars of the acoustic and visual data used.

E.1 Initialization

We initialized the parameters W by drawing each W_{id} from a Gaussian distribution with unit mean and standard deviation of $\frac{1}{3}$. Thereafter we normalized such that the average over all W_{id} was W^{init}. For MCA$_3$ we used $W^{\text{init}} = 4$ and for the R-MCA algorithms we used $W^{\text{init}} = 2$ (for this choice the sum, $\sum_d W_{id} = 50$, corresponds to the sum of the parameters used to generate the data). The parameters of the prior distribution were initialized to be $\pi_i = \frac{1}{H}$ for MCA$_3$, that is, half the value of the generating $\pi_i^{\text{gen}} = \frac{2}{H}$ in the standard bars test. For R-MCA$_2$ and R-MCA$_{\text{NN}}$ we initialized with values $\pi_i = \pi = \frac{2}{H}$. The reliability of both R-MCA algorithms is only marginally affected by the exact choices of π and W^{init}. Reliability values remained about the same even if the assumed values of π differed significantly from the generating values π_i^{gen} (see, e.g., the experiments of Figure 8 or the paragraph on 'Violations of model assumptions' in Experiments).

E.2 Annealing

In Equations (16) and (18) we make the following substitutions:

$$\mathcal{H}(x) \to \mathcal{S}_T(x) = \left(1 + \exp\left(-\tfrac{\lambda}{T-1}x\right)\right)^{-1}, \tag{35}$$

$$\overline{\pi}_i \to (\overline{\pi}_i)^\beta, \;\; I_i^{(n)} \to \beta I_i^{(n)}, \;\; I_{ab}^{(n)} \to \beta I_{ab}^{(n)}, \;\; I_{abc}^{(n)} \to \beta I_{abc}^{(n)}, \;\; \text{with} \;\; \beta = \tfrac{1}{T}\,. \tag{36}$$

while making only the substitutions of (36) in Equation (33). Here, T plays the role of a 'temperature'. In the limit of $T = 1$, β is equal to one and the sigmoidal function $\mathcal{S}_T(x)$ converges to the Heaviside function, that is, we recover the original Equations (16) and (17). A temperature $T > 1$ has the effect of leveling the differences between the parameter updates to a certain extent. For a high temperature $T \gg 1$, the differences between the parameters associated with different hidden variables vanish after a few iterations.

Smoothing the Heaviside function in (35) is a technique frequently used, for example, in the context of perceptrons, and the substitutions in (36) correspond to the standard annealing procedure for EM (Ueda and Nakano, 1998; Sahani, 1999). The slope of the sigmoidal function $\mathcal{S}_T(x)$ at $x = 0$ is parameterized by λ whose value is set to $\lambda = 0.2$.

In experiments for MCA$_3$ and R-MCA$_2$ we started learning at a relatively high temperature $T_1 > 1$ and cooled to a value T_o close to one. A final temperature $T_o > 1$ makes the system more robust and counteracts over-fitting (Weiss, 1998). Experimental results on artificial data remained essentially the same when we used $T_o = 1$ but the cooling procedure needed to be slower to avoid numerical instabilities in this case. If not otherwise stated, we used $T_o = 1.5$. For MCA$_3$ and R-MCA$_2$ we cooled from T_1 to T_o in steps of $\Delta T = \frac{T_1 - T_o}{50}$ after each iteration. However, we did not change the temperature if the parameters W still changed significantly. More precisely, we only decreased the temperature if the change in W_{id} fell below a threshold $\theta_{\Delta W}$ for all $i = 1, \ldots, H$. In formulas:

$$(\forall i: \;\; \Delta W_i \; < \; \theta_{\Delta W}) \quad \Rightarrow \quad T^{\text{new}} = T^{\text{old}} - \Delta T\,, \tag{37}$$

$$\text{where} \quad \Delta W_i = \frac{\sqrt{\sum_d (W_{id}^{\text{old}} - W_{id}^{\text{new}})^2}}{\sum_d W_{id}^{\text{old}}}\,. \tag{38}$$

Cooling conditioned on small parameter changes in this way allows the use of larger cooling steps ΔT and thus leads to learning in fewer iterations. For all trials we used a threshold of

$\theta_{\Delta W} = 2.5 \times 10^{-3}$, except for the application to more realistic data for which we ran additional trials with $\theta_{\Delta W} = 0.25 \times 10^{-3}$.

In experiments it was observed that a given system had a critical temperature T_c above which the weights did not specialize to different patterns after random initialization. Instead the parameters converged to about the same values for all hidden variables (compare Sahani, 1999; Lücke, 2004). A natural choice of an initial temperature $T_1 > 1$ is therefore a value close to this critical temperature. Experiments on different versions of the bars test showed a roughly linear dependence between the critical temperature T_c and the number of input dimensions D. Thus, in all versions of the bars test we used $T_1 = 0.4D + 1$ and $T_1 = 0.7D + 1$ for MCA$_3$ and R-MCA$_2$, respectively. In the experiments on acoustic and visual data, the critical temperatures are lower than those measured in bars tests with same D, presumably due to more homogeneous distributions of input values in those cases (generating weights in the bars test were all either 0 or 10, whereas generating weights in the naturally-derived data could take on any value in $[0, 10]$). Thus, experiments on phoneme data started at an initial temperature of $T_1 = 70$ for MCA$_3$ and $T_1 = 100$ for R-MCA$_2$; and those on visual data started at $T_1 = 2$ for MCA$_3$ (8-by-8 patches, $H = 20$) and $T_1 = 4$ for R-MCA$_2$ (10-by-10 patches, $H = 50$). In all experiments the temperature was maintained at T_1 during the first ten iterations. After the system had cooled to T_o using (37) and (38) learning was terminated once all ΔW_i remained smaller than $\theta_{\Delta W}$ for 20 iterations.

For R-MCA$_{\mathrm{NN}}$ we used a fixed temperature of $T = 16$ if not otherwise stated, and stopped after all single causes were represented by the same hidden variables for 4000 pattern presentations. In a given trial, the first pattern presentation after which the representation did not change was taken as the learning time of R-MCA$_{\mathrm{NN}}$. For the acoustic data set we used $T = 70$ and additionally report results for $T = 50$ and values $T \leq 40$. For the visual data we used $T = 2$.

For MCA$_{\mathrm{ex}}$ in Figure 6 we have used a relatively fast and fixed cooling schedule.

E.3 Algorithms for the Comparison in Figure 9

For the comparison in Figure 9 we have reproduced data reported by Spratling (2006). While we have adopted the same abbreviations as were used there, we repeat them in Table 2 for the convenience of the reader.

Algorithm	Description
NN-SC	non-negative sparse coding ($\lambda = 1$)
SC-NMF$_\mathrm{A}$	NMF with a sparseness constraint of 0.5 on the basis vectors
SC-NMF$_\mathrm{AY}$	NMF with a sparseness constraint of 0.5 on the basis vectors and 0.7 on the activations
DI	dendritic inhibition network
NN-DI	dendritic inhibition network with non-negative weights
SC-NMF$_\mathrm{Y}$	NMF with a sparseness constraint of 0.7 on the activations
S-NMF	sparse-NMF ($\alpha = 1$)
NMF$_\mathrm{div}$	NMF with divergence objective
NMF$_\mathrm{mse}$	NMF with Euclidean objective
L-NMF	local NMF

Table 2: Description of the algorithms used for the comparison in Figure 9.

E.4 Acoustic Data

The data used to study component extraction in the acoustic domain were generated from recordings of three vowels (two of them diphthongs), [aʊ], [iː], and [ɔɪ] and three consonants [k], [t], [p]. The phonemes were spoken by a male voice and recorded at 8000Hz. The data were taken from a publicly accessible data base (Sunsite, 1997). To construct spectrograms we used 1000 samples for each of the phonemes, which required truncation in three cases and padding with zeros in the other three cases. The waveforms $z(t)$ were normalized in power such that $\frac{1}{T}\sum_t (z(t))^2 = 1$. The waveforms were then linearly mixed ($z_{\text{mix}}(t) = z(t) + z'(t) + \ldots$) to produce $N = 500$ observed spectrograms. The probability of a phoneme of appearing in a mixed waveform was set to $\frac{2}{6}$. The spectrograms of these mixtures were computed using short-time Fourier transforms with 50 frequency channels ranging from 100 to 4000 Hz, with logarithmic scaling of center frequencies (see Figure 11D). We used 9 Hamming windows of 200 samples each, with successive windows overlapping by 100 samples. We then took the logarithms of the magnitudes of the 50-by-9 spectrogram entries, and linearly rescaled the top 42.8 dB of dynamic range to lie between 0 and 10, with magnitudes more than 42.8 dB below the highest intensity being clipped to 0.

E.5 Visual Data

The image that was used for the experiments on visual data has been taken from the publicly available image database of the van Hateren group at `hlab.phys.rug.nl/imlib/`. Images of the database represent light intensities linearly, which results in most images appearing relatively dark if displayed using a finite gray scale (see van Hateren and van der Schaaf, 1998, for details). We have used image number 2338 (deblurred), cut out a segment of 500-by-500 pixels in the lower left corner and scaled it down to a resolution of 250-by-250 pixels.

References

A. J. Bell and T. J. Sejnowski. The "independent components" of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

N. J. Butko and J. Triesch. Learning sensory representations with intrinsic plasticity. *Neurocomputing*, 70(7-9):1130–1138, 2007.

D. Charles and C. Fyfe. Modelling multiple-cause structure using rectification constraints. *Network: Computation in Neural Systems*, 9:167–182, 1998.

D. Charles, C. Fyfe, D. MacDonald, and J. Koetsier. Unsupervised neural networks for the identification of minimum overcomplete basis in visual data. *Neurocomputing*, 47(1-4):119–143, 2002.

P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314, 1994.

P. Dayan and L. F. Abbott. *Theoretical Neuroscience*. MIT Press, Cambridge, 2001.

P. Dayan and R. S. Zemel. Competition and multiple cause models. *Neural Computation*, 7:565–579, 1995.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

B. S. Everitt. *An Introduction to Latent Variable Models*. Chapman and Hall, 1984.

P. Földiák. Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, 64:165–170, 1990.

C. Fyfe. A neural network for PCA and beyond. *Neural Processing Letters*, 6:33–41, 1997.

N. M. Grzywacz and A. L. Yuille. A model for the estimate of local image velocity by cells in the visual cortex. *Proceedings of the Royal Society of London B*, 239:129–161, 1990.

G. F. Harpur and R. W. Prager. Development of low entropy coding in a recurrent network. *Network: Computation in Neural Systems*, 7:277–284, 1996.

G. E. Hinton and Z. Ghahramani. Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society B*, 352:1177–1190, 1997.

G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The ‘wake-sleep’ algorithm for unsupervised neural networks. *Science*, 268:1158–1161, 1995.

S. Hochreiter and J. Schmidhuber. Feature extraction through LOCOCODE. *Neural Computation*, 11:679–714, 1999.

P. O. Hoyer. Non-negative sparse coding. In *Neural Networks for Signal Processing XII: Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 557–565. 2002.

P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

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

D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13*, pages 556–562, 2001.

J. Lücke. Hierarchical self-organization of minicolumnar receptive fields. *Neural Networks*, 17/8–9: 1377–1389, 2004.

J. Lücke. A dynamical model for receptive field self-organization in V1 cortical columns. In *Proceedings of the International Conference on Artificial Neural Networks*, LNCS 4669, pages 389–398. Springer, 2007.

J. Lücke and J. D. Bouecke. Dynamics of cortical columns – self-organization of receptive fields. In *Proceedings of the International Conference on Artificial Neural Networks*, LNCS 3696, pages 31–37. Springer, 2005.

J. Lücke and M. Sahani. Generalized softmax networks for non-linear component extraction. In *Proceedings of the International Conference on Artificial Neural Networks*, LNCS 4668, pages 657–667. Springer, 2007.

J. Lücke and C. von der Malsburg. Rapid processing and unsupervised learning in a model of the cortical macrocolumn. *Neural Computation*, 16:501–533, 2004.

G. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, 2000.

R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1998.

S. J. Nowlan. Maximum likelihood competitive learning. In *Advances in Neural Information Processing Systems 2*, pages 574–582, 1990.

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

R. C. O'Reilly. Generalization in interactive networks: The benefits of inhibitory competition and Hebbian learning. *Neural Computation*, 13:1199–1241, 2001.

R. P. N. Rao, B. A. Olshausen, and M. S. Lewicki, editors. *Probabilistic Models of the Brain: Perception and Neural Function*. Neural Information Processing. The MIT Press, Cambridge, MA, 2002.

M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, 1999.

S. T. Roweis. Factorial models and refiltering for speech separation and denoising. In *Proceedings Eurospeech*, volume 7, pages 1009–1012, 2003.

M. Sahani. *Latent Variable Models for Neural Data Analysis*. PhD thesis, California Institute of Technology, Pasadena, California, 1999. URL `http://www.gatsby.ucl.ac.uk/~maneesh/thesis/`.

E. Saund. A multiple cause mixture model for unsupervised learning. *Neural Computation*, 7: 51–71, 1995.

M. W. Spratling. Learning image components for object recognition. *Journal of Machine Learning Research*, 7:793–815, 2006.

M. W. Spratling and M. H. Johnson. Preintegration lateral inhibition enhances unsupervised learning. *Neural Computation*, 14:2157–2179, 2002.

Sunsite. Sun-sounds: Phonemes. Data retrieved in 2007 through `ibiblio.org` from `ftp://sunsite.unc.edu/pub/multimedia/sun-sounds/phonemes/`, 1997.

N. Ueda and R. Nakano. Deterministic annealing EM algorithm. *Neural Networks*, 11(2):271–282, 1998.

J. H. van Hateren and A. van der Schaaf. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings of the Royal Society of London B*, 265: 359–366, 1998.

Y. Weiss. Phase transitions and the perceptual organization of video sequences. In *Advances in Neural Information Processing Systems 10*, pages 850–856, 1998.

H. Wersing and E. Körner. Learning optimized features for hierarchical models of invariant object recognition. *Neural Computation*, 15(7):1559–1588, 2003.

A. L. Yuille and D. Geiger. Winner-take-all networks. In M.A. Arbib, editor, *The handbook of brain theory and neural networks*, pages 1228–1231. MIT Press, 2003.