

# A Truncated Variational EM Approach for Spike-and-Slab Sparse Coding

Abdul-Saboor Sheikh, Jacquelyn A. Shelton and Jörg Lücke

{sheikh, shelton, luecke}@fias.uni-frankfurt.de

Frankfurt Institute for Advanced Studies (FIAS)

Goethe-Universität Frankfurt

60438 Frankfurt, Germany

## Abstract

We study inference and learning based on a sparse coding model with ‘spike-and-slab’ prior. As standard sparse coding, the used model assumes independent latent sources that linearly combine to generate data points. However, instead of using a standard sparse prior such as a Laplace distribution, we study the application of a more flexible ‘spike-and-slab’ distribution which models the absence or presence of a source’s contribution independently of its strength if it contributes. We investigate two approaches to optimize the parameters of spike-and-slab sparse coding: a novel truncated variational EM approach and, for comparison, an approach based on standard factored variational distributions. In applications to source separation we find that both approaches improve the state-of-the-art in a number of standard benchmarks, which argues for the use of ‘spike-and-slab’ priors for the corresponding data domains. Furthermore, we find that the truncated variational approach improves on the standard factored approach in source separation tasks – which hints to biases introduced by assuming posterior independence in the factored variational approach. Likewise, on a standard benchmark for image denoising, we find that the truncated variational approach improves on the factored variational approach. While the performance of the factored approach saturates with increasing number of hidden dimensions, the performance of the truncated approach improves the state-of-the-art for higher noise levels.

**Keywords:** sparse coding, spike-and-slab distributions, approximate EM, variational Bayes, unsupervised learning, source separation, denoising

# 1 Introduction

Much attention has recently been devoted to studying sparse coding models with ‘spike-and-slab’ distribution as a prior over the latent variables [Goodfellow et al., 2012, Mohamed et al., 2012, Lücke and Sheikh, 2012, Titsias and Lazaro-Gredilla, 2011, Carbonetto and Stephen, 2011, Knowles and Ghahramani, 2011, Yoshida and West, 2010]. In general, a ‘spike-and-slab’ distribution is comprised of a binary (the ‘spike’) and a continuous (the ‘slab’) part. The distribution generates a random variable by multiplying together the two parts such that the resulting value is either exactly zero (due to the binary random variable being zero) or it is a value drawn from a distribution governing the continuous part. In sparse coding models, employing spike-and-slab as a prior allows for modeling the presence or absence of latents independently of their contributions in generating an observation. For example, piano keys (as latent variables) are either pressed or not (binary part), and if they are pressed, they result in sounds with different intensities (continuous part). Note that the sounds generated by a piano are also sparse in the sense that of all keys only a relatively small number is pressed on average.

Spike-and-slab distributions can flexibly model an array of sparse distributions, lending them desirable for many types of data. Algorithms based on spike-and-slab distributions have successfully been used, e.g., for transfer learning [Goodfellow et al., 2012], regression [West, 2003, Carvalho et al., 2008, Carbonetto and Stephen, 2011, Titsias and Lazaro-Gredilla, 2011], denoising [Zhou et al., 2009, Titsias and Lazaro-Gredilla, 2011], and often represent the state-of-the-art on given benchmarks [compare Titsias and Lazaro-Gredilla, 2011, Goodfellow et al., 2012].

The general challenge with spike-and-slab sparse coding models lies in the optimization of the model parameters. Whereas the standard Laplacian prior used for sparse coding results in uni-modal posterior distributions, the spike-and-slab prior results in multi-modal posteriors [see, e.g., Titsias and Lazaro-Gredilla, 2011, Lücke and Sheikh, 2012]. Fig. 1 shows typical posterior distributions for spike-and-slab sparse coding (the model will be formally defined in the next section). The panels illustrate the posteriors for the case of a two-dimensional observed and a two-dimensional hidden space. As can be observed, the posteriors have multiple modes and the number modes increases exponentially with the dimensionality of the hidden space [Titsias and Lazaro-Gredilla, 2011, Lücke and Sheikh, 2012]. The multi-modal structure of the posteriors argues against the application of the standard maximum a-posteriori (MAP) approaches [Mairal et al., 2009, Lee et al., 2007, Olshausen and Field, 1997] or Gaussian approximations of the posterior [Seeger, 2008, Ribeiro and Opper, 2011] because they rely on uni-modal posteriors. The approaches that have been proposed in the literature are, consequently, MCMC based methods [e.g., Carvalho et al.,

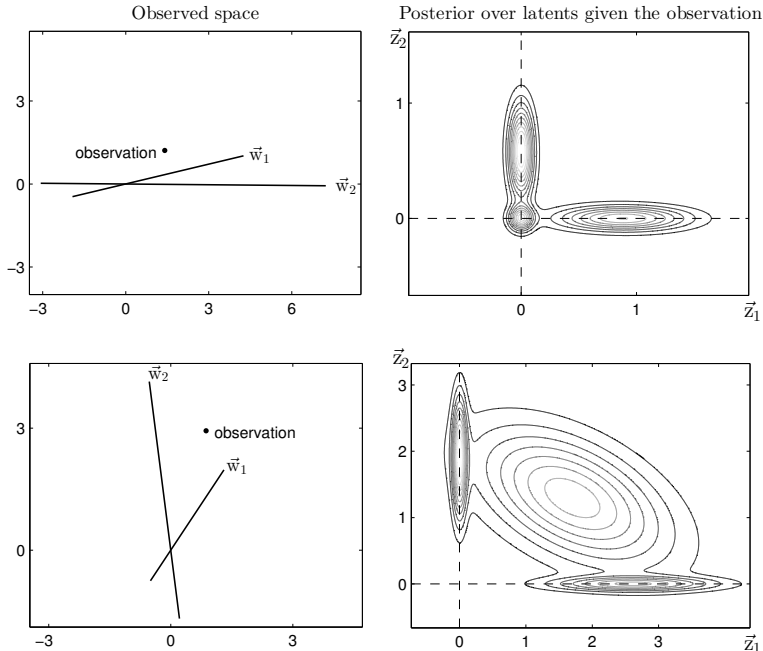


Figure 1: Left panels visualize observations generated by two different instantiations of the spike-and-slab sparse coding model (1) to (3). Solid lines are the generating bases vectors. Right panels illustrate the corresponding exact posteriors over latents computed using (16) and (19) given observations and generating model parameters. Note that the probability mass seen just along the axes or around the origin actually lies exactly on the axis. Here we have spread the mass for visualization purposes by slightly augmenting zero diagonal entries of the posterior covariance matrix in (19).

2008, Zhou et al., 2009, Mohamed et al., 2012] and variational EM methodologies [e.g., Zhou et al., 2009, Titsias and Lazaro-Gredilla, 2011, Goodfellow et al., 2012]. While MCMC approaches are more general and more accurate given sufficient computational resources, variational approaches are usually more efficient. Especially in high dimensional spaces, the multi-modality of the posteriors is a particular challenge for MCMC approaches; consequently, recent applications to large hidden spaces have been based on variational EM optimization [Titsias and Lazaro-Gredilla, 2011, Goodfellow et al., 2012]. The variational approaches applied to spike-and-slab models thus far [see Yoshida and West, 2010, Rattray et al., 2009, Goodfellow et al., 2012, Titsias and Lazaro-Gredilla, 2011] assume a factorization of the posteriors over the latent dimensions, i.e., the hidden dimensions are assumed to be independent a-posteriori. This means that any dependencies, e.g. explaining-away effects including correlations (compare Fig.1) are ignored and not accounted for. But what

consequences does such a neglect of posterior structure have? Does it result in biased parameter estimates and is it relevant for practical tasks? Biases induced by factored variational inference in latent variable models have indeed been observed before [MacKay, 2001, Ilin and Valpola, 2003, Turner and Sahani, 2011]. For instance, in source separation tasks, optimization through factored inference can be biased towards finding mixing matrices that represent orthogonal sparse directions, because such solutions are most consistent with the assumed a-posteriori independence [see Ilin and Valpola, 2003, for a detailed discussion]. Therefore, the posterior independence assumption in general may result in suboptimal solutions.

In this work we study a variational EM approach for spike-and-slab sparse coding which does not assume a-posteriori independence while being able to model multiple modes. Instead of using factored distributions or Gaussians, the novel approach is based on posterior distributions truncated to regions of high probability mass [Lücke and Eggert, 2010] – an approach which has recently been applied to different models [see e.g., Puertas et al., 2010, Shelton et al., 2011, Dai and Lücke, 2012]. In contrast to the previously studied factored approaches [Titsias and Lazaro-Gredilla, 2011, Goodfellow et al., 2012, Mohamed et al., 2012], the truncated approach will furthermore take advantage of the fact that in the case of a Gaussian slab and Gaussian noise model, integrals over the continuous latents can be obtained in closed-form [Lücke and Sheikh, 2012]. This implies that the posteriors over latent space can be computed exactly if the sums over the binary part are exhaustively evaluated over exponentially many states. This enumeration of the binary part becomes computationally intractable for high-dimensional hidden spaces. However, by applying the truncated variational distribution exclusively to the binary part of the hidden space, we can still fully benefit from the analytical tractability of the continuous integrals.

In this work, we systematically compare the truncated approach to a recently suggested factored variational approach [Titsias and Lazaro-Gredilla, 2011]. A direct comparison of the two variational approaches will allow for answering the questions about potential drawbacks and biases of both optimization procedures. As approaches assuming factored variational approximations have recently shown state-of-the-art performances [Titsias and Lazaro-Gredilla, 2011, Goodfellow et al., 2012], understanding their strengths and weaknesses is crucial for further advancements of sparse coding approaches and their many applications. Comparison with other approaches that are not necessarily based on the spike-and-slab model will allow for accessing the potential advantages of the spike-and-slab model itself.

In section 2 we will introduce the used spike-and-slab sparse coding generative model, and briefly discuss the factored variational approach which has recently been applied for parameter

optimization. In section 3 we derive the closed-form EM parameter update equations for the introduced spike-and-slab model. Based on these equations, in section 4 we derive the truncated variational EM algorithm for efficient learning in high dimensions. In section 5, we numerically evaluate the algorithm and compare it to factored variational and other approaches. Finally, in section 6 we discuss the results, and present details of the derivations and experiments in the Appendix.

## 2 Spike-and-slab Sparse Coding

The spike-and-slab sparse coding model assumes like standard sparse coding a linear superposition of basis functions, independent latents, and Gaussian observation noise. The main difference is that a spike-and-slab distribution is used as a prior. Spike-and-slab distributions have long been used for different models [e.g., Mitchell and Beauchamp, 1988, among many others] and also variants of sparse coding with spike-and-slab priors have been studied previously [compare West, 2003, Garrigues and Olshausen, 2007, Knowles and Ghahramani, 2007, Teh et al., 2007, Carvalho et al., 2008, Paisley and Carin, 2009, Zhou et al., 2009]. In this work we study a generalization of the spike-and-slab sparse coding model studied by Lücke and Sheikh [2012]. The data generation process in the model assumes an independent Bernoulli prior for each component of the the binary latent vector  $\vec{s} \in \{0, 1\}^H$  and a multivariate Gaussian prior for the continuous latent vector  $\vec{z} \in \mathbb{R}^H$ :

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

$$p(\vec{z}|\Theta) = \mathcal{N}(\vec{z}; \vec{\mu}, \Psi), \quad (2)$$

where  $\pi_h$  defines the probability of  $s_h$  being equal to one and where  $\vec{\mu}$  and  $\Psi$  parameterize the mean and covariance of  $\vec{z}$  respectively. The parameters  $\vec{\mu} \in \mathbb{R}^H$  and  $\Psi \in \mathbb{R}^{H \times H}$  parameterizing the Gaussian slab in (2) generalize the spike-and-slab model used in [Lücke and Sheikh, 2012]. The two latent variables  $\vec{s}$  and  $\vec{z}$  are combined via point-wise multiplication  $(\vec{s} \odot \vec{z})_h = s_h z_h$ . The resulting hidden variable  $(\vec{s} \odot \vec{z})$  follows a ‘spike-and-slab’ distribution, i.e., the variable entries have continuous values or are exactly zero. Given such a latent vector, a  $D$ -dimensional observation  $\vec{y} \in \mathbb{R}^D$  is generated by linearly superimposing a set of basis functions  $W$  and adding Gaussian noise:

$$p(\vec{y} | \vec{s}, \vec{z}, \Theta) = \mathcal{N}(\vec{y}; W(\vec{s} \odot \vec{z}), \Sigma), \quad (3)$$

where each column of the matrix  $W \in \mathbb{R}^{D \times H}$  is a basis function  $W = (\vec{w}_1, \dots, \vec{w}_H)$  and where the matrix  $\Sigma \in \mathbb{R}^{D \times D}$  parameterizes the observation noise. We use  $\Theta = (W, \Sigma, \vec{\pi}, \vec{\mu}, \Psi)$  to denote

all the model parameters. Note that having a spike-and-slab prior implies that for high levels of sparsity (i.e., low values of  $\pi_h$ ) the latents assume exact zeros with high probability. This is an important distinction compared to the Laplace or Cauchy distributions used for standard sparse coding [Olshausen and Field, 1997].

The spike-and-slab sparse coding algorithm we derive in this work is based on the model (1) to (3). The factored variational approach [Titsias and Lazaro-Gredilla, 2011] that we use for later comparison is based on a similar model, which we will refer to from now on as the MTMKL model. The MTMKL model is both a constrained and generalized version of the model we study. On one hand, it is more constrained by assuming the same sparsity for each latent, i.e.,  $\pi_h = \pi_{h'}$  (for all  $h, h'$ ); and by using a diagonal covariance matrix for the observation noise,  $\Sigma = \text{diag}(\sigma_1^2, \dots, \sigma_D^2)$ . On the other hand, it is a generalization by drawing the basis functions  $W$  from Gaussian processes. The model (1) to (3) can then be recovered as a special case of the MTMKL model if the Gaussian processes are Dirac delta functions. For parameter optimization, the MTMKL model uses a standard factored variational optimization. In the case of spike-and-slab models, this factored approach means that the exact posterior  $p(\vec{s}, \vec{z} | \vec{y})$  is approximated by a variational distribution  $q_n(\vec{s}, \vec{z}; \Theta)$  that assumes the combined latents to be independent a-posteriori [Titsias and Lazaro-Gredilla, 2011, Goodfellow et al., 2012]:

$$q_n(\vec{s}, \vec{z}; \Theta) = \prod_{h=1}^H q_n^{(h)}(s_h, z_h; \Theta), \quad (4)$$

where  $q_n^{(h)}$  are distributions only depending on  $s_h$  and  $z_h$  and not on any of the other latents. A detailed account of the MTMKL optimization algorithm is given by Titsias and Lazaro-Gredilla [2011] and for the later numerical experiments, we use the source code provided along with that publication<sup>1</sup>.

### 3 Expectation Maximization (EM) for Parameter Optimization

In order to learn the model parameters  $\Theta$  given a set of  $N$  independent data points  $\{\vec{y}^{(n)}\}_{n=1, \dots, N}$  with  $\vec{y}^{(n)} \in \mathbb{R}^D$ , we maximize the data likelihood  $\mathcal{L} = \prod_{n=1}^N p(\vec{y}^{(n)} | \Theta)$  by applying the Expectation Maximization (EM) algorithm. Instead of directly maximizing the likelihood, the EM algorithm [in the form studied by Neal and Hinton, 1998] maximizes the free-energy, a lower bound of the

---

<sup>1</sup>we downloaded the code from <http://www.well.ox.ac.uk/~mtitsias/code/varSparseCode.tar.gz>

log-likelihood given by:

$$\mathcal{F}(\Theta^{\text{old}}, \Theta) = \sum_{n=1}^N \left\langle \log p(\vec{y}^{(n)}, \vec{s}, \vec{z} | \Theta) \right\rangle_n + H(\Theta^{\text{old}}), \quad (5)$$

where  $\langle \cdot \rangle_n$  denotes the expectation under the posterior over the latents  $\vec{s}$  and  $\vec{z}$  given  $\vec{y}^{(n)}$

$$\langle f(\vec{s}, \vec{z}) \rangle_n = \sum_{\vec{s}} \int_{\vec{z}} p(\vec{s}, \vec{z} | \vec{y}^{(n)}, \Theta^{\text{old}}) f(\vec{s}, \vec{z}) d\vec{z} \quad (6)$$

and  $H(\Theta^{\text{old}}) = -\sum_{\vec{s}} \int_{\vec{z}} p(\vec{s}, \vec{z} | \vec{y}^{(n)}, \Theta^{\text{old}}) \log(p(\vec{s}, \vec{z} | \vec{y}^{(n)}, \Theta^{\text{old}})) d\vec{z}$  is the Shannon entropy, which only depends on parameter values held fixed during the optimization of  $\mathcal{F}$  w.r.t.  $\Theta$  in the M-step. Note that  $\sum_{\vec{s}}$  is a summation over all possible binary vectors  $\vec{s}$ .

The EM algorithm iteratively optimizes the free-energy by alternating between two steps. First, in the E-step given the current parameters  $\Theta^{\text{old}}$ , the relevant expectation values under the posterior  $p(\vec{s}, \vec{z} | \vec{y}^{(n)}, \Theta^{\text{old}})$  are computed. Next, the M-step uses these posterior expectations and maximizes the free-energy  $\mathcal{F}(\Theta^{\text{old}}, \Theta)$  w.r.t.  $\Theta$ . Iteratively applying E- and M-steps locally maximizes the data likelihood. In the following section we will first derive the M-step equations which themselves will require expectation values over the posteriors (Eqn. 6). The required expressions for these expectations (the E-step) will be derived afterwards.

### 3.1 M-step Parameter Updates

The M-step parameter updates of the model are canonically obtained by setting the derivatives of the free-energy (5) w.r.t. the second argument to zero. Details of the derivations are given in Appendix A and the resulting update equations are as follows:

$$W = \frac{\sum_{n=1}^N \vec{y}^{(n)} \langle \vec{s} \odot \vec{z} \rangle_n^T}{\sum_{n=1}^N \langle (\vec{s} \odot \vec{z})(\vec{s} \odot \vec{z})^T \rangle_n}, \quad (7)$$

$$\vec{\pi} = \frac{1}{N} \sum_{n=1}^N \langle \vec{s} \rangle_n, \quad (8)$$

$$\Sigma = \frac{1}{N} \sum_{n=1}^N \left[ \vec{y}^{(n)} (\vec{y}^{(n)})^T - 2(W \langle \vec{s} \odot \vec{z} \rangle_n) (\vec{y}^{(n)})^T + W (\langle (\vec{s} \odot \vec{z})(\vec{s} \odot \vec{z})^T \rangle_n) W^T \right], \quad (9)$$

$$\vec{\mu} = \frac{\sum_{n=1}^N \langle \vec{s} \odot \vec{z} \rangle_n}{\sum_{n=1}^N \langle \vec{s} \rangle_n} \quad (10)$$

$$\text{and } \Psi = \frac{\sum_{n=1}^N \langle (\vec{s} \odot \vec{z})(\vec{s} \odot \vec{z})^T \rangle_n - \vec{\mu} \vec{\mu}^T \sum_{n=1}^N \langle \vec{s} \vec{s}^T \rangle_n}{\sum_{n=1}^N \langle \vec{s} \vec{s}^T \rangle_n}. \quad (11)$$

### 3.2 E-step Expectation Values

The M-step equations (7) to (11) require expectation values w.r.t. the posterior distribution be computed over the whole latent space which requires either analytical solutions or approximations of integrals over the latent space. For the derivation of closed-form E-step equations it is useful to note that the discrete latent variable  $\vec{s}$  can be combined with the basis function matrix  $W$  so that we can rewrite (3) as

$$p(\vec{y} | \vec{s}, \vec{z}, \Theta) = \mathcal{N}(\vec{y}; \tilde{W}_{\vec{s}} \vec{z}, \Sigma), \quad (12)$$

where we have defined  $(\tilde{W}_{\vec{s}})_{dh} = W_{dh}s_h$  such that  $W(\vec{s} \odot \vec{z}) = \tilde{W}_{\vec{s}} \vec{z}$ .

Now first note that the marginal data likelihood  $p(\vec{y} | \Theta)$  can be derived in closed-form after marginalizing the joint  $p(\vec{y}, \vec{s}, \vec{z} | \Theta)$  over  $\vec{z}$ :

$$p(\vec{y}, \vec{s} | \Theta) = \mathcal{B}(\vec{s}; \vec{\pi}) \int \mathcal{N}(\vec{y}; \tilde{W}_{\vec{s}} \vec{z}, \Sigma) \mathcal{N}(\vec{z}; \vec{\mu}, \Psi) d\vec{z} \quad (13)$$

$$= \mathcal{B}(\vec{s}; \vec{\pi}) \mathcal{N}(\vec{y}; \tilde{W}_{\vec{s}} \vec{\mu}, C_{\vec{s}}), \quad (14)$$

where  $C_{\vec{s}} = \Sigma + \tilde{W}_{\vec{s}} \Psi \tilde{W}_{\vec{s}}^T$ . The second step follows from standard identities for Gaussian random variables [e.g., Bishop, 2006]. We can then sum the resulting expression over  $\vec{s}$  to obtain

$$p(\vec{y} | \Theta) = \sum_{\vec{s}} \mathcal{B}(\vec{s}; \vec{\pi}) \mathcal{N}(\vec{y}; \tilde{W}_{\vec{s}} \vec{\mu}, C_{\vec{s}}). \quad (15)$$

Thus, the marginal distribution takes the form of a Gaussian mixture model with  $2^H$  mixture components indexed by  $\vec{s}$ . However, unlike in a standard Gaussian mixture model, note that the mixing proportions and the parameters of the mixture components are not independent but coupled together. Therefore, the following steps will lead to closed-form EM updates that are notably not a consequence of closed-form EM for classical Gaussian mixtures. In contrast, Gaussian mixture models assume independent mixing proportions and independent component parameters. By using Eqns. 14 and 15 the posterior over the binary latents  $p(\vec{s} | \vec{y}, \Theta)$  is given by:

$$p(\vec{s} | \vec{y}, \Theta) = \frac{p(\vec{s}, \vec{y} | \Theta)}{p(\vec{y} | \Theta)} = \frac{\mathcal{B}(\vec{s}; \vec{\pi}) \mathcal{N}(\vec{y}; \tilde{W}_{\vec{s}} \vec{\mu}, C_{\vec{s}})}{\sum_{\vec{s}'} \mathcal{B}(\vec{s}'; \vec{\pi}) \mathcal{N}(\vec{y}; \tilde{W}_{\vec{s}'} \vec{\mu}, C_{\vec{s}'})}. \quad (16)$$

We can now consider the factorization of the posterior  $p(\vec{s}, \vec{z} | \vec{y}, \Theta)$  into the posterior over the binary part  $p(\vec{s} | \vec{y}, \Theta)$  and the posterior over the continuous part given the binary state  $p(\vec{z} | \vec{s}, \vec{y}, \Theta)$ :

$$p(\vec{s}, \vec{z} | \vec{y}, \Theta) = p(\vec{s} | \vec{y}, \Theta) p(\vec{z} | \vec{s}, \vec{y}, \Theta). \quad (17)$$



Like the first factor in (17), the second factor is also analytically tractable and given by:

$$\begin{aligned}
p(\vec{z} | \vec{s}, \vec{y}, \Theta) &= \frac{p(\vec{s} | \Theta) p(\vec{z} | \Theta) p(\vec{y} | \vec{z}, \vec{s}, \Theta)}{p(\vec{s} | \Theta) \int p(\vec{y} | \vec{z}, \vec{s}, \Theta) p(\vec{z} | \Theta) d\vec{z}} \\
&\propto \mathcal{N}(\vec{z}; \vec{\mu}, \Psi) \mathcal{N}(\vec{y}; \tilde{W}_{\vec{s}} \vec{z}, \Sigma) \\
&= \mathcal{N}(\vec{z}; \vec{\kappa}_{\vec{s}}, \Lambda_{\vec{s}}),
\end{aligned} \tag{18}$$

where the last step again follows from standard Gaussian identities with definitions

$$\begin{aligned}
\Lambda_{\vec{s}} &= (\tilde{W}_{\vec{s}}^T \Sigma^{-1} \tilde{W}_{\vec{s}} + \Psi_{\vec{s}}^{-1})^{-1}, \\
\vec{\kappa}_{\vec{s}}^{(n)} &= (\vec{s} \odot \vec{\mu}) + \Lambda_{\vec{s}} \tilde{W}_{\vec{s}}^T \Sigma^{-1} (\vec{y}^{(n)} - \tilde{W}_{\vec{s}} \vec{\mu})
\end{aligned} \tag{19}$$

and with  $\Psi_{\vec{s}} = \Psi(\text{diag}(\vec{s}))$ . The full posterior distribution can thus be written as

$$p(\vec{s}, \vec{z} | \vec{y}^{(n)}, \Theta) = \frac{\mathcal{B}(\vec{s}; \vec{\pi}) \mathcal{N}(\vec{y}^{(n)}; \tilde{W}_{\vec{s}} \vec{\mu}, C_{\vec{s}}) \mathcal{N}(\vec{z}; \vec{\kappa}_{\vec{s}}^{(n)}, \Lambda_{\vec{s}})}{\sum_{\vec{s}'} \mathcal{B}(\vec{s}'; \vec{\pi}) \mathcal{N}(\vec{y}^{(n)}; \tilde{W}_{\vec{s}'} \vec{\mu}, C_{\vec{s}'})}. \tag{20}$$

Equation 20 represents the crucial result for the computation of the E-step below because, first, it shows that the posterior does not involve analytically intractable integrals and, second, for fixed  $\vec{s}$  and  $\vec{y}^{(n)}$  the dependency on  $\vec{z}$  follows a Gaussian distribution. This special form allows for the derivation of analytical expressions for the expectation values as required for the M-step updates. Because of the Gaussian form the integrations over the continuous part are straight-forward and the resulting expressions are given by:

$$\langle \vec{s} \rangle_n = \sum_{\vec{s}} q_n(\vec{s}; \Theta) \vec{s}, \tag{21}$$

$$\langle \vec{s} \vec{s}^T \rangle_n = \sum_{\vec{s}} q_n(\vec{s}; \Theta) \vec{s} \vec{s}^T, \tag{22}$$

$$\langle \vec{s} \odot \vec{z} \rangle_n = \sum_{\vec{s}} q_n(\vec{s}; \Theta) \vec{\kappa}_{\vec{s}}^{(n)}, \tag{23}$$

$$\text{and } \langle (\vec{s} \odot \vec{z})(\vec{s} \odot \vec{z})^T \rangle_n = \sum_{\vec{s}} q_n(\vec{s}; \Theta) (\Lambda_{\vec{s}} + \vec{\kappa}_{\vec{s}}^{(n)} (\vec{\kappa}_{\vec{s}}^{(n)})^T). \tag{24}$$

Note that the left-hand-sides of the expressions above are expectation values over the full latent space w.r.t. the posterior  $p(\vec{s}, \vec{z} | \vec{y}^{(n)}, \Theta)$ , whereas the right-hand-sides now take the form of expectation values only over the binary part w.r.t. the posterior  $p(\vec{s} | \vec{y}^{(n)}, \Theta)$  in Eqn. 16. The derivations of E-step equations (21) to (24) are a generalization of the derivations by Lücke and Sheikh [2012]. While Gaussian identities and marginalizations have been used to obtain analytical results for mixture-of-Gaussian priors before [e.g. Attias, 1999, Garrigues and Olshausen, 2007, Olshausen and Millman, 2000], the above equations are the first closed-form solutions for the spike-and-slab model [first appearing in Lücke and Sheikh, 2012]. The observation that the Gaussian slab and

Gaussian noise model allows for analytically tractable integrals has, in parallel work, also been noted and used by [Mohamed et al., 2012].

Iteratively computing the E-step equations using the current parameters  $\Theta$  for equations (21) to (24) and the M-step equations (7) to (11), represents a closed-form and exact EM algorithm which increases the data likelihood of the model to (possibly local) maxima.

## 4 Truncated Variational EM

While being exact, the execution of the above EM algorithm results in considerable computational costs for larger-scale problems. Without approximations, the computational resources required scale exponentially with the number of hidden dimensions  $H$ . This can be seen by considering the expected values w.r.t. the posterior  $p(\vec{s} | \vec{y}, \Theta)$  above, which each require a summation over all binary vectors  $\vec{s} \in \{0, 1\}^H$ . For tasks involving low dimensional hidden spaces the exact algorithm is still applicable. However, for higher dimensional problems approximations are required. Still, we can make use of the closed-form EM solutions by applying an approximation solely to the binary part. Instead of sampling-based or factored approximations to the posterior  $p(\vec{s}, \vec{z} | \vec{y}, \Theta)$ , we use a truncated variational approximation to the posterior  $p(\vec{s} | \vec{y}^{(n)}, \Theta)$  in Eqn. 16. The truncated approximation is defined to be proportional to the true posteriors on subspaces of the latent space with high probability mass [compare Lücke and Eggert, 2010, Puertas et al., 2010]. More concretely, a posterior distribution  $p(\vec{s} | \vec{y}^{(n)}, \Theta)$  is approximated by a distribution  $q_n(\vec{s}; \Theta)$  that only has support on a subset  $\mathcal{K}_n \subseteq \{0, 1\}^H$  of the state space:

$$q_n(\vec{s}; \Theta) = \frac{p(\vec{s}, \vec{y}^{(n)} | \Theta)}{\sum_{\vec{s}' \in \mathcal{K}_n} p(\vec{s}', \vec{y}^{(n)} | \Theta)} \delta(\vec{s} \in \mathcal{K}_n), \quad (25)$$

where  $\delta(\vec{s} \in \mathcal{K}_n)$  is an indicator function, i.e.,  $\delta(\vec{s} \in \mathcal{K}_n) = 1$  if  $\vec{s} \in \mathcal{K}_n$  and zero otherwise. Note that the definition of  $q_n(\vec{s}; \Theta)$  in (25) neither assumes uni-modality like MAP [Mairal et al., 2009, Lee et al., 2007, Olshausen and Field, 1997] or Gaussian approximations of the posterior [Ribeiro and Opper, 2011, Seeger, 2008], nor does it assume a-posteriori independence of the latents as factored approximations [Jordan et al., 1999, Goodfellow et al., 2012, Titsias and Lazaro-Gredilla, 2011]. Moreover, the approximation can inherently exploit the fact that the continuous part of the latents can be integrated out. The basic assumption behind the approximation in (25) is that the posterior mass is sufficiently concentrated in  $\mathcal{K}_n$  – an assumption that in general holds for data that are assumed to be generated by sparse latent causes.

It is shown by Lücke and Eggert [2010] that for appropriately defined subspaces  $\mathcal{K}_n$  the KL-divergence between the true posteriors and their truncated approximations converges to values close to zero. For our approach the subsets  $\mathcal{K}_n$  are defined based on index sets  $I_n \subseteq \{1, \dots, H\}$  which contain the  $H'$  most relevant dimensions for the corresponding data points  $\vec{y}^{(n)}$ :

$$\mathcal{K}_n = \{\vec{s} \mid \sum_h s_h \leq \gamma \text{ and } \forall h \notin I_n : s_h = 0\} \cup \mathcal{U}, \quad (26)$$

where  $I_n$  contains those  $H'$  hidden dimensions with highest values of a selection (or scoring) function  $\mathcal{S}_h(\vec{y}^{(n)}, \Theta)$  (which we define later). The set  $\mathcal{U}$  is defined as  $\mathcal{U} = \{\vec{s} \mid \sum_h s_h = 1\}$  and ensures that  $\mathcal{K}_n$  contains all singleton states [compare Lücke and Eggert, 2010]. Otherwise,  $\mathcal{K}_n$  only contains vectors with at most  $\gamma$  non-zero entries and with non-zero entries only permitted for  $h \in I_n$ . By choosing an appropriate index set  $I_n$  for each data point, Eqn. 25 results in a good approximation to the posterior. In our case of the sparse spike-and-slab model of Eqns. 1 to 3 the truncated approximate posterior is given by:

$$p(\vec{s} \mid \vec{y}^{(n)}, \Theta) \approx q_n(\vec{s}; \Theta) = \frac{\mathcal{N}(\vec{y}^{(n)}; \tilde{W}_{\vec{s}} \vec{\mu}, C_{\vec{s}}) \mathcal{B}(\vec{s}; \vec{\pi})}{\sum_{\vec{s}' \in \mathcal{K}_n} \mathcal{N}(\vec{y}^{(n)}; \tilde{W}_{\vec{s}'} \vec{\mu}, C_{\vec{s}'}) \mathcal{B}(\vec{s}'; \vec{\pi})} \delta(\vec{s} \in \mathcal{K}_n). \quad (27)$$

To obtain some intuition for the approximation in the case of spike-and-slab sparse coding note that by far most generated data points lie close to low-dimensional hyper-planes spanned by the models' basis functions. The posterior mass given such data points is consequently concentrated on low-dimensional hyper-planes spanned by a small number of hidden dimensions (compare Fig. 1, upper-right). The posterior mass in higher-dimensional subspaces is negligible. If the hidden dimensions in  $I_n$  are appropriately selected, the variational approximation (27) approximates the true posterior with high accuracy by matching its distributions on the low-dimensional hyper-planes. To ensure high accuracy, the dimensionality of these hyper-planes (approximation parameter  $\gamma$ ) will be chosen relatively high compared to the average number of contributing hidden dimensions.

The truncated approximation (27) can now be used to compute the expectation values which are required for the M-step equations. If we use the variational distributions in Eqn. 27 for  $q_n(\vec{s}; \Theta)$  on the right-hand-sides of Eqns. 21 to 24, we obtain:

$$\sum_{\vec{s}} q_n(\vec{s}; \Theta) f(\vec{s}) = \frac{\sum_{\vec{s} \in \mathcal{K}_n} \mathcal{N}(\vec{y}^{(n)}; \tilde{W}_{\vec{s}} \vec{\mu}, C_{\vec{s}}) \mathcal{B}(\vec{s}; \vec{\pi}) f(\vec{s})}{\sum_{\vec{s}' \in \mathcal{K}_n} \mathcal{N}(\vec{y}^{(n)}; \tilde{W}_{\vec{s}'} \vec{\mu}, C_{\vec{s}'}) \mathcal{B}(\vec{s}'; \vec{\pi})}, \quad (28)$$

where  $f(\vec{s})$  denotes any of the (possibly parameter dependent) functions of (21) to (24). Instead of having to compute sums over the entire binary state space with  $2^H$  states, only sums over subsets  $\mathcal{K}_n$  have to be computed. Since for many applications the posterior mass is finally concentrated in small volumes of the state space, the approximation quality can stay high even for relatively small sets  $\mathcal{K}_n$ .

## 4.1 Computational Complexity

The truncated E-step defined by (21) to (24) with (28) scales with the approximation parameters  $\gamma$  and  $H'$  which can be defined independently of the latent dimensionality  $H$ . The complexity scales as  $\mathcal{O}(N \sum_{\gamma'=0}^{\gamma} \binom{H'}{\gamma'} (D + \gamma')^3)$ , where the  $D^3$  term can be dropped from the cubic expansion if the observed noise  $\Sigma$  is considered to be diagonal or homoscedastic (i.e.,  $\Sigma = \sigma^2 I$ ). Also note that the truncated approximation yields sparse matrices in Eqns. 25 and 27 which results in more efficient and tractable matrix operations.

Although the total number of data points  $N$  above defines a theoretical upper bound, in practice we can further benefit from the preselection step of the truncated approach to achieve significantly improved runtime performance. Clustering the data points using the index sets  $I_n$  saves us from redundantly performing various computationally expensive operations involved in Eqns. 19 and 27, that given a state  $\vec{s} \in \mathcal{K}_n$  are independent of individual data points sharing the same subspace  $\mathcal{K}_n$ . Furthermore, such a batch processing strategy is also readily parallelizable as the truncated E-step can be performed independently for individual data clusters (see Appendix C for details). Using the batch execution mode we have observed an average runtime speedup of up to an order of magnitude.

## 4.2 Selection function

To compose appropriate subsets  $\mathcal{K}_n$  a selection function  $\mathcal{S}_h(\vec{y}^{(n)}, \Theta)$  is defined, which prior to each E-step selects the relevant hidden dimensions (i.e., the elements of the index set  $I_n$ ) that are most likely to have contributed to the data point  $\vec{y}^{(n)}$ . Selection functions can be based on upper-bounds for probabilities  $p(s_h = 1 | \vec{y}^{(n)}, \Theta)$  [compare Lücke and Eggert, 2010, Puertas et al., 2010] or deterministic functions such as the scalar product between a basis vector and a data point [derived from noiseless limits applied to observed space; compare Lücke and Eggert, 2010].

For the sparse coding model under consideration we define a selection function as follows:

$$\mathcal{S}_h(\vec{y}^{(n)}, \Theta) = \mathcal{N}(\vec{y}^{(n)}; \tilde{W}_{\vec{s}_h} \vec{\mu}, C_{\vec{s}_h}) \propto p(\vec{y}^{(n)} | \vec{s} = \vec{s}_h, \Theta), \quad (29)$$

where  $\vec{s}_h$  represents a singleton state in which only the entry  $h$  is non-zero. The selection function (29) is basically the data likelihood given a singleton state  $\vec{s}_h$ . The function does not take into account the probability of the state itself (i.e.,  $p(\vec{s}_h | \Theta)$ ), as this may introduce a bias against less active latent dimensions. The accuracy of the selection function will directly be evaluated in the next section.

Equations (25) to (27) replace the computation of the expectation values w.r.t. the exact posterior, and represent the approximate EM algorithm used in the experiments section. The algorithm will be applied without any further mechanisms such as annealing as we found it to be very robust in the form derived above. Furthermore, no data preprocessing such as mean subtraction or variance normalization will be used in any of the experiments. To distinguish the algorithm from others in comparative experiments, we will refer to it as *Gaussian Sparse Coding* (GSC) algorithm in order to emphasize the special Gaussian case of the spike-and-slab model used.

## 5 Numerical Experiments

We investigate the performance of the GSC algorithm on artificial data as well as various realistic source separation and denoising benchmarks. For all experiments the algorithm was implemented to run on arrays of CPU nodes as described in [Bornschein et al., 2010]. We extended the technique to make our implementation more efficient and suitable for parallelization by making use of the observation discussed in section 4.1 and Appendix C. In all the experiments, the GSC parameters were randomly initialized<sup>2</sup>. The choice of GSC truncation parameters  $H'$  and  $\gamma$  is in general straight-forward: the larger they are the closer the match to exact EM but the higher are also the computational costs. The parameters are hence capped by the available computational resources. However, empirically we observed that often much smaller than maximally affordable values were sufficient<sup>3</sup>. Note that factored variational approaches do usually not allow to adjust the accuracy / complexity trade-off using a simple parameter change.

### 5.1 Reliability of the Selection Function

To assess the reliability of the selection function we perform a number of experiments on small scale artificial data generated by the model, such that we can compute both the exact (16) and truncated (27) posteriors. To control for the quality of the truncated posterior approximation – and thus the selection function – we compute the ratio between posterior mass within the truncated space  $\mathcal{K}_n$

---

<sup>2</sup>We randomly and uniformly initialized the  $\pi_h$  between 0.05 and 0.95.  $\bar{\mu}$  was initialized with normally distributed random values and the diagonal of  $\Psi$  was initialized with strictly positive uniformly distributed random values. We set  $\Sigma$  to the covariance across the data points, and the elements of  $W$  were independently drawn from a normal distribution with zero mean and unit variance.

<sup>3</sup>Compare Appendix B for trade-off between complexity and accuracy of the truncated approach

and the overall posterior mass [compare Lücke and Eggert, 2010]:

$$Q^{(n)} = \frac{\sum_{\vec{s} \in \mathcal{K}_n} \int_{\vec{z}} p(\vec{s}, \vec{z} | \vec{y}^{(n)}, \Theta) d\vec{z}}{\sum_{\vec{s}' \in \mathcal{K}_n} \int_{\vec{z}'} p(\vec{s}', \vec{z}' | \vec{y}^{(n)}, \Theta) d\vec{z}'} = \frac{\sum_{\vec{s} \in \mathcal{K}_n} \mathcal{B}(\vec{s}; \vec{\pi}) \mathcal{N}(\vec{y}^{(n)}; \tilde{W}_{\vec{s}} \vec{\mu}, C_{\vec{s}})}{\sum_{\vec{s}' \in \mathcal{K}_n} \mathcal{B}(\vec{s}'; \vec{\pi}) \mathcal{N}(\vec{y}^{(n)}; \tilde{W}_{\vec{s}'} \vec{\mu}, C_{\vec{s}'})}, \quad (30)$$

where the integrals over the latent  $\vec{z}$  in (30) are again given in closed-form. Note that  $Q^{(n)}$  ranges from zero to one and that it is directly related [see Lücke and Eggert, 2010] to the KL-divergence between the approximation  $q_n$  in Eqn. 27 and the true posterior:

$$D_{KL}(q_n(\vec{s}, \vec{z}; \Theta), p(\vec{s}, \vec{z} | \vec{y}^{(n)}, \Theta)) = -\log(Q^{(n)}). \quad (31)$$

If  $Q^{(n)}$  is close to one, the KL-divergence is close to zero.

Data for the control experiments were generated by linearly superimposing basis functions that take the form of horizontal and vertical bars [see e.g., Földiák, 1990, Hoyer, 2002] on a  $D = D_2 \times D_2$  pixel grid, where  $D_2 = H/2$ . This gives us  $D_2$  possible horizontal as well as vertical locations for bars of length  $D_2$ , which together form our generating bases  $W^{\text{gen}}$ . Each bar is then randomly assigned either a positive or negative intensity of 10. We set the sparsity such that there are on average two active bars per data point i.e.,  $\pi_h^{\text{gen}} = 2/H$  for all  $h \in H$ . We assume homoscedastic observed noise  $\Sigma^{\text{gen}} = \sigma^2 I_D$ , where  $\sigma^2 = 2$ . The mean of the generating slab is i.i.d. drawn from a Gaussian:  $\vec{\mu}^{\text{gen}} \sim \mathcal{N}(0, 5)$ , and the covariance of the slab is  $\Psi^{\text{gen}} = I_H$ . We generate  $N = 1000$  data points. We run experiments with different sets of values for the truncation parameters  $(H', \gamma) \in \{(4, 4), (5, 4), (5, 3)\}$  for each  $H \in \{10, 12\}$ . Each run consisted of 50 EM iterations and after each run we computed the Q-value over all the data points. For all the experiments we find the average Q-values to be above 0.99, which shows that the state subspaces (26) constructed from the  $H'$  latents chosen through the selection function (29) contain almost the entire posterior probability mass in this case.

## 5.2 Recovery of Sparse Directions on Synthetic Data

First we assess the performances both the GSC approach (using the truncated variational approximation) and MTMKL approach (which uses a factored variational approximation) on synthetic data generated by standard sparse coding models. In one set of experiments we generate data using sparse coding with Cauchy prior [Olshausen and Field, 1996], and in a second set of experiments we use sparse coding with a Laplace prior [Olshausen and Field, 1996, Lee et al., 2007]. For each trial in an experiment a new mixing matrix  $W^{\text{gen}}$  was generated without any constraints on the sparse directions (i.e., matrices were non-orthogonal in general). In both sets of experiments we simultaneously vary both the observed and latent dimensions  $D$  and  $H$  between 20 and 100, and

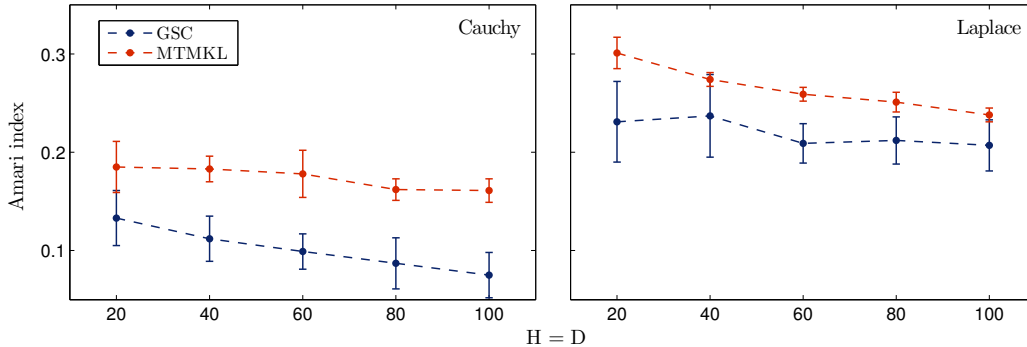


Figure 2: Performance of GSC (with  $H' = \gamma = \frac{H}{10}$ ) vs. MTMKL on data generated by standard sparse coding models both with Cauchy and Laplace priors. Performance compared on the Amari index (32).

repeat 15 trials per given dimensionality. For each trail we randomly generated a new dataset with  $N = 5000$ . Per trial, we perform 50 iterations of both algorithms. The GSC truncation parameters  $H'$  and  $\gamma$  were set to  $\frac{H}{10}$ . We assess the performances of the algorithms in terms of how well the bases  $W$  inferred by each of them align with the ground truth bases  $W^{\text{gen}}$ . As a measure of discrepancy between the generating and the recovered bases we use the Amari index [Amari et al., 1996]:

$$A(W) = \frac{1}{2H(H-1)} \sum_{h,h'=1}^H \left( \frac{|O_{hh'}|}{\max_{h''} |O_{hh''}|} + \frac{|O_{hh'}|}{\max_{h''} |O_{h''h'}|} \right) - \frac{1}{H-1}, \quad (32)$$

where  $O_{hh'} = (W^{-1}W^{\text{gen}})_{hh'}$ . Note that the Amari index is either positive or zero. It is zero only when the basis vectors of  $W$  and  $W^{\text{gen}}$  represent the same set of orientations, which in our case implies a precise recovery of the (ground truth) sparse directions.

The results for GSC and MTMKL in Fig. 2 show that both approaches do relatively well in recovering the sparse directions, which shows that they are robust against the model mismatch imposed by generating from models with other priors. Furthermore, we observe that the GSC approach consistently recovers the sparse directions more accurately.

### 5.3 Source Separation

On synthetic data we have seen that spike-and-slab sparse coding can effectively recover sparse directions such as those generated by standard sparse coding models. As many signals such as acoustic speech data are sparse, and as different sources mix linearly, the assumptions of sparse coding match such data well. Source separation is consequently a natural application domain of

sparse coding approaches, and well suited for benchmarking novel spike-and-slab as well as other sparse coding algorithms. To systematically study the a-posteriori independence assumption in factored variational approaches, we monitor the recovery of sparse directions of GSC and MTMKL for an increasing degree of the mixing matrix’s non-orthogonality. Fig. 3 shows the performance of both the methods based on three different source separation benchmarks obtained from [ICALAB Cichocki et al., 2007]. The error bars are generated by repeating 15 trials per experiment. The x-axis in the figure represents the degree of orthogonality of the ground truth mixing bases  $W^{\text{gen}}$ . Starting from strictly orthogonal at the left, the bases were made increasingly non-orthogonal by randomly generating orthogonal bases and adding Gaussian distributed noise to them with  $\sigma = (4, 10, 20)$  respectively. For Fig. 3 no observation noise was added to the mixed sources. For both the algorithms we performed 100 iterations per run<sup>4</sup>. The GSC truncation parameters  $H'$  and  $\gamma$  were set to 10 for all the following experiments, therefore for *10halo* the GSC inference was exact. As can be observed, both approaches recover the sparse directions well. While performance on the EEG19 dataset is the same, GSC consistently performs better than MTMKL on *10halo* and *Speech20*. If observation noise is added, the difference can become still more pronounced for some datasets. Fig. 4 shows the performance in the case of *Speech20* (with added Gaussian noise with  $\sigma = 2.0$ ), for instance. Along the x-axis orthogonality increases, again. While the performance of MTMKL decreases with decreasing orthogonality, performance of GSC increases in this case. For other datasets increased observation noise may not have such effects, however (see Appendix, Fig. 10 for two examples).

Next we look at MAP based sparse coding algorithms for the source separation task. Publicly available methods which we compare with are the lasso based least angle regression [SPAMS; Mairal et al., 2009] and the efficient sparse coding algorithms [ESCA; Lee et al., 2007] with epsilonL<sub>1</sub>-penalty<sup>5</sup>. The performance is tested on another set of ICALAB [Cichocki et al., 2007] benchmarks used previously [Suzuki and Sugiyama, 2011, Lücke and Sheikh, 2012]. Following Suzuki and Sugiyama [2011] we use  $N = 200$  and  $N = 500$  data points from each benchmark and generate observed data by mixing the benchmark sources with randomly generated orthogonal bases and adding no noise to the observed data. For each experiment we performed 50 trials with a new randomly generated orthogonal data mixing matrix  $W^{\text{gen}}$  and new parameter initialization in each trial. The GSC inference was exact for these experiments with better results obtained with observed

---

<sup>4</sup>For the MTMKL algorithm we observed convergence after 100 iterations while the GSC algorithm continued to improve with more iterations. However, the reported results are obtained with 100 iterations.

<sup>5</sup>For both the algorithms, optimal values for regularization parameters were chosen through cross-validation.



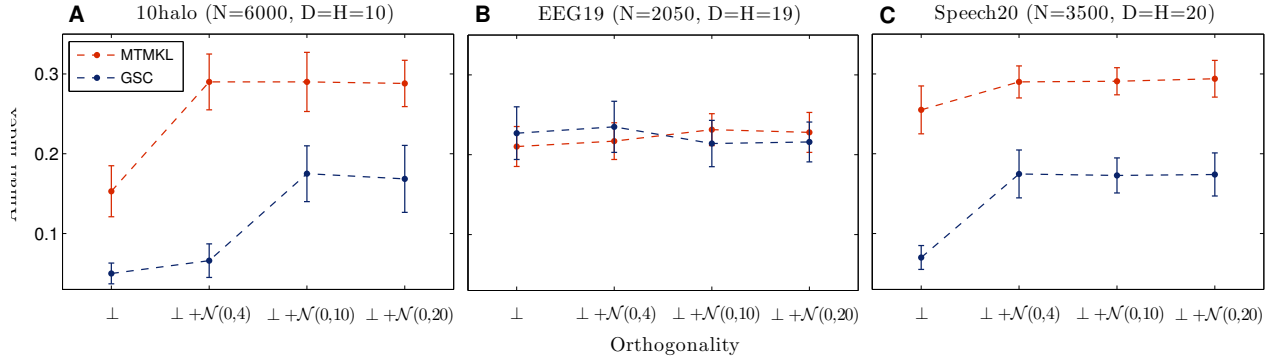


Figure 3: Performance of GSC vs. MTMKL on source separation benchmarks with varying degrees of orthogonality of the mixing bases. The orthogonality on the x-axis varies from being orthogonal  $\perp$  to increasingly non-orthogonal mixing as randomly generated orthogonal bases are perturbed by adding Gaussian noise  $\mathcal{N}(0, \sigma)$  to them. No observation noise was assumed for these experiments. Performances are compared on the Amari index (32).

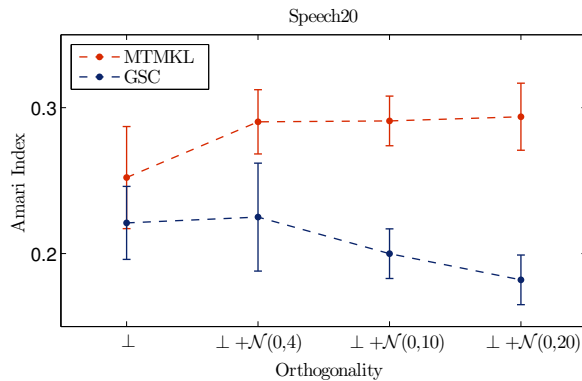


Figure 4: Performance of GSC vs. MTMKL in terms of the Amari index (32) on the Speech20 benchmark with varying degrees of orthogonality of the mixing bases and Gaussian noise (with  $\sigma = 2$ ) added to observed data. The orthogonality on the x-axis varies from being orthogonal  $\perp$  to increasingly non-orthogonal mixing as randomly generated orthogonal bases are perturbed by adding Gaussian noise  $\mathcal{N}(0, \sigma)$  to them.

datasets			Amari index - mean (std.)			
name	H = D	N	GSC	MTMKL	SPAMS	ESCA
10halo	10	200	0.27(.04)	<b>0.21(.05)</b>	0.28(0)	0.31(.02)
		500	<b>0.17(.03)</b>	0.20(.03)	0.29(0)	0.29(.02)
Sergio7	7	200	<b>0.19(.05)</b>	<b>0.19(.03)</b>	<b>0.18(0)</b>	0.27(.04)
		500	<b>0.13(.04)</b>	0.23(.04)	0.19(0)	0.18(.04)
Speech4	4	200	<b>0.13(.04)</b>	<b>0.14(.03)</b>	0.18(0)	0.23(.02)
		500	<b>0.10(.04)</b>	0.14(.08)	0.16(0)	0.17(0)
c5signals	5	200	0.29(.08)	<b>0.24(.08)</b>	0.39(0)	0.47(.05)
		500	<b>0.31(.06)</b>	<b>0.32(.03)</b>	0.42(0)	0.48(.05)

Table 1: Performance of GSC, MTMKL and other publicly available sparse coding algorithms on benchmarks for source separation. Performances are compared based on the Amari index (32). Bold values highlight the best performing algorithm(s).

noise constrained to be homoscedastic<sup>6</sup>. We performed up to 350 iterations of the GSC algorithm (with more iterations continuing to improve the performance) while for the other algorithms we observed convergence between 100 and 300 iterations.

Tab. 1 lists the performances of the algorithms. As can be observed, the spike-and-slab based models perform better than the standard sparse coding models for all except of one experiment (Sergio7, 200 data points) where SPAMS performs comparable well (or slightly better). Among the spike-and-slab models, GSC performs best for all settings with 500 data points, while MTMKL is better in two cases for 200 data points<sup>7</sup>. Note that further improvements on some settings in Tab. 1 can be obtained by algorithms constrained to assume orthogonal bases [Suzuki and Sugiyama, 2011, Lücke and Sheikh, 2012]. However, for *10halo* and *speech4* GSC and MTMKL are better without such an explicit constraint.

<sup>6</sup>To infer homoscedastic noise we set in the M-step the updated noise matrix  $\Sigma$  to  $\sigma^2 I_D$  where  $\sigma^2 = \text{Tr}(\Sigma)/D$ .

<sup>7</sup>By considering Tab. 1 performance not necessarily increases with an increased number of data points. Note in this respect, however, that the used data points are not independent samples. Following Suzuki and Sugiyama [2011] we always took consecutive 200 or 500 data points (after an offset) from each of the benchmarks. Therefore, due to time-dependencies in the signals, the underlying data point statistics change with the number of data points.

## 5.4 Computational Complexity vs. Performance

In terms of complexity, GSC and MTMKL algorithms are significantly different, so we also looked at the trade-off between their computational costs versus performance. Fig.5 shows performance against compute time for both algorithms. The error bars for the Speech20 plot were generated from 15 trials per experiment. For MTMKL we obtained the plot by increasing the number of iterations from 50 to 100 and 1000, while for the GSC plot we performed 100 iterations with  $H' = \gamma \in [2, 3, 5, 7, 10]$ . For the image denoising task (described next), the MTMKL plot was generated from a run with  $H = 64$  latents<sup>8</sup> and the number of iterations going up to 12K. The GSC plot was generated from  $H = 400$  latents and  $H'$  and  $\gamma$  being 10 and 5 respectively. Up to 120 iterations were performed to obtain the last point for the GSC plot. As can be observed for both tasks, the performance of MTMKL saturates from certain runtime values onwards. GSC on the other hand continues to show improved performance with increasing computational resources.

## 5.5 Image Denoising

Finally, we investigate performance of the GSC algorithm on the standard “house” benchmark for denoising which has been used for the evaluation of similar approaches [e.g., Li and Liu, 2009, Zhou et al., 2009] including the MTMKL spike-and-slab approach. The MTMKL approach currently represents the state-of-the-art on this benchmark [see Titsias and Lazaro-Gredilla, 2011]. For the task an input noisy image is generated by adding Gaussian noise (with zero mean and standard deviation determining the noise level) to the  $256 \times 256$  image (see Fig. 6). Following the previous studies, we generated 62,001 overlapping (shifted by 1 pixel)  $8 \times 8$  patches from the noisy image. We then applied 65 iterations of the GSC algorithm for  $H \in \{64, 256\}$  for different noise levels  $\sigma \in \{15, 25, 50\}$ . The truncation parameters  $H'$  and  $\gamma$  for each run are listed in Tab.2. We assumed homoscedastic observed noise with a priori unknown variance in all these experiments (as the MTMKL model). Tab.2 compares the denoising results of different algorithms in terms of the peak signal-to-noise (PSNR) ratio. We found that for the low noise level ( $\sigma = 15$ ) GSC is competitive with other approaches but with MTMKL performing slightly better. For the higher noise levels of  $\sigma = 25$  and  $\sigma = 50$ , GSC outperforms all the other approaches including the MTMKL approach that represented the state-of-the-art. In Fig. 6 we show our result for noise level 25. The figure contains both the noisy and the GSC denoised image along with the inferred sparsity vector  $\vec{\pi}$  and all bases with appearance probabilities significantly larger than zero (sorted from high such

---

<sup>8</sup>Titsias and Lazaro-Gredilla [2011] report that for the denoising task they observe no performance improvements for larger number of latents.

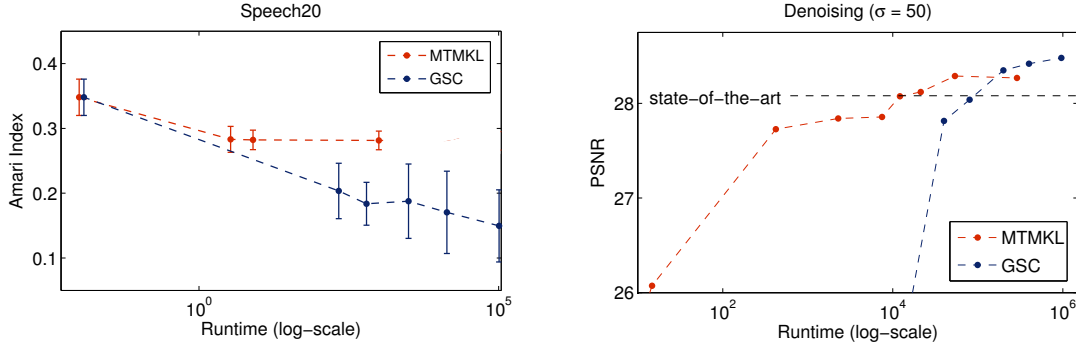


Figure 5: Runtime vs. performance comparison of GSC and MTMKL on source separation and denoising tasks. Source separation is compared on the Amari index (the lower the better) while the denoising is compared on the peak signal-to-noise (PSNR) ratio (the higher the better).

probabilities to low ones). We also tried GSC with higher numbers of latent dimensions. Although for low noise levels of  $\sigma = 15$  and  $\sigma = 25$  we did not notice significant improvements, we observed a further increase for  $\sigma = 50$ . For instance, for  $H = 400$ ,  $H' = 10$  and  $\gamma = 8$ , we obtained a PSNR of 28.48dB.

As for source separation described in section 5.4, we also compared performance vs. computational demand of both algorithms for the task of image denoising. As can be observed by considering Fig. 5, MTMKL performs better when computational resources relatively limited. However, when increasingly more computational resources are made available, MTMKL does not improve much further on its performance while GSC continuously increases and eventually outperforms MTMKL on this task.

Noise	PSNR (dB)						
	Noisy img	MTMKL <sup>exp.</sup>	K-SVD <sup>mis.</sup>	*K-SVD <sup>match</sup>	Beta pr.	GSC (H=64)	GSC (H=256)
$\sigma=15$	24.59	<b>34.29</b>	30.67	34.22	34.19	32.68 (H'=10, $\gamma=8$ )	33.78 (H'=18, $\gamma=3$ )
$\sigma=25$	20.22	31.88	31.52	32.08	31.89	31.10 (H'=10, $\gamma=8$ )	<b>32.01</b> (H'=18, $\gamma=3$ )
$\sigma=50$	14.59	28.08	19.60	27.07	27.85	28.02 (H'=10, $\gamma=8$ )	<b>28.35</b> (H'=10, $\gamma=8$ )

Table 2: Comparison of the GSC algorithm with other methods applied to the “house” benchmark. The compared methods are: MTMKL [Titsias and Lazaro-Gredilla, 2011], K-SVD [Li and Liu, 2009], and the method by Zhou et al. [2009]. Bold values highlight the best performing algorithm(s). \*High values for K-SVD matched are not made bold-faced as the method assumes the noise variance to be known a-priori [see Li and Liu, 2009].

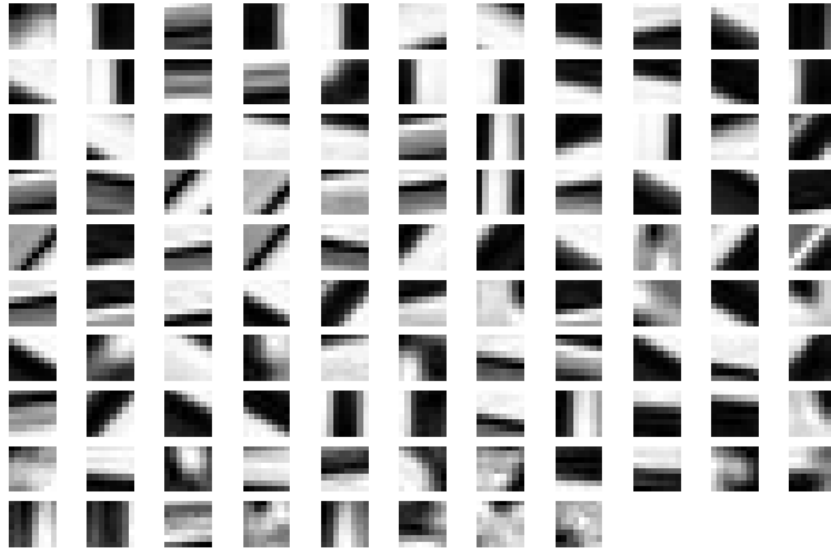
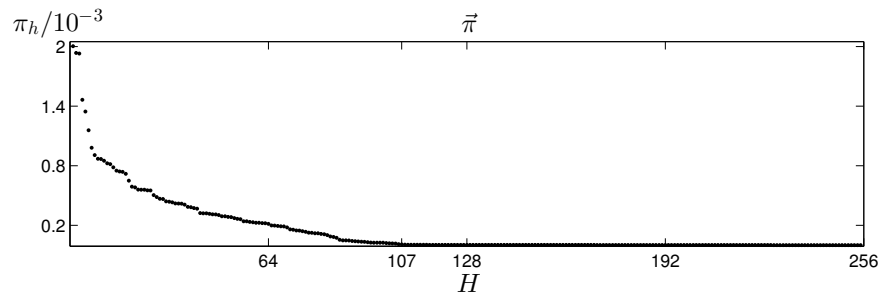


Figure 6: Top left: Noisy “house” image with  $\sigma = 25$ . Top right: GSC denoised image. Middle: Inferred sparsity values  $\pi_h$  in descending order indicate that finally around 107 of in total 256 latent dimensions significantly contribute to model the data. Bottom: Basis functions (ordered from left to right, top to bottom) corresponding to the first 107 latent dimensions sorted w.r.t. the decreasing sparsity values  $\pi_h$ .

## 6 Discussion

The last years have seen a surge in the application of sparse coding algorithms to different research domains, along with developments of new sparse coding approaches with increased capabilities. There are currently different lines of research followed for developing new algorithms: One direction is based on the standard sparse coding algorithm [Olshausen and Field, 1996] with Laplace prior and parameter optimization using maximum a-posteriori (MAP) estimates of the latent posteriors for efficient training. This original approach has since been made more efficient and precise. Many sparse coding algorithms based on the MAP estimation are continuously being developed and successfully applied in a variety of settings [e.g., Lee et al., 2007, Mairal et al., 2009]. Another line of research aims at a fully Bayesian description of sparse coding and emphasizes greater flexibility by using different (possibly non-Gaussian) noise models and estimations of the number of hidden dimensions. The great challenge of these general models is the procedure of parameter estimation. For instance, the model in [Mohamed et al., 2012] uses Bayesian methodology involving conjugate priors and hyper-parameters in combination with Laplace approximation and different sampling schemes.

A line of research falling in between conventional and fully Bayesian approaches is represented by the truncated variational approach studied here and by other very recent developments [Titsias and Lazaro-Gredilla, 2011, Goodfellow et al., 2012]. While these approaches are all based on spike-and-slab generalizations of sparse coding as fully Bayesian approaches, they maintain deterministic approximation procedures for parameter optimization. Variational approximations allow for applications to large hidden spaces, posing a challenge for sampling approaches especially in cases of multi-modal posteriors. Using the novel and existing approaches in different experiments of this study, we have confirmed the advantages of spike-and-slab priors for sparse coding, and the scalability of variational approximations for such models. The newly developed truncated variational algorithm scales almost linearly with the number of hidden dimensions for fixed truncation parameters (see for instance the scaling behavior in supplemental Fig. 8 for  $H$  going up to 1024). The MTMKL algorithm by Titsias and Lazaro-Gredilla [2011] has been applied on the same scale. Using a similar approach also based on factored distributions Goodfellow et al. [2012] report results for up to a couple of thousands latent dimensions (albeit on small input dimensions and having a more constrained generative model). Sampling based algorithms for non-parametric and fully Bayesian approaches are more general but have not been applied to such large scales.

A main focus of this work and reasoning behind the algorithm’s development is due to the long-known biases introduced by factored variational approximations [Ilin and Valpola, 2003, MacKay,

2001, Turner and Sahani, 2011]. Our systematic comparison of the GSC algorithm to the method by Titsias and Lazaro-Gredilla [2011] confirms the earlier observation [Ilin and Valpola, 2003] that factored variational approaches are biased towards orthogonal bases. If we compare the performance of both algorithms on the recovery of non-orthogonal sparse directions, the performance of the factored variational approach is consistently lower than the performance of the truncated variational algorithm (Fig. 2). The same applies for experiments for unmixing real signals in which we increased the non-orthogonality (Fig. 3A,C; suppl. Fig.10); although for some data performance is very similar (Fig. 3B). Also if sources are mixed orthogonally, we usually observe better performance of the truncated variational approach (Tab. 1), which is presumably due to the more general underlying prior (i.e., a fully parameterized Gaussian slab). Overall, GSC is the best performing algorithm on source separation tasks involving non-orthogonal sparse directions [compare Suzuki and Sugiyama, 2011, for algorithms constrained to orthogonal bases]. For some datasets with few data points, we observed an equal or better performance of the MTMKL approach, which can be explained by their Bayesian treatment of the model parameters (see Tab. 1, performance with 200 data points). Notably, both approaches are consistently better on source separation benchmarks than the standard sparse coding approaches SPAMS [Mairal et al., 2009] and ESCA [Lee et al., 2007] (see Tab. 1). This may be taken as evidence for the better suitability of a spike-and-slab prior for such types of data.

Finally, we compared the performance of factored and truncated variational approximations on a standard denoising task (see Tab. 2). The high PSNR values observed for both approaches again in general speak for the strengths of spike-and-slab sparse coding. The MTMKL model represented the state-of-the-art on this benchmark, so far. Differences of MTMKL to previous approaches are small, but this is due to the nature of such long-standing benchmarks (compare, e.g., the MNIST data set). For the same denoising task with standard noise levels of  $\sigma = 25$  and  $\sigma = 50$  we found the GSC model to further improve the state-of-the-art (compare Tab. 2 with data by Li and Liu [2009], Zhou et al. [2009], Titsias and Lazaro-Gredilla [2011]). While we observed a continuous increase of performance with the number of hidden dimensions used for GSC, the MTMKL algorithm [Titsias and Lazaro-Gredilla, 2011] is reported to reach saturation at  $H = 64$  latent dimensions. As the learned sparse directions become less and less orthogonal the more overcomplete the setting gets, this saturation may again be due to the bias introduced by the factored approach. GSC with  $H = 256$  improves the state-of-the-art with 32.01dB for  $\sigma = 25$  and with 28.35dB for  $\sigma = 50$  (with even higher PSNR for  $H = 400$ ). Because of assuming an independent Bernoulli prior per latent dimension, GSC can also prune out latent dimensions by inferring very low values of  $\pi_h$  for the

bases that make negligible contribution in the inference procedure. This can be observed in Fig. 6, where for the application of GSC to the denoising task with  $\sigma = 25$ , we found only about 107 of the 256 basis functions to have significant probabilities to contribute to the task. This means that GSC with about 100 basis functions can be expected to achieve almost the same performance as GSC with 256 basis functions. However, in practice we observed that the average performance increases with more basis functions because local optima can more efficiently be avoided. Note that this observation is not limited to the particular approach studied here; also for other approaches to sparse learning, efficient avoidance of local optima has been reported if the number of assumed hidden dimensions was increased [e.g. Spratling, 2006, Lücke and Sahani, 2008]. In comparison to MTMKL, GSC can make use of significantly more basis functions. It uses about 100 functions while MTMKL performance saturates at about 64 as mentioned previously. On the other hand, we found MTMKL to perform better on the low noise level setting (see  $\sigma = 15$  in Tab. 2) or when relatively limited computational resources are available (see Fig. 5).

In conclusion, we have studied a novel learning algorithm for sparse coding with spike-and-slab prior and compared it with a number of sparse coding approaches including other spike-and-slab based methods. The results we obtained show that the truncated variational approach is a competitive method. It shows that posterior dependencies and multi-modality can be captured by a scalable variational approach. Furthermore, the direct comparison with a factored approach in source separation experiments also confirms earlier observations that assumptions of a-posteriori independence introduces biases, and that avoiding such biases, e.g. by a truncated approach, improves the state-of-the-art on source separation benchmarks as well as on standard denoising tasks. However, we also find that under certain constraints and settings, factored variational learning for spike-and-slab sparse coding may perform as good or better. In general, our results argue in favor of spike-and-slab sparse coding models and recent efforts for developing improved algorithms for inference in such models.

**Acknowledgements.** We acknowledge support from the German Research Foundation (DFG) in the project LU 1196/4-2 (J. Lücke), the German Federal Ministry of Education and Research (BMBF), project 01GQ0840 (A.-S. Sheikh and J. Shelton). Furthermore, we acknowledge support by the Frankfurt Center for Scientific Computing (CSC).



## References

- S. Amari, A. Cichocki, and H. H. Yang. A new learning algorithm for blind signal separation. In *NIPS*, pages 757–763. MIT Press, 1996.
- H. Attias. Independent factor analysis. *Neural Computation*, 11:803 – 851, 1999.
- C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- J. Bornschein, Z. Dai, and J. Lücke. Approximate EM learning on large computer clusters. In *NIPS Workshop: LCCC*. 2010.
- P. Carbonetto and M. Stephen. Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies. *Bayesian Analysis Journal*, 2011.
- C. M. Carvalho, J. Chang, J. E. Lucas, J. R. Nevins, Q. Wang, and M. West. High-dimensional sparse factor modeling: Applications in gene expression genomics. *Journal of the American Statistical Association*, 103(484):1438–1456, 2008.
- A. Cichocki, S. Amari, K. Siwek, T. Tanaka, A.H. Phan, and R. Zdunek. ICALAB-MATLAB Toolbox Version 3, 2007.
- Z. Dai and J. Lücke. Autonomous cleaning of corrupted scanned documents – a generative modeling approach. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2012.
- P. Földiák. Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, 64:165–170, 1990.
- P. Garrigues and B. A. Olshausen. Learning horizontal connections in a sparse coding model of natural images. In *NIPS*, 2007.
- I. Goodfellow, A. Courville, and Y. Bengio. Large-scale feature learning with spike-and-slab sparse coding. In *ICML*, 2012.
- 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 557–65. 2002.
- A. Ilin and H. Valpola. On the effect of the form of the posterior approximation in variational learning of ica models. In *Proceedings ICA*, pages 915–920, 2003.
- M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

- D. Knowles and Z. Ghahramani. Infinite sparse factor analysis and infinite independent components analysis. In *Proc. ICA*, pages 381–388, 2007.
- D. Knowles and Z. Ghahramani. Nonparametric Bayesian sparse factor models with application to gene expression modeling. *The Annals of Applied Statistics*, 5(2B):1534–1552, June 2011.
- H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*, volume 20, pages 801–08, 2007.
- H. Li and F. Liu. Image denoising via sparse and redundant representations over learned dictionaries in wavelet domain. *ICIG '09*, pages 754–758, 2009.
- J. Lücke and J. Eggert. Expectation truncation and the benefits of preselection in training generative models. *Journal of Machine Learning Research*, 11:2855–900, 2010.
- J. Lücke and M. Sahani. Maximal causes for non-linear component extraction. *Journal of Machine Learning Research*, 9:1227–67, 2008.
- J. Lücke and A. S. Sheikh. Closed-form EM for sparse coding and its application to source separation. In *Proc. LVA/ICA*, LNCS, pages 213–221, 2012.
- D. J. C. MacKay. Local minima, symmetry-breaking, and model pruning in variational free energy minimization. 2001. Online publication: [www.inference.phy.cam.ac.uk/mackay/minima.ps.gz](http://www.inference.phy.cam.ac.uk/mackay/minima.ps.gz).
- J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *ICML*, page 87, 2009.
- T. J. Mitchell and J. J. Beauchamp. Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404):1023–1032, 1988.
- S. Mohamed, K. Heller, and Z. Ghahramani. Evaluating Bayesian and L1 approaches for sparse unsupervised learning. In *ICML*, 2012.
- 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.
- B. Olshausen and D. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–9, 1996.

- B. Olshausen and D. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, December 1997. ISSN 00426989. doi: 10.1016/S0042-6989(97)00169-7.
- B. Olshausen and K. Millman. Learning sparse codes with a mixture-of-Gaussians prior. *Advances in Neural Information Processing Systems*, 12:841–847, 2000.
- J. Paisley and L. Carin. Nonparametric factor analysis with beta process priors. ICML '09, pages 777–784, 2009.
- G. Puertas, J. Bornschein, and J. Lücke. The maximal causes of natural scenes are edge filters. In *Advances in Neural Information Processing Systems*, volume 23, pages 1939–47. 2010.
- M. Rattray, O. Stegle, K. Sharp, and J. Winn. Inference algorithms and learning theory for Bayesian sparse factor analysis. 012002, 2009. doi: 10.1088/1742-6596/197/1/012002.
- F. Ribeiro and M. Opper. Expectation propagation with factorizing distributions: A gaussian approximation and performance results for simple models. *Neural Computation*, 23:10471069, 2011. ISSN 0899-7667.
- M. Seeger. Bayesian inference and optimal design for the sparse linear model. *Journal of Machine Learning Research*, 9:759–813, June 2008. ISSN 1532-4435.
- J. Shelton, J. Bornschein, A.-S. Sheikh, P. Berkes, and J. Lücke. Select and sample - a model of efficient neural inference and learning. *Advances in Neural Information Processing Systems*, 24, 2011.
- M. Spratling. Learning image components for object recognition. *Journal of Machine Learning Research*, 7:793–815, 2006.
- T. Suzuki and M. Sugiyama. Least-squares independent component analysis. *Neural Computation*, 23(1):284–301, 2011.
- Y. W. Teh, D. Görür, and Z. Ghahramani. Stick-breaking construction for the indian buffet process. *JMLR*, 2:556–563, 2007.
- M. Titsias and M. Lazaro-Gredilla. Spike and slab variational inference for multi-task and multiple kernel learning. In *NIPS 24*, pages 2339–2347. 2011.
- R. E. Turner and M. Sahani. *Bayesian Time Series Models*, chapter Two problems with variational expectation maximisation for time-series models. Cambridge University Press, 2011.

- M. West. Bayesian factor regression models in the "large p, small n" paradigm. In *Bayesian Statistics*, pages 723–732. Oxford University Press, 2003.
- R. Yoshida and M. West. Bayesian learning in sparse graphical factor models via variational mean-field annealing. *JMLR*, 99:1771–1798, 2010. ISSN 1532-4435.
- M. Zhou, H. Chen, J. Paisley, L. Ren, G. Sapiro, and L. Carin. Non-parametric bayesian dictionary learning for sparse image representations. In *NIPS 22*, pages 2295–2303. 2009.

## Appendix

We give details on derivations and numerical experiments.

### A Derivation of M-step Equations

Our goal is to optimize the free-energy w.r.t.  $\Theta$ :

$$\begin{aligned} \mathcal{F}(\Theta^{\text{old}}, \Theta) &= \sum_{n=1}^N \left\langle \log p(\vec{y}^{(n)}, \vec{s}, \vec{z} | \Theta) \right\rangle_n + H(\Theta^{\text{old}}) \\ &= \sum_{n=1}^N \sum_{\vec{s}} \int_{\vec{z}} p(\vec{s}, \vec{z} | \vec{y}^{(n)}, \Theta^{\text{old}}) \\ &\quad \left[ \log(p(\vec{y}^{(n)} | \vec{s}, \vec{z}, \Theta)) + \log(p(\vec{z} | \vec{s}, \Theta)) + \log(p(\vec{s} | \Theta)) \right] d\vec{z} + H(\Theta^{\text{old}}), \end{aligned}$$

where

$$\begin{aligned} \log(p(\vec{y}^{(n)} | \vec{s}, \vec{z}, \Theta)) &= -\frac{1}{2} (\log(2\pi^D) + \log |\Sigma|) \\ &\quad -\frac{1}{2} \left( \vec{y}^{(n)} - W(\vec{s} \odot \vec{z}) \right)^{\text{T}} \Sigma^{-1} \left( \vec{y}^{(n)} - W(\vec{s} \odot \vec{z}) \right), \\ \log(p(\vec{z} | \vec{s}, \Theta)) &= -\frac{1}{2} \left( \log(2\pi^{|\vec{s}|}) + \log |\Psi \odot \vec{s} \vec{s}^{\text{T}}| \right) \\ &\quad -\frac{1}{2} \left( (\vec{z} - \vec{\mu}) \odot \vec{s} \right)^{\text{T}} (\Psi \odot \vec{s} \vec{s}^{\text{T}})^{-1} \left( (\vec{z} - \vec{\mu}) \odot \vec{s} \right) \end{aligned}$$

$$\text{and} \quad \log(p(\vec{s} | \Theta)) = \sum_{h=1}^H \log(\pi_h^{s_h} (1 - \pi_h)^{1-s_h}).$$

The free-energy thus takes the form:

$$\begin{aligned} \mathcal{F}(\Theta^{\text{old}}, \Theta) &= \sum_{n=1}^N \sum_{\vec{s}} \int_{\vec{z}} q_n(\vec{s}, \vec{z}; \Theta^{\text{old}}) \\ &\quad \left[ -\frac{1}{2} (\log(2\pi^D) + \log |\Sigma|) - \frac{1}{2} \left( \vec{y}^{(n)} - W(\vec{s} \odot \vec{z}) \right)^{\text{T}} \Sigma^{-1} \left( \vec{y}^{(n)} - W(\vec{s} \odot \vec{z}) \right) \right. \\ &\quad -\frac{1}{2} (\log(2\pi^{|\vec{s}|}) + \log |\Psi \odot \vec{s} \vec{s}^{\text{T}}|) \\ &\quad -\frac{1}{2} \left( (\vec{z} - \vec{\mu}) \odot \vec{s} \right)^{\text{T}} (\Psi \odot \vec{s} \vec{s}^{\text{T}})^{-1} \left( (\vec{z} - \vec{\mu}) \odot \vec{s} \right) \\ &\quad \left. + \sum_{h=1}^H \log(\pi_h^{s_h} (1 - \pi_h)^{1-s_h}) \right] d\vec{z} + H(\Theta^{\text{old}}), \end{aligned}$$

where  $q_n(\vec{s}, \vec{z}; \Theta^{\text{old}})$  denotes the posterior  $p(\vec{s}, \vec{z} | \vec{y}^{(n)}, \Theta^{\text{old}})$ . Now we can derive the M-step equations (7) to (11) by canonically setting the derivatives of the free-energy above w.r.t. each parameter in  $\Theta$  to zero.

## Optimization of the Data Noise

Let us start with the derivation of the M-step equation for  $\Sigma$ :

$$\begin{aligned}
& \frac{\partial}{\partial \Sigma} \mathcal{F}(\Theta^{\text{old}}, \Theta) \\
&= \sum_{n=1}^N \sum_{\vec{s}} \int_{\vec{z}} q_n(\vec{s}, \vec{z}; \Theta^{\text{old}}) \\
&\quad \left[ -\frac{1}{2} \frac{\partial}{\partial \Sigma} (\log |\Sigma|) - \frac{1}{2} \frac{\partial}{\partial \Sigma} (\vec{y}^{(n)} - W(\vec{s} \odot \vec{z}))^T \Sigma^{-1} (\vec{y}^{(n)} - W(\vec{s} \odot \vec{z})) \right] d\vec{z} \\
&= \sum_{n=1}^N \sum_{\vec{s}} \int_{\vec{z}} q_n(\vec{s}, \vec{z}; \Theta^{\text{old}}) \\
&\quad \left[ -\frac{1}{2} \Sigma^{-1} + \frac{1}{2} \Sigma^{-2} (\vec{y}^{(n)} - W(\vec{s} \odot \vec{z})) (\vec{y}^{(n)} - W(\vec{s} \odot \vec{z}))^T \right] d\vec{z} \stackrel{!}{=} 0 \\
&\Rightarrow \Sigma = \frac{1}{N} \sum_{n=1}^N \sum_{\vec{s}} \int_{\vec{z}} q_n(\vec{s}, \vec{z}; \Theta^{\text{old}}) \left[ (\vec{y}^{(n)} - W(\vec{s} \odot \vec{z})) (\vec{y}^{(n)} - W(\vec{s} \odot \vec{z}))^T \right] d\vec{z} \\
&= \frac{1}{N} \sum_{n=1}^N \left[ \vec{y}^{(n)} (\vec{y}^{(n)})^T - 2(W \langle \vec{s} \odot \vec{z} \rangle_n) (\vec{y}^{(n)})^T + W (\langle (\vec{s} \odot \vec{z}) (\vec{s} \odot \vec{z})^T \rangle_n) W^T \right],
\end{aligned}$$

where  $\langle \cdot \rangle_n$  denotes the expectation value in Eqn. 6.

## Optimization of the Bases

We will now derive the M-step update for the basis functions  $W$ :

$$\begin{aligned}
& \frac{\partial}{\partial W} \mathcal{F}(\Theta^{\text{old}}, \Theta) \\
&= \sum_{n=1}^N \sum_{\vec{s}} \int_{\vec{z}} q_n(\vec{s}, \vec{z}; \Theta^{\text{old}}) \left[ -\frac{1}{2} \frac{\partial}{\partial W} (\vec{y}^{(n)} - W(\vec{s} \odot \vec{z}))^T \Sigma^{-1} (\vec{y}^{(n)} - W(\vec{s} \odot \vec{z})) \right] d\vec{z} \\
&= \sum_{n=1}^N \sum_{\vec{s}} \int_{\vec{z}} q_n(\vec{s}, \vec{z}; \Theta^{\text{old}}) \left[ -\frac{1}{\Sigma} \left( \vec{y}^{(n)} (\vec{s} \odot \vec{z})^T - W(\vec{s} \odot \vec{z}) (\vec{s} \odot \vec{z})^T \right) \right] d\vec{z} \stackrel{!}{=} 0 \\
&\Rightarrow W = \frac{\sum_{n=1}^N \vec{y}^{(n)} \langle \vec{s} \odot \vec{z} \rangle_n^T}{\sum_{n=1}^N \langle (\vec{s} \odot \vec{z}) (\vec{s} \odot \vec{z})^T \rangle_n}.
\end{aligned}$$

## Optimization of the Sparsity Parameter

Here we take the derivative of the free-energy w.r.t.  $\vec{\pi}$ :

$$\begin{aligned}
\frac{\partial}{\partial \vec{\pi}} \mathcal{F}(\Theta^{\text{old}}, \Theta) &= \sum_{n=1}^N \sum_{\vec{s}} \int_{\vec{z}} q_n(\vec{s}, \vec{z}; \Theta^{\text{old}}) \left[ \frac{\partial}{\partial \vec{\pi}} \left( \vec{s} \log \vec{\pi} + (1 - \vec{s}) \log(1 - \vec{\pi}) \right) \right] d\vec{z} \\
&= \sum_{n=1}^N \sum_{\vec{s}} q_n(\vec{s}; \Theta^{\text{old}}) \left[ \frac{\vec{s}}{\vec{\pi}} - \frac{(1-\vec{s})}{(1-\vec{\pi})} \right] \stackrel{!}{=} 0
\end{aligned}$$

$$\Rightarrow \vec{\pi} = \frac{1}{N} \sum_{n=1}^N \langle \vec{s} \rangle_n.$$

## Optimization of the Latent Mean

Now we derive the M-step update for the mean  $\vec{\mu}$  of the Gaussian slab:

$$\begin{aligned} & \frac{\partial}{\partial \vec{\mu}} \mathcal{F}(\Theta^{\text{old}}, \Theta) \\ &= \sum_{n=1}^N \sum_{\vec{s}} \int_{\vec{z}} q_n(\vec{s}, \vec{z}; \Theta^{\text{old}}) \left[ -\frac{1}{2} \frac{\partial}{\partial \vec{\mu}} \left( (\vec{z} - \vec{\mu}) \odot \vec{s} \right)^{\text{T}} (\Psi \odot \vec{s} \vec{s}^{\text{T}})^{-1} \left( (\vec{z} - \vec{\mu}) \odot \vec{s} \right) \right] d\vec{z} \\ &= \sum_{n=1}^N \sum_{\vec{s}} \int_{\vec{z}} q_n(\vec{s}, \vec{z}; \Theta^{\text{old}}) \left[ (\Psi \odot \vec{s} \vec{s}^{\text{T}})^{-1} \left( (\vec{z} - \vec{\mu}) \odot \vec{s} \right) \right] d\vec{z} \stackrel{!}{=} 0 \\ &\Rightarrow \vec{\mu} = \frac{\sum_{n=1}^N \langle \vec{s} \odot \vec{z} \rangle_n}{\sum_{n=1}^N \langle \vec{s} \rangle_n}. \end{aligned}$$

## Optimization of the Latent Covariance

Lastly we derive the M-step update for the latent covariance  $\Psi$ :

$$\begin{aligned} & \frac{\partial}{\partial \Psi} \mathcal{F}(\Theta^{\text{old}}, \Theta) \\ &= \sum_{n=1}^N \sum_{\vec{s}} \int_{\vec{z}} q_n(\vec{s}, \vec{z}; \Theta^{\text{old}}) \\ & \quad \left[ -\frac{1}{2} \frac{\partial}{\partial \Psi} (\log |\Psi \odot \vec{s} \vec{s}^{\text{T}}|) - \frac{1}{2} \frac{\partial}{\partial \Psi} \left( (\vec{z} - \vec{\mu}) \odot \vec{s} \right)^{\text{T}} (\Psi \odot \vec{s} \vec{s}^{\text{T}})^{-1} \left( (\vec{z} - \vec{\mu}) \odot \vec{s} \right) \right] d\vec{z} \\ &= \sum_{n=1}^N \sum_{\vec{s}} \int_{\vec{z}} q_n(\vec{s}, \vec{z}; \Theta^{\text{old}}) \\ & \quad \left[ -\frac{1}{2} (\Psi \odot \vec{s} \vec{s}^{\text{T}})^{-1} + \frac{1}{2} (\Psi \odot \vec{s} \vec{s}^{\text{T}})^{-2} \left( (\vec{z} - \vec{\mu}) \odot \vec{s} \right) \left( (\vec{z} - \vec{\mu}) \odot \vec{s} \right)^{\text{T}} \right] d\vec{z} \stackrel{!}{=} 0 \\ &\Rightarrow \Psi = \frac{\sum_{n=1}^N \langle (\vec{z} \vec{z}^{\text{T}} - \vec{\mu} \vec{\mu}^{\text{T}}) \odot \vec{s} \vec{s}^{\text{T}} \rangle_n}{\sum_{n=1}^N \langle \vec{s} \vec{s}^{\text{T}} \rangle_n} \\ & \quad = \frac{\sum_{n=1}^N \langle (\vec{s} \odot \vec{z})(\vec{s} \odot \vec{z})^{\text{T}} \rangle_n - \vec{\mu} \vec{\mu}^{\text{T}} \sum_{n=1}^N \langle \vec{s} \vec{s}^{\text{T}} \rangle_n}{\sum_{n=1}^N \langle \vec{s} \vec{s}^{\text{T}} \rangle_n}. \end{aligned}$$

## B Performance vs. Complexity Trade-Off

If the approximation parameters  $H'$  and  $\gamma$  are held constant, the computational cost of the algorithm scales with the computational cost of the selection function. If the latter cost scales linearly

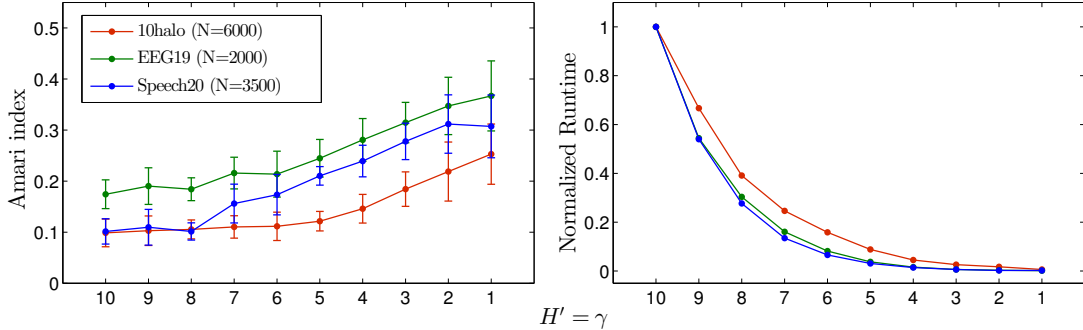


Figure 7: Performance of the GSC on 10halo, EEG19 and Speech20 benchmarks for decreasing truncation parameters  $H'$  and  $\gamma$ . The right panel shows how the computational demand of the truncated variational algorithm decreases with the values of the truncation parameters. The runtime plots are normalized by the runtime value obtained for  $H' = \gamma = 10$  for each of the benchmarks.

with  $H$  (as is the case here), then so does the overall computational complexity [compare complexity considerations by Lücke and Eggert, 2010]). This is consistent with numerical experiments in which we measured the increase in computational demand (see Fig. 8). In experiments with  $H$  increasing from 16 to 1024, we observed a, finally, close to linear increase of computational costs. Note, however, that a larger  $H$  implies a larger number of parameters, and thus may require more data points to prevent overfitting. Although a larger dataset increases computational demand, our truncated approximation algorithm allows us to take advantage of parallel computing architecture in order to more efficiently deal with large datasets (see Appendix C for details). Therefore in practice, we can weaken the extent of an increase in computational cost due to a higher demand for data. Furthermore, we examined the benefit of using GSC (in terms of average speedup over EM iterations) versus the cost regarding algorithmic performance. We compared approximation parameters in the range of  $H' = \gamma = [1, 10]$  and again observed the performance of the algorithm on the task of source separation (with randomly generated orthogonal ground truth mixing bases and no observed noise). Fig.7 shows that a high accuracy can still be achieved for relatively small values of  $H' = \gamma$  which, at the same time, results in strongly reduced computational demands.

## C Dynamic Data Repartitioning for Batch/Parallel Processing

We have seen in section 4 that the truncated variational approach discriminatively selects the most relevant causes of an observation  $\vec{y}$  in order to approximate the posterior distribution over the latent space. In practice we can further benefit from the latents' preselection by making us



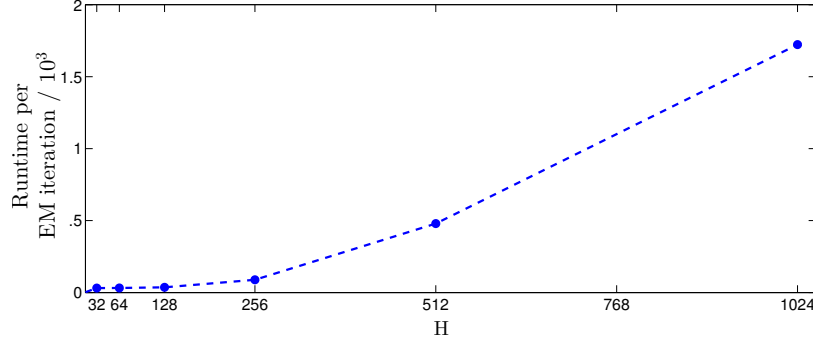


Figure 8: Time scaling behavior of GSC for increasing latent dimensions  $H$  and fixed truncation parameters  $H'$  and  $\gamma$ .

of it for clustering the observed data for batch processing, which in turn allows for an efficient parallel implementation of the truncated EM algorithm. Prior to each E-step, we can partition the data by clustering it based on the most relevant causes chosen by the preselection step for each  $\vec{y}$ . The resulting clusters can then be distributed among multiple compute cores to perform the E-step (Eqns. 25 to 27) in parallel. This approach not only pursues a natural partitioning of data, but in a parallel setting, it can prove to be more efficient than a uniform distribution of data as in [Bornschein et al., 2010]. By maximizing the similarity among data points assigned to an individual processing unit, we can minimize across all the units redundant computations involved in Eqns. 19 and 27, that are tied to specific states of the causes. Particularly, as shown in the main text, the truncated posterior (25) takes the following form:

$$p(\vec{s}, \vec{z} | \vec{y}^{(n)}, \Theta) \approx \frac{\mathcal{N}(\vec{y}^{(n)}; \vec{\mu}_{\vec{s}}, C_{\vec{s}}) \mathcal{B}(\vec{s}; \vec{\pi}) \mathcal{N}(\vec{z}; \vec{\kappa}_{\vec{s}}^{(n)}, \Lambda_{\vec{s}})}{\sum_{\vec{s}' \in \mathcal{K}_n} \mathcal{N}(\vec{y}^{(n)}; \vec{\mu}_{\vec{s}'}, C_{\vec{s}'}) \mathcal{B}(\vec{s}'; \vec{\pi})} \delta(\vec{s} \in \mathcal{K}_n). \quad (33)$$

Here the parameters  $\vec{\mu}_{\vec{s}}, C_{\vec{s}}$  and  $\Lambda_{\vec{s}}$  entirely depend on a state  $\vec{s}$  of causes. Also,  $\vec{\kappa}_{\vec{s}}^{(n)}$  takes prefactors that can be precomputed given  $\vec{s}$ . To compute (33), it turns out that our dynamic data redistribution strategy is more efficient than a static (and uniform) data distribution approach. This is illustrated in Fig. 9, which shows empirical E-step speedup over the static data distribution strategy taken as a baseline. The error bars were generated by performing 15 trials per given data size  $N$ . For all the trials, model scale (i.e., data dimensionality) and truncation approximation parameters were kept constant<sup>9</sup>. Each trial was run in parallel on 24 computing nodes. The red

<sup>9</sup>The observed and the latent dimensions of the GSC model were 25 and 20 respectively. The truncation approximation parameters  $H'$  and  $\gamma$  (maximum number of active causes in a given latent state) were 8 and 5 respectively.

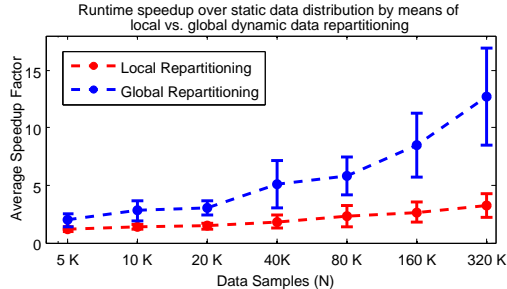


Figure 9: Runtime speedup of the truncated variational E-step (Eqns. 25 to 27) with the static data distribution strategy taken as a baseline. The red plot shows the speedup when initially uniformly distributed data samples were only clustered locally by each processing unit, while the blue plot shows the speedup as a result of globally clustering and redistributing the data. The runtimes include the time taken by clustering and repartitioning modules.

plot in the figure also shows the speedup as a result of an intermediate approach. There we initially uniformly distributed the data samples which were then only locally clustered by each processing unit at every E-step. The blue plot on the other hand shows the speedup as a result of globally clustering and redistributing the data prior to an E-step. It is important to note that all reported results also take into account the cost of data clustering and repartitioning.

We optimize the data clustering process by having each processing unit cluster its own data locally and then merging the resulting clusters globally. To avoid unfair workload distribution, we also bound the maximum cluster size. Currently we pick (per iteration) top  $\alpha$  percentile of cluster sizes as the threshold. Any cluster larger than  $\alpha$  is evenly broken into smaller clusters of maximum  $\alpha$  size<sup>10</sup>. Moreover, to minimize communication overhead among computational units, we only recluster and distribute indices of the data points. This entails that the actual data must reside in a shared memory structure which is efficiently and dynamically accessible by all the computational units. Otherwise, all the units require their own copy of the whole dataset.

It is important to note that although we have illustrated the gains of dynamic data repartitioning using a specific sparse coding model which typically involves state-dependent computationally expensive operations, the technique itself is inherently generic as it is based on a variational EM approach which can be applied to a larger range of models and problems [see Lücke and Eggert, 2010, Sec 4].

<sup>10</sup>The  $\alpha$  for the reported experiments was 5.

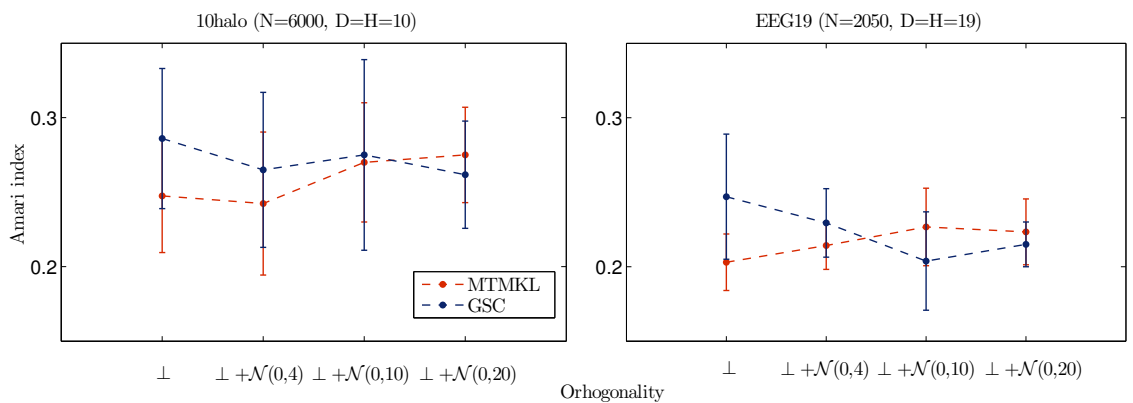


Figure 10: Source separation with observation noise. Performance of GSC vs. MTMKL on 10halo and EEG19 benchmarks with varying degrees of orthogonality of the mixing bases and Gaussian noise added to observations. Performance of GSC vs. MTMKL on the Speech20 benchmark with varying degrees of orthogonality of the mixing bases with Gaussian noise added to observed data. The orthogonality on the x-axis varies from being orthogonal  $\perp$  to increasingly non-orthogonal mixing as randomly generated orthogonal bases are perturbed by adding Gaussian noise  $\mathcal{N}(0, \sigma)$  to them. Performance is compared on the Amari index (32).