On Scalable Inference and Learning in Spike-and-Slab Sparse Coding

vorgelegt von

M.Sc.

Abdul-Saboor Sheikh

geb. in Kuwait-Stadt

von der Fakultät IV – Elektrotechnik und Informatik

der Technischen Universität Berlin

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften

- Dr. rer. nat. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr. Henning Sprekeler
Gutachter: Prof. Dr. Manfred Opper
Gutachter: Prof. Dr. Jörg Lücke
Gutachter: Prof. Dr. Klaus Obermayer

Tag der wissenschaftlichen Aussprache: 14. November 2016

Berlin 2017

Acknowledgements

For this undertaking I extend my gratitude to many more than I can individually remember to mention here. Nevertheless, I would like to begin with thanking Prof. Jöerg Lücke for giving me the opportunity to pursue doctoral research in his group. My discussions with him shaped the problems that now serve as the basis for this thesis and his guidance has greatly contributed to the work presented herein. At the same time he has also provided me the opportunity to work and develop myself independently as a researcher. I would also like to thank Prof. Manfred Opper for agreeing to be my official advisor at the Technical University of Berlin. In addition to Prof. Lücke and Prof. Opper, I am very grateful to Prof. Klaus Obermayer for being on my review committee. I truly appreciate their feedback about my work.

I would also like to acknowledge Bernstein Focus Neurotechnology (BFNT) Frankfurt, German Research Foundation (DFG), the German Ministry of Research and Education (BMBF) and the Cluster of Excellence "Hearing4all" for financially supporting my research. I would like to thank all my group members for our time together. In particular I want to mention Jacquelyn Shelton for her moral support, many discussions and fruitful collaborations. I also had a chance to learn a lot from working together with Jöerg Bornschein. I deeply regard both of my colleagues for their constant support and help both at and outside work. I would further like to name Zhenwen Dai, Georgios Exarchakis, Marc Henniges, Christian Keck and Philip Sterne to express my gratitude towards them.

None of this would have been possible if it had not been for the enormous support of my sincere friend and lovely wife Wajiha. I always found her bearing with me the ebb and flow of my work. While she shared my successes, she also reinvigorated my confidence and my motivation, always helping me overcome my frustrations and move forward. I also want to specially thank my loving parents and my caring brother and sister for their support. Most importantly, I need to mention my little Ayla, whose arrival gave me the final nudge of will to complete this work.

I would like to acknowledge Frankfurt Institute for Advanced Studies (FIAS), Frankfurt Center for Scientific Computing (CSC Frankfurt), Neural Information Processing Group at the Technical University of Berlin and the Machine Learning group at the University of Oldenburg for providing the compute infrastructure and logistical support that allowed me to carry out my work. I furthermore thank anonymous reviewers of the published work presented in this thesis. Last but not the least, I want to thank my dear friend Christian Moewes for his feedback and help with the German translation of the abstract.

Berlin, 24th of June 2016 Abdul-Saboor Sheikh

Abstract

Sparse coding is a widely applied latent variable analysis technique. The standard formulation of sparse coding assumes Laplace as a prior distribution for modeling the activations of latent components. In this work we study sparse coding with spike-and-slab distribution as a prior for latent activity. A spike-and-slab distribution has its probability mass distributed across a 'spike' at zero and a 'slab' spreading over a continuous range. For its capacity to induce exact zeros with a higher likelihood, a spike-and-slab prior distribution constitutes a more accurate model of sparse coding. The distribution as a prior also allows for the sparseness of latent activity to be directly inferred from observed data, which essentially makes spike-and-slab sparse coding more flexible and self-adaptive to a wide range of data distributions. By modeling the slab with a Gaussian distribution, we furthermore show that in contrast to the standard approach to sparse coding, we can indeed derive closed-form analytical expressions for exact inference and learning in linear spike-and-slab sparse coding. However, as the posterior landscape of a spike-and-slab prior turns out to be highly multi-modal with a prohibitive exploration cost, in addition to the exact method, we also develop subspace and Gibbs sampling based approximate inference techniques for scalable applications of the linear model. We contrast our approximation methods with variational approximation for scalable posterior inference in linear spike-and-slab sparse coding. We further combine the Gaussian spike-and-slab prior with a nonlinear generative model, which assumes a point-wise maximum combination rule for the generation of observed data. We analyze the model as a precise encoder of low-level features such as edges and their occlusions in visual data. We again combine subspace selection with Gibbs sampling to overcome the analytical intractability of performing exact inference in the model.

We numerically analyze our methods on both synthetic and real data for their verification and comparison with other approaches. We assess the linear spike-and-slab approach on source separation and image denoising benchmarks. In most experiments we obtain competitive or state-of-the-art results, while we find that spike-and-slab sparse coding overall outperforms other comparable approaches. By extracting thousands of latent components from a large amount of training data we further demonstrate that our subspace Gibbs sampler is among the most scalable posterior inference methods for a linear sparse coding approach. For the nonlinear model we experiment with artificial and real images to demonstrate that the components learned by the model lie closer to the ground-truth and are easily interpretable as the underlying generative causes of the input. We find that in comparison to standard sparse coding, the nonlinear spike-and-slab approach can compressively encode images using naturally sparse and discernible compositions of latent components. We also demonstrate that the components inferred by the model from natural image patches are statistically more consistent with respect to their structure and distribution to the response patterns of simple cells in the primary visual cortex of the brain.

This work thereby contributes novel methods for sophisticated inference and learning in spike-and-slab sparse coding, while it also empirically showcases their functional efficacy through a variety of applications.

Zusammenfassung

Sparse Coding ist eine weit verbreitete Technik der latenten Variablenanalyse. Die Standardformulierung von Sparse Coding setzt a priori eine Laplace-Verteilung zur Modellierung der Aktivierung von latenten Komponenten voraus. In dieser Arbeit untersuchen wir Sparse Coding mit einer a priori Spike-and-Slab-Verteilung für latente Aktivität. Eine Spike-and-Slab-Verteilung verteilt ihre Wahrscheinlichkeitsmasse um ein Aktionspotential ("Spike") um Null und eine dicke Verteilung ("slab") über einen kontinuierlichen Wertebereich. Durch die Induktion von exakten Nullen mit einer höheren Wahrscheinlichkeit erzeugt eine A-priori-Spike-and-Slab-Verteilung ein genaueres Modell von Sparse Coding. Als A-priori-Verteilung erlaubt sie es uns die Seltenheit von latenten Komponenten direkt von Daten abzuleiten, sodass ein Spike-and-Slab-getriebenes Modell von Sparse Coding sich besser verschiedensten Verteilungen von Daten anpasst. Durch das Modellieren des Slab mittels einer Gauß-Verteilung zeigen wir, dass – im Gegensatz zur Standardformulierung von Sparse Coding – wir in der Tat geschlossene analytische Ausdrücke ableiten können, um eine exakte Ableitung und das Lernen eines linearen Spike-and-Slab-Sparse-Coding-Modell durchzuführen. Weil eine Spike-and-Slab-A-priori-Verteilung zu einer hoch multimodalen A-posteriori-Landschaft mit viel zu hohen Suchkosten führt, entwickeln wir zusätzlich zur exakten Methode Näherungslösungen basierend auf einem Teilraum und Gibbs-Sampling für skalierbare Anwendungen des Modells. Wir vergleichen unseren Ansatz der näherungsweisen Inferenz mit näherungsweiser Variationsrechnung des linearen Spike-and-Slab-Sparse Coding. Des Weiteren kombinieren wir die Spike-and-Slab-A-priori-Verteilung mit einem nichtlinearen Sparse-Coding-Modell, das eine punktweise Maximum-Kombinationsregel zur Datengenerierung voraussetzt. Wir analysieren das Modell als genauen Kodierer von untergeordneten Merkmalen in Bildern wie z.B. Kanten und deren Okklusionen. Wir lösen die analytische Ausweglosigkeit, eine Ableitung von multimodalen A-posteriori-Verteilungen im Modell durchzuführen, durch die Kombination von Gibbs-Sampling und der Auswahl eines Teilraums, um eine skalierbare Prozedur für die approximative Inferenz des Modells zu entwickeln.

Wir analysieren unsere Methode numerisch durch synthetische und wirkliche Daten zum Nachweis und Vergleich mit anderen Ansätzen. Wir bewerten den linearen Spike-and-Slab-Ansatz mittels Maßstäben für die Quellentrennung und zur Rauschunterdrückung in Bildern. In den meisten Experimenten erhalten wir vergleichsweise oder die beste Resultate. Gleichzeitig finden wir, dass Spike-and-Slab-Sparse-Coding insgesamt andere vergleichbare Ansätze übertrifft. Durch die Extraktion von Tausenden von latenten Komponenten aus einer riesigen Menge an Trainingsdaten zeigen wir des Weiteren, dass unserer Teilraum Gibbs-Sampler zu den skalierbarsten Inferenzmethoden der linearen Sparse-Coding-Modelle gehört. Für das nichtlineare Modell experimentieren wir mit künstlichen und echten Bildern zur Demonstration, dass die von dem Modell gelernten Komponenten näher an der "Ground Truth" liegen und leichter zu interpretieren sind als die zugrundeliegenden generierenden Einflüsse der Eingabe. Wir finden, dass – im Vergleich zu Standard-Sparse-Coding – der nichtlineare Spike-and-Slab-Ansatz Bilder komprimierend kodieren kann durch natürliche dünnbesetzte und klar erkennbare Kompositionen von latenten Komponenten. Wir zeigen auch, dass die vom Modell abgeleiteten Komponenten von natürlichen Bildern statistisch konsistenter sind in ihrer Struktur und Verteilung mit dem Antwortmuster von einfachen Zellen im primären visuellen Kortex.

Diese Arbeit leistet durch neue Methoden zur komplexen Inferenz und zum Erlernen

von Spike-and-Slab-Sparse-Coding einen Beitrag und demonstriert deren praktikable Wirksamkeit durch einen Vielzahl von Anwendungen.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Latent Variable Models and Sparse Coding

Latent variable analysis is ubiquitously applied in the domain of machine learning. Its application spans across a broad range of tasks such as dimensionality reduction, feature extraction, clustering, speech and image recognition, classification etc. (Olshausen and Field, 1996; Lee and Seung, 1999; Hyvärinen and Oja, 2000; Bishop, 2006; Hinton et al., 2006; Raina et al., 2007; Bengio, 2009, among many others). Latent variable analysis techniques can take numerous formulations, but the main principle behind them is to express their input in terms of basic constituents, which essentially makes it easier to interpret the input for further processing. For human perception, such a component extraction mechanism is quite natural. We find ourselves predisposed towards breaking down our sensory input into various components in order to understand and act in our daily environments. For instance, the brain is very quick at decomposing its visual input into physical objects and shapes, which in turn can be composed of multiple components. Similarly an auditory input can be easily separated by the brain into individual sound sources, where each of the sources e.g., music, speech, etc. can itself be a composition of a number of auditory components.

In its basic form, a latent variable analysis technique can be described as a dual-layer graphical model. One of the two layers is termed as 'observed', since the units in this layer represent individual elements (e.g., pixels of an image) of an observation. The second 'latent' layer units represent the basic components (e.g., real-world objects, sound sources, etc.), which in different combinations are considered to get transformed into various instances of the observed data.

If the latent components are defined to be intrinsic features of the observed data, then latent variable models (LVMs) can be applied to generate feature codes of the data by projecting it back onto the latents (e.g., by assessing the degree to which a latent component is expressed in an observation). Such feature codes can provide more descriptive expressions of the observed data, which can prove to be more insightful for various kinds of analysis

tasks such as data clustering, classification, etc. LVMs can also be stacked together to form multi-layer feature extraction hierarchies, such that the latent layer of one LVM becomes the input for the observed layer of the next model on top. Top layers in such hierarchies can be interpreted as representations of more 'abstract' concepts defining the observations, whereas the bottom layers can be seen as detectors of more primitive features of the data. Such architectures have proven to be very successful in machine learning (e.g., Hinton et al., 2006; Bengio, 2009; Kavukcuoglu et al., 2010b; Bengio et al., 2013).

A large number of latents gives LVMs more expressive power for generating rich (high-dimensional) feature codes, however, for any given observation it can be expected that only a few of the features are found to be its representative constituents. This gives rise to the notion of sparse representation in feature space. In biology the evidence for such a feature detection mechanism was discovered by Hubel and Wiesel (1959). The work demonstrated how the so called simple cells in the primary visual cortex ($V1$) act as edge-detecting filters with sensitivities to different spatial locations, edge widths and orientations on visual stimuli received from retina through the visual pathway in the brain. The work also highlighted the sparse nature of the activation response of simple cells. Selective tuning properties of the cells implied that in a large population, only a very tiny fraction would have their excitation pattern activated by a natural stimulus with confined edge-like structure in it.

The functional behavior of the $V1$ simple cells was later computationally formulated as an LVM with sparsely activated latents (Olshausen and Field, 1996). The sparse LVM a-priori defined its latent variables to be driven by a heavy-tailed distribution (i.e., Cauchy) such that a high probability mass around the null value introduced a bias towards sparse activations of the variables for encoding the observed data into a latent space. The model took a data-driven approach to infer the latent components from observed data such that sparse combinations of the components could reconstruct the data with a minimal disparity. From small patches of natural world images fed as observations, the model extracted spatially localized recurring structures as intrinsic components of the data. For their similarity with the retinal input, the image patches indeed produced latent components which qualitatively resembled the edge-detecting filters discovered earlier in the $V1$ area of the brain. Under the sparsity constraint, the data-driven learning enabled the evolution of latent components as complementary low-level features of data the model was fed as observations. Therefore for having the potential for its application in a variety of machine learning tasks e.g., classification, denoising, source separation, etc., the sparse coding (SC) approach has since become one of the most widely applied and researched latent variable analysis techniques in the field (Elad and Aharon, 2006; Aharon et al., 2006; Raina et al., 2007; Mairal et al., 2008; Yang et al., 2009; Wang et al., 2010; Kavukcuoglu et al., 2010a, to name a few).

Model Formulation

Probabilistic formulation of sparse coding (or LVMs in general) delivers a more accurate and robust approach to data-driven modeling of a real process; a-priori defining the latent activations to be governed by heavy-tailed distributions captures the sparsity property of the latent feature representation, while treating the observed units as random variables allows a model to absorb observation noise or compensate for a modeling mismatch e.g., due to an approximated formulation of the underlying observation generating process. Varying instances of sparse coding can differ according to the distributions assumed over the latent or the observed variables, or based on a defined interaction of the latent components for generating the observed data. For example in standard sparse coding (Olshausen and Field, 1996), the latents are taken to be Cauchy distributed with Gaussian observation noise, while the latent components are assumed to superimpose linearly to generate an observation.

Advancements in the area of sparse coding have focused both on scaling it up to a large number of latent components (see e.g., Lee et al., 2007; Seeger, 2008; Mairal et al., 2010; Ribeiro and Opper, 2011, among numerous others) and on varied probabilistic formulations and nonlinear component interactions (e.g., Dayan and Zemel, 1995; Olshausen and Millman, 2000; Lücke and Sahani, 2008; West, 2003; Yang et al., 2010; Mohamed et al., 2012; Henniges et al., 2014).

Research Direction

This work investigates an arguably more accurate formulation of sparse coding, which results from assuming a specific prior distribution over the latents, namely the 'spike-and-slab' prior. A spike-and-slab distribution comprises a 'spike' of probability mass at the null value, while the rest of its probability mass is distributed over a range of non-zero values (hence the 'slab'). As a result, a spike-and-slab distribution can generate exact zeros with a significantly higher probability as compared to more popular sparse priors such as Cauchy or Laplace. In the context of sparse coding, the capacity to induce true sparsity has therefore recently made spike-and-slab a more prevalent choice for a prior distribution over latents (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, among numerous others).

Optimizing sparse coding models for a set of observed training data in general involves finding a-posteriori latent representations for each observation. This often entails an analytically or computationally intractable exploration of all possible latent activity patterns spanning over the prior domain. In the case of spike-and-slab driven latents, the problem however becomes more pronounced as the structure of the prior develops into a highly multi-

modal probability landscape. Given an observation for linearly interacting components for instance, the a-posteriori spike-and-slab latent space has exponentially many probability modes with respect to the number of latents (Lücke and Sheikh, 2012; Titsias and Lazaro-Gredilla, 2011), where each mode can span subspaces of plausible latent representations of the given observation. This work will focus on the development and analysis of novel optimization procedures for spike-and-slab sparse coding, which intrinsically allow for multi-modal exploration of the latent space probabilities for both linear and nonlinear instances of component interactions.

Before going into further details, we will briefly look at the probabilistic formulation of directed LVMs. Afterwards we will describe a generic and widely applied framework for optimizing the LVMs given the observed data for training, which will then serve as the basis for the methods we will introduce later.

Directed Latent Variable Models

Let us denote an observation by $\vec{y} \in \mathbb{R}^D$ and a latent variable by $\vec{s} \in \mathbb{R}^H$. If we posit that $\vec{y}$ is caused by an interaction between latent components associated with $\vec{s}$, then we can define a probabilistic generative process as follows:

$$\vec{s} \sim p(\vec{\theta}) = \prod_{h=1}^{H} p(\theta_h)\,, \tag{1.1}$$

$$\vec{y}\,|\,\vec{s} \sim p(f(W, \vec{s}), \dots)\,, \tag{1.2}$$

where $W \in \mathbb{R}^{D \times H}$ is defined to be a column-dominated matrix of latent causes or components. Whether or not a component $\vec{W}^{(h)}$ contributes to the generation of an observation $\vec{y}$ depends on the activation of the latent s_h. The latents in (1.1) are assumed to be a-priori independent such that their activities do not affect each other. One can also assume a structured prior over the latents to relax the a-priori independence assumption. However, in comparison to the dual-layer structure (Figure 1.1) of the model (1.1) and (1.2), such a model could employ deep hierarchies of latent layers (see Hinton et al., 2006; Bengio, 2009, for instance). The optimization of deep models poses an even greater challenge of latent space exploration, therefore a general practice is to approximate them as stacked dual-layer LVMs, where each latent layer in a deep hierarchy is greedily optimized by treating the layer and the one preceding it as the latent and the observed layers of a dual-layer LVM (e.g., Hinton et al., 2006; Bengio, 2009; Kavukcuoglu et al., 2010b; Bengio et al., 2013). Then in a final phase the parameters of all the layers are jointly "fine-tuned" to perform a specific task, typically classification. Such applications of deep LVMs have been shown to be quite successful in many machine learning tasks (Bengio et al., 2013). For a detailed discussion

on deep architectures, we refer to the works of Bengio (2009); Bengio et al. (2013).

The stochastic nature of the generative process (1.2) makes the model more robust against uncertainties that are often relevant when dealing with real data; for instance observation noise, model mismatch, etc. The expected value of the generative distribution of $\vec{y}$ is given by the component interaction $f(W, \vec{s})$, while depending on the distribution other parameters may also influence the generative process. For example, a linear model with Gaussian noise would take two parameters: the expected value $f(W, \vec{s}) = W\vec{s}$ and an additional scale parameter σ. Similarly one can define a sparse prior on the latents $h \in H$ to instantiate a sparse coding model. We will now reserve Θ to refer to the entire set of parameters of an LVM.

In our formulation, we take Θ to be a set of scalar parameters, which when estimated from training data qualify as point estimators. This can be different in a fully Bayesian setting, where instead of scalar values, distributions over parameters are defined (and subsequently fitted during training) for model parameterization. While fully Bayesian methods can for example more robustly avoid the local optima occurring due to sensitivity to random initialization or few training data, point estimator based approaches offer more on the front of computational and analytical tractability. With a focus on large-scale learning, we therefore remain with the point-based estimation of parameters in this work; nonetheless, we empirically observe that our proposed methods do not exhibit either any particular sensitivity to initialization or a tendency of converging to local optima.

Figure 1.1 depicts the generative process (1.1) and (1.2) as a directed graphical model. The top-down arrows from the latent to the observed layer illustrate the conditional dependency of an observation on the latents in (1.2). In contrast to the directed generative modeling view taken here, there are also other ways to formulate LVMs. For instance feed-forward artificial neural networks (ANNs) take the form of bottom-up graphical models, where the arrows point from an input to the next hidden layer (Jordan et al., 1999). Another example is a Restricted Boltzmann machine (RBM) (e.g., Jordan et al., 1999; Hinton et al., 2006), which is an undirected graphical model. Similarly, other models can form graphical topologies with intra-layer or recurrent connections, but from a probabilistic perspective, each formulation is a way to establish how the different variables of a model conditionally depend on the others when the model is activated in a specific manner. For instance in a top-down pass through Figure 1.1, the latents s_h act independently in (1.1) to generate $\vec{y}$ conditioned on $\vec{s}$ (1.2), but when the model is presented with $\vec{y}$ with a goal to infer the unobserved $\vec{s}$, the a-posteriori activities of the individual latents become conditionally dependent on $\vec{y}$ and also on each other for jointly representing the sought-after vector $\vec{s}$. This illustrates how given the topology of a model, conditioning one set of variables on the others may either prove to be simple or complicated; for the directed LVM (1.1) and (1.2), the generative distribution

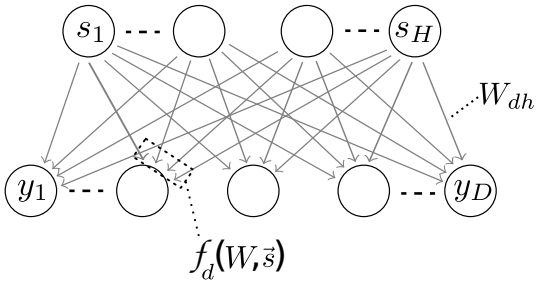

Figure 1.1: Visualization of a directed latent variable model.

$p(\vec{y} \,|\, \vec{s}; \Theta)$ has a straight-forward formulation, but when the process is inverted, $p(\vec{s} \,|\, \vec{y}; \Theta)$ spans over the joint latent domain. Challenges associated with the computation of such conditionals indeed have a crucial role in the formulation of different LVMs and the methods proposed for their training and application (see e.g., Jordan et al., 1999; Wainwright and Jordan, 2008). For our purposes, we will now sketch a generic framework for parameter optimization in directed LVMs, which we will employ later for developing our methods.

Parameter Optimization

The task of learning in an LVM is about optimizing the values of its parameters such that the model can "explain" the observed data it sees for training through its learned latent representations. With respect to generative modeling, this can be perceived as a search for the "right" set of parameter values, which in principle enable a generative model to mimic training data.[1] For a probabilistic model, a natural measure to test its goodness of fit as a generative model for the data at hand is the data likelihood, which is the probability of the data computed under the model given the values of its parameters Θ. Hence the learning task for the LVM (1.1) and (1.2) can be formulated as a search for optimal values for Θ, such that the likelihood is maximized for provided training data. This approach to parameter optimization is termed as *maximum likelihood estimation*.

If we use an instance of the model (1.1) and (1.2) to generate N mutually independent tuples of data $(\mathcal{S}, \mathcal{Y}) = \{\vec{s}^{(n)}, \vec{y}^{(n)}\}_{n=1,\ldots,N}$, then we can compute the joint likelihood given Θ as:

$$L(\Theta; \mathcal{S}, \mathcal{Y}) := p(\mathcal{S}, \mathcal{Y} \,|\, \Theta) = \prod_{n=1}^{N} p(\vec{s}^{(n)} \,|\, \Theta)\, p(\vec{y}^{(n)} \,|\, \vec{s}^{(n)}, \Theta).$$

In practice however when $\mathcal{S}$ is not observed, we must go through each possible realization

[1] provided that the complexity of the model under consideration allows it to be a suitable generative model for the data.

of the latent variables to compute the exact marginalized log-likelihood[2] of $\mathcal{Y}$:

$$\mathcal{L}(\Theta; \mathcal{Y}) := \log p(\mathcal{Y} \,|\, \Theta) = \sum_{n=1}^{N} \log \int p(\vec{s} \,|\, \Theta)\, p(\vec{y}^{\,(n)} \,|\, \vec{s}, \Theta)\, \mathrm{d}\vec{s}. \tag{1.3}$$

Expectation Maximization

To find Θ that maximizes (1.3), starting from a random initialization we can in principle let the parameters evolve iteratively by step-wise updating their values in the direction of the gradient $\mathrm{d}\mathcal{L}/\mathrm{d}\Theta$, however with integration over the latent space, the gradient expressions usually turn out to be rather intricate to follow. Therefore to instead arrive at a simplified optimization problem, we derive a lower-bound of the log-likelihood by introducing a distribution $q(\vec{s} \,|\, \tilde{\Theta})$ such that:

$$\begin{aligned}
\mathcal{L}(\Theta; \mathcal{Y}) &= \sum_{n=1}^{N} \log \int q(\vec{s} \,|\, \tilde{\Theta}) \left[\frac{p(\vec{s} \,|\, \Theta)\, p(\vec{y}^{\,(n)} \,|\, \vec{s}, \Theta)}{q(\vec{s} \,|\, \tilde{\Theta})} \right] \mathrm{d}\vec{s} \\
&\geq \sum_{n=1}^{N} \int q(\vec{s} \,|\, \tilde{\Theta}) \log \left[\frac{p(\vec{y}^{\,(n)}, \vec{s} \,|\, \Theta)}{q(\vec{s} \,|\, \tilde{\Theta})} \right] \mathrm{d}\vec{s} \\
&= \sum_{n=1}^{N} \int \Big[q(\vec{s} \,|\, \tilde{\Theta}) \log p(\vec{y}^{\,(n)}, \vec{s} \,|\, \Theta) - q(\vec{s} \,|\, \tilde{\Theta}) \log q(\vec{s} \,|\, \tilde{\Theta}) \Big] \mathrm{d}\vec{s} \\
&= \sum_{n=1}^{N} \Big[\Big\langle \log p(\vec{y}^{\,(n)}, \vec{s} \,|\, \Theta) \Big\rangle_q + H(q) \Big],
\end{aligned} \tag{1.4}$$

where $\langle \cdot \rangle_q$ denotes the expected value with respect to $q(\vec{s} \,|\, \tilde{\Theta})$ and $H(q)$ is the *Shannon entropy* of q, which is independent of Θ, the parameters of our interest. We obtain the so called *free-energy* (1.4) as a result of applying *Jensen's inequality*, which states that $\mathrm{f}(\langle \cdot \rangle_{\mathrm{q}}) \geq \langle \mathrm{f}(\cdot) \rangle_{\mathrm{q}}$ for a concave function f and a distribution q such that $\int \mathrm{q}(x)\mathrm{d}x = 1, \mathrm{q}(x) \geq 0$.

Optimizing the free-energy with respect to Θ is not as convoluted as the optimization of the marginal log-likelihood for the parameters. For many models, closed-form analytical solutions for computing optimal parameter values can be derived by canonically setting the partial derivatives of the free-energy with respect to individual parameters in Θ to zero. However, since it is a lower bound of the marginal likelihood (1.3), we also need the other distribution q in (1.4) to be optimal to make the bound as tight as possible. Let us therefore

[2]The logarithm is solely applied for mathematical convenience. It otherwise does not change the main objective due to its monotonically increasing nature.

consider:

$$
\begin{aligned}
&\mathcal{L}(\Theta; \mathcal{Y}) - \mathcal{F}(q, \Theta) \\
&\qquad = \log p(\mathcal{Y} \,|\, \Theta) + \sum_{n=1}^{N} \int q(\vec{s} \,|\, \tilde{\Theta}) \log \left[\frac{q(\vec{s} \,|\, \tilde{\Theta})}{p(\vec{y}^{\,(n)}, \vec{s} \,|\, \Theta)} \right] \mathrm{d}\vec{s} \\
&\qquad = \log p(\mathcal{Y} \,|\, \Theta) + \sum_{n=1}^{N} \int q(\vec{s} \,|\, \tilde{\Theta}) \log \left[\frac{q(\vec{s} \,|\, \tilde{\Theta})}{p(\vec{s} \,|\, \vec{y}^{\,(n)}, \Theta)\, p(\vec{y}^{\,(n)} \,|\, \Theta)} \right] \mathrm{d}\vec{s} \\
&\qquad = \log p(\mathcal{Y} \,|\, \Theta) + \sum_{n=1}^{N} \int q(\vec{s} \,|\, \tilde{\Theta}) \log \left[\frac{q(\vec{s} \,|\, \tilde{\Theta})}{p(\vec{s} \,|\, \vec{y}^{\,(n)}, \Theta)} \right] \mathrm{d}\vec{s} - \sum_{n=1}^{N} \log\, p(\vec{y}^{\,(n)} \,|\, \Theta) \\
&\qquad = \sum_{n=1}^{N} \int q(\vec{s} \,|\, \tilde{\Theta}) \log \left[\frac{q(\vec{s} \,|\, \tilde{\Theta})}{p(\vec{s} \,|\, \vec{y}^{\,(n)}, \Theta)} \right] \mathrm{d}\vec{s} \\
&\qquad = \sum_{n=1}^{N} D_{KL}(q(\vec{s} \,|\, \tilde{\Theta}) \,||\, p(\vec{s} \,|\, \vec{y}^{\,(n)}, \Theta)) \geq 0. \qquad (1.5)
\end{aligned}
$$

Here $D_{KL}(\mathrm{p} \,||\, \mathrm{q})$ is the *Kullback-Leibler* (KL) divergence, a measure of discrepancy of a distribution q as compared to the distribution p. The KL-divergence is zero only when $\mathrm{p}(x) = \mathrm{q}(x)$ over the entire domain of x. Hence according to (1.5), to equate (1.4) with (1.3) the distribution $q(\vec{s} \,|\, \tilde{\Theta})$ should be replaced with $p(\vec{s} \,|\, \vec{y}^{\,(n)}, \Theta)$, which is the posterior distribution over the latent space given an observation $\vec{y}^{\,(n)}$ and current Θ. The distribution can be computed by applying *Bayes' rule*:

$$
p(\vec{s} \,|\, \vec{y}^{\,(n)}, \Theta) = \frac{p(\vec{y}^{\,(n)} \,|\, \vec{s}, \Theta)\, p(\vec{s} \,|\, \Theta)}{\int p(\vec{y}^{\,(n)} \,|\, \vec{s}\,', \Theta)\, p(\vec{s}\,' \,|\, \Theta) \mathrm{d}\vec{s}\,'}. \qquad (1.6)
$$

Now we can rewrite the free-energy as:

$$
\mathcal{F}(\Theta^{\mathrm{old}}, \Theta) := \sum_{n=1}^{N} \left[\left\langle \log p(\vec{y}^{\,(n)}, \vec{s} \,|\, \Theta) \right\rangle_n + H(\Theta^{\mathrm{old}}) \right], \qquad (1.7)
$$

where $\langle \cdot \rangle_n$ denotes the expectation with respect to the posterior $p(\vec{s} \,|\, \vec{y}^{\,(n)}, \Theta^{\mathrm{old}})$ and $H(\Theta^{\mathrm{old}})$ is the posterior entropy. For current parameter values denoted by Θ^{old}, the gap between (1.7) and the marginal data likelihood (1.3) can be locally closed by computing the posteriors as in (1.6). Then an updated parameter set Θ can be obtained to maximize the expected data likelihood under the computed posteriors by setting $\mathrm{d}\mathcal{F}(\Theta^{\mathrm{old}}, \Theta)/\mathrm{d}\Theta = 0$. Repeating the two steps for an iterative optimization of parameters is formally known as the *Expectation Maximization* (EM) algorithm. As described by Neal and Hinton (1998), the EM algorithm is a coordinate-ascent procedure, which is guaranteed to never decrease the data likelihood.

In practice, the likelihood is increased to a (possibly local) maximum by alternating between

$$\textbf{E-step:} \quad \textit{compute } p(\vec{s}\,|\,\vec{y}^{\,(n)}, \Theta^{\text{old}})$$
$$\text{and } \textbf{M-step:} \quad \underset{\Theta}{\textit{argmax}}\ \mathcal{F}(\Theta^{\text{old}}, \Theta).$$

The E- and M-steps break down the difficult problem of likelihood optimization into two relatively simpler optimization tasks: for given observations, the E-step infers the expected values of the unobserved latents using current model parameters, whereas provided the observations and their latent expectations, the M-step computes maximum likelihood estimates of model parameters by optimizing a lower-bound of the marginal data likelihood.

Although deriving expressions for M-step parameter updates maybe analytically involved, but the costly bit of the EM algorithm is the E-step, which computes posterior probabilities that often involve an analytically intractable integration over the latent space (1.6). Even if the latent space is assumed to be discrete, the cost of enumerating through each possible state is computationally prohibitive for large latent spaces. A feasible solution therefore is to approximate the exact posterior. For instance for standard sparse coding with Laplace prior, maximum a-posteriori (MAP) based methods have been proposed (for instance Mairal et al., 2010; Lee et al., 2007), which approximate the latent posterior by finding one most likely configuration of the latent variables to explain an observation. Other methods propose to approximate the true posterior with a simpler (uni-modal) distribution (e.g., Seeger, 2008; Ribeiro and Opper, 2011), or factorize the posterior by assuming a-posteriori independence among latents (see Jordan et al., 1999; Wainwright and Jordan, 2008, for variational approximation and inference). There are also sampling based methods, which stochastically explore the latent space to approximate the posterior distribution (see e.g., Shelton et al., 2011; Mohamed et al., 2012). Which of the approximate inference schemes works better or worse depends on a specific instance of the model and also sometimes on its application. In the case of spike-and-slab sparse coding, due to the multi-modal nature of its posterior landscape, a MAP or a uni-modal approximation for instance (Lee et al., 2007; Seeger, 2008; Ribeiro and Opper, 2011) is prone to be more suboptimal, whereas variational (Titsias and Lazaro-Gredilla, 2011; Goodfellow et al., 2012; Sheikh et al., 2014) and sampling based (Mohamed et al., 2012; Shelton et al., 2015) inference methods have been successfully applied to train spike-and-slab sparse coding models.

Chapter Outline

The rest of this thesis is divided as follows:

In Chapter 2 we derive a closed-form solution for running the exact EM algorithm for maximum likelihood learning in linear spike-and-slab sparse coding, where we the

latent slab is considered to be Gaussian. We perform simulations for empirical verification of our learning framework and to gauge the model as an approximation to standard sparse coding (Olshausen and Field, 1996). We also assess model's competitiveness through standard benchmarks, comparing it to other latent variable models on a source separation task. The chapter has been published as Lücke and Sheikh (2012).

Chapter 3 proposes an extension of the model we study in Chapter 2. In this chapter we develop a scalable learning scheme for the spike-and-slab model by applying a truncated variational approach for approximated posterior inference in sparse LVMs (Lücke and Eggert, 2010). We conduct a systematic analysis of the proposed method and contrast it with factored variational inference in spike-and-slab sparse coding (Titsias and Lazaro-Gredilla, 2011; Goodfellow et al., 2012). We further compare our method with a number of established sparse coding models on various tasks and benchmarks. The chapter has appeared as Sheikh et al. (2014).

Chapter 4 applies the hybrid inference framework of Shelton et al. (2011) for large-scale learning in linear spike-and-slab sparse coding. By combining latent preselection (Lücke and Eggert, 2010) with sampling, we formulate an efficient procedure for approximate posterior inference in the model. To assess and demonstrate the scalability of the method, we apply it on a large scale to natural image patches. We then contrast the method with respect to its scalability against other comparable approaches. Through experiments involving ground-truth information, we also look at the convergence properties of the select and sample based inference in comparison to purely sampling driven learning applied to the model. In experiments on benchmark data, we further compare the performance of the proposed method on the task of image denoising. Parts of this chapter have been published as Sheikh and Lücke (2016).

Chapter 5 introduces a nonlinear spike-and-slab sparse coding method as a more accurate model for low-level occlusions in visual data. As the exact posterior inference in the model turns out to be analytically intractable, we present an approximate inference scheme based on Shelton et al. (2011) for scalable learning in the model. Through numerous experiments on both real and synthetic datasets, we do a comparative analysis of latent features/components learned by the nonlinear model with those extracted by standard sparse coding with Laplace prior. Finally we perform a qualitative comparison of features learned by the model on natural image data with *in vivo* neural findings from different physiological studies (i.e., Ringach, 2002; Usrey et al., 2003; Niell and Stryker, 2008). The chapter is published as Shelton et al. (2015) with earlier results appearing in Shelton et al. (2012).

Chapter 2

Gaussian Sparse Coding

In the previous chapter we introduced the EM algorithm (Section 1.5.1), which defines a generic framework for iteratively finding a maximum likelihood parameterization of an LVM given observed training data. We saw that while the M-step performs parameter updates that can be obtained by taking the derivatives of the free-energy with respect to model parameters to zero, it is the E-step which involves computing expected values of the missing or latent data that turns out to be more challenging (see Equation 1.6). Even though there are LVMs such as instances of mixture and factor analysis models for which the E-step can be performed exactly (see Bishop, 2006, for example), for methods like sparse coding with continuous latent priors, e.g., Cauchy or Laplace, the E-step posteriors cannot be computed in closed-form (see Lee et al., 2007; Mairal et al., 2010).

In this chapter we propose exact solutions for both E- and M-steps for linear sparse coding with a continuous prior, where we assume the prior to be a spike-and-slab distribution. We show that by modeling the slab as a standard Gaussian, we can derive closed-form analytical expressions for computing posteriors over the latent variables. In the M-step we infer all the model parameters from training data including the sparsity of individual latents, which in the case of standard sparse coding is done outside the EM learning loop e.g., through manual tuning of an external hyperparameter (Lee et al., 2007; Mairal et al., 2010). The sparsity parameter of the spike-and-slab prior also allows the model to be quite flexible: probabilistic PCA can be recovered as a special case of the model, however the sparsity can also be adjusted/learned to simulate sparse latent distributions (see Figures 2.1 and 2.3).

In numerical simulations we perform method verification via parameter recovery experiments. We show that our model is highly accurate in fitting the data generated by standard sparse coding models (Olshausen and Field, 1996; Lee et al., 2007). Using standard benchmarks on a source separation task, we also demonstrate the competitiveness of our model in a comparison with other latent variable approaches. The main analytical results of this chapter are outlined in more detail in Appendix 2.E.

Closed-form EM for Sparse Coding and its Application to Source Separation

Jörg Lücke* and Abdul-Saboor Sheikh*

FIAS, Goethe-University Frankfurt, 60438 Frankfurt, Germany
{luecke,sheikh}@fias.uni-frankfurt.de
*joint first authorship

Appeared in Proc. LVA/ICA, LNCS pp. 213-221, 2012.

Abstract: We define and discuss the first sparse coding algorithm based on closed-form EM updates and continuous latent variables. The underlying generative model consists of a standard 'spike-and-slab' prior and a Gaussian noise model. Closed-form solutions for E- and M-step equations are derived by generalizing probabilistic PCA. The resulting EM algorithm can take all modes of a potentially multi-modal posterior into account. The computational cost of the algorithm scales exponentially with the number of hidden dimensions. However, with current computational resources, it is still possible to efficiently learn model parameters for medium-scale problems. Thus the model can be applied to the typical range of source separation tasks. In numerical experiments on artificial data we verify likelihood maximization and show that the derived algorithm recovers the sparse directions of standard sparse coding distributions. On source separation benchmarks comprised of realistic data we show that the algorithm is competitive with other recent methods.

Introduction

Probabilistic generative models are a standard approach to model data distributions and to infer instructive information about the data generating process. Methods like principle component analysis, factor analysis, or sparse coding (SC) (e.g., Olshausen and Field, 1996) have all been formulated in the form of probabilistic generative models. Moreover, independent component analysis (ICA), which is a very popular approach to blind source separation, can also be recovered from sparse coding in the limit of zero observation noise (e.g., Dayan and Abbott, 2001).

A standard procedure to optimize parameters in generative models is the application of Expectation Maximization (EM) (e.g., Neal and Hinton, 1998). However, for many generative models the optimization using EM is analytically intractable. For stationary data only the most elementary models such as mixture models and factor analysis (which contains probabilistic PCA as special case) have closed-form solutions for E- and M-step equations. EM for more elaborate models requires approximations. In particular, sparse coding models (Olshausen and Field, 1996; Lee et al., 2007; Seeger, 2008, and many more) require approximations because integrals over the latent variables do not have closed-form solutions.

In this work we study a generative model that combines the Gaussian prior of probabilistic PCA (p-PCA) with a binary prior distribution. Distributions combining binary and continuous parts have been discussed and used as priors before (e.g., Mitchell and Beauchamp, 1988) and are commonly referred to as 'spike-and-slab' distributions. Also sparse coding variants with spike-and-slab distributions have been studied previously (compare West, 2003; Knowles and Ghahramani, 2007;

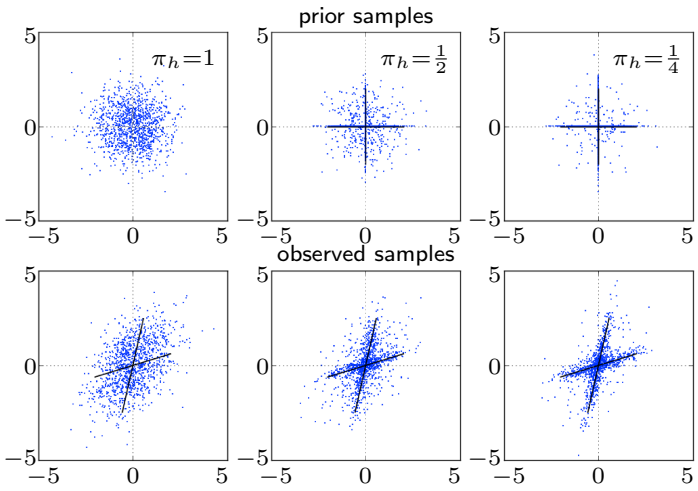

Figure 2.1: Distributions generated by the GSC generative model. The left column shows the distributions generated for $\pi_h = 1$ for all h. In this case the model generates p-PCA distributions. The middle column shows an intermediate value of π_h. The generated distributions are not Gaussians anymore but have a slight star shape. The right column shows distributions for small values of π_h. The generated distributions have a salient star shape similar to standard sparse coding distributions.

Teh et al., 2007; Paisley and Carin, 2009; Knowles and Ghahramani, 2011; Mohamed et al., 2010). However, in this work we show that combining binary and Gaussian latents maintains the p-PCA property of having a closed-form solution for EM optimization. We can, therefore, derive an algorithm that uses exact posteriors with potentially many modes to update model parameters.

The Gaussian Sparse Coding (GSC) model

Let us first consider a pair of H–dimensional i.i.d. latent vectors, a continuous $\vec{z}\in\mathbb{R}^H$ and a binary $\vec{b}\in\{0,1\}^H$ with:

$$p(\vec{b}\,|\Theta) \;=\; \prod_{h=1}^{H} \pi_h^{b_h}\,(1-\pi_h)^{1-b_h} = \text{Bernoulli}(\vec{b};\vec{\pi}) \tag{2.1}$$

$$\text{and}\;\; p(\vec{z}\,|\Theta) \;=\; \mathcal{N}(\vec{z};\,\vec{0},\,I_H), \tag{2.2}$$

where π_h parameterizes the probability of non-zero entries. After generation, both hidden vectors are combined using a pointwise multiplication operator: i.e., $(\vec{b}\odot\vec{z})_h = b_h\,z_h$ for all h. The resulting hidden random variable is a vector of continuous values and zeroes, and it follows a 'spike-and-slab' distribution. Given a hidden vector (which we will denote by $\vec{b}\odot\vec{z}$) , we generate a D–dimensional observation $\vec{y}\in\mathbb{R}^D$ by linearly combining a set of basis functions W and adding Gaussian noise:

$$p(\vec{y}\,|\,\vec{b},\vec{z},\Theta) \;=\; \mathcal{N}(\vec{y};\,W(\vec{b}\odot\vec{z}),\Sigma), \tag{2.3}$$

where $W\in\mathbb{R}^{D\times H}$ is the matrix containing the basis functions $\vec{W}^{(h)}$ as columns, and $\Sigma\in\mathbb{R}^{D\times D}$ is a covariance matrix parameterizing the data noise. The latents'

priors (2.1) and (2.2) together with their pointwise combination and the noise distribution (2.3) define the generative model under consideration. As a special case, the model contains probabilistic PCA (or factor analysis). This can easily be seen by setting all π_h equal to one.

The model (2.1) to (2.3) is capable of generating a broad range of distributions including sparse coding like distributions. This is illustrated in Fig. 2.1 where the parameters π_h allow for continuously changing PCA-like to a SC-like distribution.

While the generative model itself has been studied previously (West, 2003; Teh et al., 2007; Paisley and Carin, 2009; Knowles and Ghahramani, 2011), we will show that a closed-form EM algorithm can be derived, which can be applied to blind source separation tasks. We will refer to the generative model (2.1) to (2.3) as the *Gaussian Sparse Coding* (GSC) model in order to stress that a specific spike-and-slab prior (Gaussian slab) in conjunction with a Gaussian noise model is used. The GSC model is thus an instance of the spike-and-slab sparse coding model (or alternatively known *sparse factor analysis* models; (see e.g., West, 2003; Teh et al., 2007; Paisley and Carin, 2009; Knowles and Ghahramani, 2011).

Expectation Maximization (EM) for Parameter Optimization

Consider a set of N independent data points $\{\vec{y}^{\,(n)}\}_{n=1,\ldots,N}$ with $\vec{y}^{\,(n)} \in \mathbb{R}^D$. For these data we seek parameters $\Theta = (W, \Sigma, \vec{\pi})$ that maximize the data likelihood $\mathcal{L} = \prod_{n=1}^{N} p(\vec{y}^{\,(n)} \,|\, \Theta)$ under the GSC generative model. We employ Expectation Maximization (EM) algorithm for parameter optimization. The EM algorithm (Neal and Hinton, 1998) optimizes the data likelihood w.r.t. the parameters Θ by iteratively maximizing the free-energy given by:

$$\begin{aligned}\mathcal{F}(\Theta^{\text{old}}, \Theta) \;=\; \sum_{n=1}^{N} \sum_{\vec{b}} \int_{\vec{z}} p(\vec{b}, \vec{z} \,|\, \vec{y}^{\,(n)}, \Theta^{\text{old}}) \Big[\log\big(p(\vec{y}^{\,(n)} \,|\, \vec{b}, \vec{z}, \Theta)\big) \\ + \log\big(p(\vec{b} \,|\, \Theta)\big) + \log\big(p(\vec{z} \,|\, \Theta)\big) \Big] \,\mathrm{d}\vec{z} \;+\; H(\Theta^{\text{old}}) \,, \quad (2.4)\end{aligned}$$

where $H(\Theta^{\text{old}})$ is an entropy term only depending on parameter values held fixed during the optimization of $\mathcal{F}$ w.r.t. Θ. Note that integration over the hidden space involves an integral over the continuous part and a sum over the binary part.

Optimizing the free-energy consists of two steps: given the current parameters Θ^{old} the posterior probability is computed in the E-step; and given the posterior, $\mathcal{F}(\Theta^{\text{old}}, \Theta)$ is maximized w.r.t. Θ in the M-step. Iteratively applying E- and M-steps locally maximizes the data likelihood.

M-step parameter updates

Let us first consider the maximization of the free-energy in the M-step before considering expectation values w.r.t. to the posterior in the E-step. Given a generative model, conditions for a maximum free-energy are canonically derived by setting the derivatives of $\mathcal{F}(\Theta^{\text{old}}, \Theta)$ w.r.t. the second argument to zero. For the GSC

model we obtain the following parameter updates:

$$W = \Big(\sum_{n=1}^{N} \vec{y}^{\,(n)} \big\langle \vec{b} \odot \vec{z} \big\rangle_n^{\mathrm{T}}\Big)\Big(\sum_{n=1}^{N} \big\langle (\vec{b} \odot \vec{z})(\vec{b} \odot \vec{z})^{\mathrm{T}} \big\rangle_n\Big)^{-1}, \tag{2.5}$$

$$\begin{aligned} \Sigma &= \frac{1}{N}\sum_{n=1}^{N}\Big[\vec{y}^{\,(n)}(\vec{y}^{\,(n)})^{\mathrm{T}} - 2\big(W\big\langle \vec{b} \odot \vec{z} \big\rangle_n\big)(\vec{y}^{\,(n)})^{\mathrm{T}} \\ &\qquad + W\big(\big\langle (\vec{b} \odot \vec{z})(\vec{b} \odot \vec{z})^{\mathrm{T}} \big\rangle_n W^{\mathrm{T}}\big)\Big] \\ \text{and } \vec{\pi} &= \frac{1}{N}\sum_{n=1}^{N}\big\langle \vec{b} \big\rangle_n, \quad \text{where } \big\langle f(\vec{b},\vec{z}) \big\rangle_n = \sum_{\vec{b}} \int_{\vec{z}} p(\vec{b},\vec{z}\,|\,\vec{y}^{\,(n)},\Theta^{\mathrm{old}})\; f(\vec{b},\vec{z})\,\mathrm{d}\vec{z}. \end{aligned} \tag{2.6}$$

Equations (2.5) to (2.6) define a new set of parameter values $\Theta = (W, \Sigma, \vec{\pi})$ given the current values Θ^{old}. These 'old' parameters are only used to compute the sufficient statistics $\big\langle \vec{b} \big\rangle_n$, $\big\langle \vec{b} \odot \vec{z} \big\rangle_n$ and $\big\langle (\vec{b} \odot \vec{z})(\vec{b} \odot \vec{z})^{\mathrm{T}} \big\rangle_n$ of the model.

Expectation Values

Although the derivation of M-step equations can be analytically intricate, it is the E-step that, for most generative models, poses the major challenge. Source of the problems involved are analytically intractable integrals required for posterior distributions and for expectation values w.r.t. the posterior. The true posterior is therefore often replaced by an approximate distribution (see, e.g., Bishop, 2006; Seeger, 2008) or in the form of factored variational distributions (Jordan et al., 1999; Jaakkola, 2000). The most frequently used approximation is the maximum-a-posterior (MAP) estimate (see, e.g., Olshausen and Field, 1996; Lee et al., 2007) which replaces the true posterior by a delta-function around the posterior's maximum value. Alternatively, analytically intractable expectation values are often approximated using sampling approaches. Using approximations always implies, however, that many analytical properties of exact EM are not maintained. Approximate EM iterations may, for instance, decrease the likelihood or may not recover (local or global) likelihood optima in many cases. There are nevertheless, a limited number of models with exact EM solutions; e.g., mixture models such as the mixture-of-Gaussians, p-PCA or factor analysis etc. Our novel work here extends the set of known models with exact EM solutions. By following along the same lines as for the p-PCA derivations, we maintain in our E-step the analytical tractability of computing expectation values w.r.t. the posterior of the GSC model (2.6).

Posterior Probability

First observe that the discrete latent variable $\vec{b}$ of the GSC model can be directly combined with the basis functions, i.e., $W(\vec{b} \odot \vec{z}) = \tilde{W}_{\vec{b}}\,\vec{z}$, where $(\tilde{W}_{\vec{b}})_{dh} = W_{dh} b_h$. Now we apply the Bayes' rule to write down the posterior:

$$p(\vec{b},\vec{z}\,|\,\vec{y}^{\,(n)},\Theta) = \frac{\mathcal{N}(\vec{y}^{\,(n)};\,\tilde{W}_{\vec{b}}\,\vec{z},\Sigma)\,\mathcal{N}(\vec{z};\,\vec{0},I_H)\,p(\vec{b}\,|\,\Theta)}{\sum_{\vec{b}'}\int \mathcal{N}(\vec{y}^{\,(n)};\,\tilde{W}_{\vec{b}'}\,\vec{z}',\Sigma)\,\mathcal{N}(\vec{z}';\,\vec{0},I_H)\,p(\vec{b}'\,|\,\Theta)\,\mathrm{d}\vec{z}'}. \tag{2.7}$$

Note that given a state $\vec{b}$ in (2.7), the Gaussian governing the observations $\vec{y}^{\,(n)}$ is only dependent on the Gaussian over the continuous latent $\vec{z}$, which is analytically

independent of $\vec{b}$. We can exploit this joint relation to refactorize the Gaussians. Using Gaussian identities the posterior can be rewritten as:

$$\begin{aligned} p(\vec{b}, \vec{z} \,|\, \vec{y}^{\,(n)}, \Theta) &= \frac{\mathcal{N}(\vec{y}^{\,(n)}; \vec{0}, C_{\vec{b}})\, p(\vec{b} \,|\, \Theta)\, \mathcal{N}(\vec{z};\, \vec{\kappa}^{\,(n)}_{\vec{b}}, \Lambda_{\vec{b}})}{\sum_{\vec{b}'} \mathcal{N}(\vec{y}^{\,(n)}; \vec{0}, C_{\vec{b}'})\, p(\vec{b}' \,|\, \Theta) \int \mathcal{N}(\vec{z}';\, \vec{\kappa}^{\,(n)}_{\vec{b}'}, \Lambda_{\vec{b}'})\, \mathrm{d}\vec{z}'} \\ &= p(\vec{b} \,|\, \vec{y}^{\,(n)}, \Theta)\; \mathcal{N}(\vec{z};\, \vec{\kappa}^{\,(n)}_{\vec{b}}, \Lambda_{\vec{b}}), \qquad (2.8) \\ \text{where } C_{\vec{b}} &= \tilde{W}_{\vec{b}} \tilde{W}^{\mathrm{T}}_{\vec{b}} + \Sigma, \qquad \Lambda_{\vec{b}} = \left(\tilde{W}^{T}_{\vec{b}}\, \Sigma^{-1}\, \tilde{W}_{\vec{b}} + I_H\right)^{-1} \\ \text{and } \vec{\kappa}^{\,(n)}_{\vec{b}} &= \Lambda_{\vec{b}}\, \tilde{W}^{\mathrm{T}}_{\vec{b}}\, \Sigma^{-1}\, \vec{y}^{\,(n)}. \qquad (2.9) \end{aligned}$$

Equations (2.8) to (2.9) represent the crucial result for the computation of the E-step below because, first, they show that the posterior does not involve analytically intractable integrals and, second, for fixed $\vec{b}$ 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 parameter updates.

E-step Equations

Derived from (2.8), the expectation values are computed as:

$$\begin{aligned} \left\langle \vec{b} \right\rangle_n &= \sum_{\vec{b}} p(\vec{b} \,|\, \vec{y}^{\,(n)}, \Theta)\; \vec{b}, \qquad (2.10) \\ \left\langle \vec{b} \odot \vec{z} \right\rangle_n &= \sum_{\vec{b}} p(\vec{b} \,|\, \vec{y}^{\,(n)}, \Theta)\; \vec{\kappa}^{\,(n)}_{\vec{b}} \\ \text{and} \quad \left\langle (\vec{b} \odot \vec{z})(\vec{b} \odot \vec{z})^{\mathrm{T}} \right\rangle_n &= \sum_{\vec{b}} p(\vec{b} \,|\, \vec{y}^{\,(n)}, \Theta) \left(\Lambda_{\vec{b}} + \vec{\kappa}^{\,(n)}_{\vec{b}} (\vec{\kappa}^{\,(n)}_{\vec{b}})^{\mathrm{T}}\right). \qquad (2.11) \end{aligned}$$

Note that we have to use the current values $\Theta = \Theta^{\mathrm{old}}$ for all parameters on the right-hand-side. The E-step equations (2.10) to (2.11) represent a closed-form solution for expectation values required for the closed-form M-step (2.5) to (2.6).

Relation to the Mixture of Gaussians

The special form of the posterior in (2.8) allows the derivation of a closed-form experession of the data likelihood: i.e., $p(\vec{y} \,|\, \Theta) = \sum_{\vec{b}}\, \mathrm{Bernoulli}(\vec{b}; \vec{\pi})\; \mathcal{N}(\vec{y}^{\,(n)}; \vec{0}, C_{\vec{b}})$. Note that in principle, this form can be reproduced by a Gaussian mixture model. However, such a model would consist of 2^H mixture components, with strongly dependent mixing proportions and covariance matrices $C_{\vec{b}}$. Closed-form EM-updates can in general not be derived for such dependencies. The standard updates for mixtures of Gaussians require independently parameterized mixing proportions and components. Therefore, the closed-form EM-solutions for the GSC model is not a consequence of closed-form EM for classical Gaussian mixtures.

Numerical Experiments

GSC parameter optimization is non-convex, However, as for all algorithms based on closed-form EM, the GSC algorithm always increases the data likelihood at least to a local maxima. We first numerically investigate how frequently local optima are obtained. Later we assess model's performace on more practical tasks.

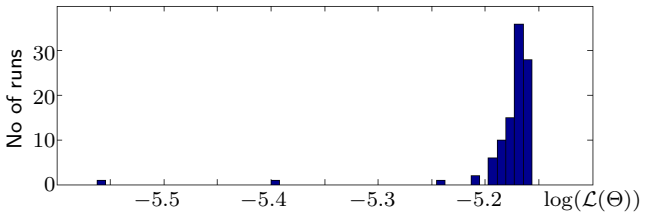

Figure 2.2: Histogram of likelihood values for 100 runs of the GSC algorithm on data generated by a SC model with Cauchy prior. Almost all runs converged to high likelihood values.

Model verification

First, we verify on artificial data that the algorithm increases the likelihood and that it can recover the parameters of the generating distribution. For this, we generated $N = 500$ data points $\vec{y}^{(n)}$ from the GSC generative model (2.1) to (2.3) with $D = H = 2$. We used randomly initialized generative parameters [1]. The algorithm was run 250 times on the generated data. For each run we performed 300 EM iterations. For each run, we randomly and uniformly initialized π_h between 0.05 and 10, set Σ to the covariance accross the data points, and the elements of W we chose to be independently drawn from a normal distribution with zero mean and unit variance. In all runs the generating parameter values were recovered with high accuracy. Runs with different generating parameters produced essentially the same results.

Recovery of sparse directions

To test the model's robustness w.r.t. a relaxation of the GSC assumptions, we applied the GSC algorithm to data generated by standard sparse coding models. We used a standard Cauchy prior and a Gaussian noise model (Olshausen and Field, 1996) for data generation. Fig. 2.3 second panel shows data generated by this sparse coding model while the first panel shows the prior density along one of its hidden dimensions. We generated $N = 500$ data points with $H = D = 2$. We then applied the GSC algorithm with the same parameter initialization as in the previous experiment. We performed 100 trials using 300 EM iterations per trial. Again, the algorithm converged to high likelihood values in most runs (see Fig. 2.2). As a performace measure for this experiment we investigated how well the heavy tails (i.e., the sparse directions) of standard SC were recovered. As a performance metric, we used the Amari index (Amari et al., 1995):

$$A(W) \quad = \quad \frac{1}{2H(H-1)} \sum_{h,h'=1}^{H} \left(\frac{|O_{hh'}|}{\max_{h''} |O_{hh''}|} + \frac{|O_{hh'}|}{\max_{h''} |O_{h''h'}|} \right) - \frac{1}{H-1} \quad (2.12)$$

where $O_{hh'} := \left(W^{-1} W^{\text{gen}}\right)_{hh'}$. The mean Amari index of all runs with high likelihood values was below 10^{-2}, which shows a very accurate recovery of the sparse directions. Fig. 2.3 (right panel) visualizes the distribution recovered by the GSC algorithm in a typical run. The dotted red lines show the density contours of

[1] We obtained W^{gen} by independently drawing each matrix entry from a normal distribution with zero mean and standard deviation 3. π_h^{gen} values were drawn from a uniform distribution between 0.05 and 1, $\Sigma = \sigma^{\text{gen}} I_D$ (where σ^{gen} was uniformly drawn between 0.05 and 10).

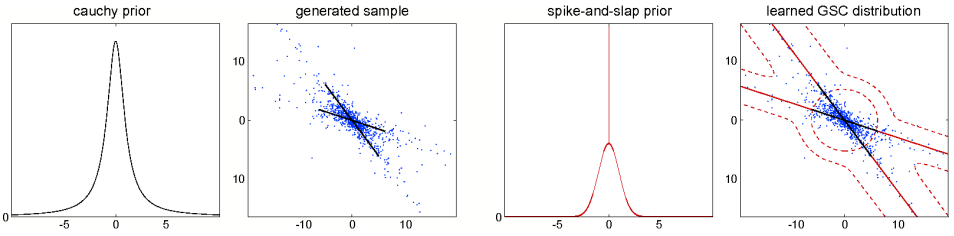

Figure 2.3: Comparison of standard sparse coding and GSC. **Left panels**: Cauchy distribution (along one hidden dimension) as a standard SC prior (Olshausen and Field, 1996) and data generated by it. **Right panels**: Spike-and-slab distribution (one of the hidden dimensions) inferred by the GSC algorithm along with inferred sparse directions (solid red lines) and posterior data density contours (dotted red lines).

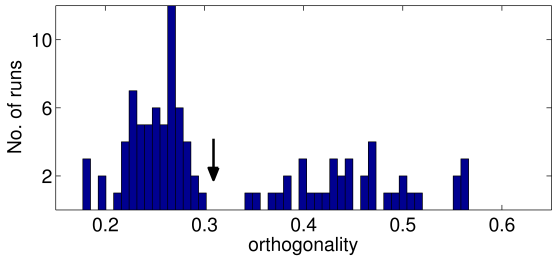

Figure 2.4: Histogram of the deviation from orthogonality of the W matrix for 100 runs of the GSC algorithm on the `Speech4` benchmark ($N = 500$). A clear cluster of the most orthogonal runs can automatically be detected: the threshold of runs considered is defined to be the minimum after the cluster (black arrow).

the learned distribution $p(\vec{y}\,|\,\Theta)$. High accuracy in the recovery of the generating sparse directions (solid black lines) can be observed by comparison with the recovered directions (solid red lines). The results of experiments are qualitatively the same if we increase the number of hidden and observed dimensions; e.g., for $H = D = 4$ we found the algorithm converged to a high likelihood in 91 (with average Amari index below 10^{-2}) of 100 runs.

Other than standard SC with Cauchy prior, we also ran the algorithm on data generated by SC with Laplace prior (Olshausen and Field, 1996; Lee et al., 2007). There for $H = D = 2$, we converged to high likelihood values in 99 of 100 runs with an average Amari index 0.06. In the experiment with $H = D = 4$ the algorithm converged to a high likelihood in 97 of 100 runs. The average Amari index of all runs with high likelihoods was 0.07 in this case.

Source separation

We applied the GSC algorithm to publicly available benchmarks. We used the non-artificial benchmarks of Suzuki and Sugiyama (2011). The datasets mainly contain acoustic data obtained from ICALAB (Cichocki et al., 2007). We generated the observed data by mixing the benchmark sources using randomly generated orthogonal mixing matrix (we followed Suzuki and Sugiyama, 2011). Again, the Amari index (2.12) was used as a performace measure.

For all the benchmarks we used $N = 200$ and $N = 500$ data points (as selected

datasets		Amari index (standard deviation)					
name	N	GSC	GSC$^{\perp}$	NG-LICA	KICA	FICA	JADE
10halo	200	0.34(0.05)	**0.29(0.03)**	**0.29(0.02)**	0.38(0.03)	0.33(0.07)	0.36(0.00)
	500	0.27(0.01)	0.27(0.01)	**0.22(0.02)**	0.37(0.03)	**0.22(0.03)**	0.28(0.00)
Sergio7	200	0.23(0.06)	0.20(0.06)	**0.04(0.01)**	0.38(0.04)	0.05(0.02)	0.07(0.00)
	500	0.18(0.05)	0.17(0.03)	0.05(0.02)	0.37(0.03)	**0.04(0.01)**	**0.04(0.00)**
Speech4	200	0.25(0.05)	**0.17(0.04)**	0.18(0.03)	0.29(0.05)	0.20(0.03)	0.22(0.00)
	500	0.11(0.04)	**0.05(0.01)**	0.07(0.00)	0.10(0.04)	0.10(0.04)	0.06(0.00)
c5signals	200	0.39(0.03)	0.44(0.05)	0.12(0.01)	0.25(0.15)	**0.10(0.02)**	0.12(0.00)
	500	0.41(0.05)	0.44(0.04)	0.06(0.04)	0.07(0.06)	**0.04(0.02)**	0.07(0.00)

Table 2.1: Performance of different algorithms on benchmarks for source separation. Data for NG-LICA, KICA, FICA, and JADE are taken from Suzuki and Sugiyama (2011). Performances are compared based on the Amari index (2.12). Bold values highlight the best performing algorithm(s).

by Suzuki and Sugiyama, 2011). We applied GSC to the data using the same initialization as described before. For each experiment we performed 100 trials with a random new parameter initialization per trial. The first column of Tab. 1 list average Amari indices obtained including all trials per experiment[2]. It is important to note that all the other algorithms listed in the comparison assume orthogonal mixing matrices, while the GSC algorithm does not. Therefore in the column 'GSC$^{\perp}$' in Tab. 1, we report statistics that are only computed over the runs which infered the most orthogonal W matrices. As a measure of orthogonality we used the maximal deviation from 90^o between any two axes. Fig. 2.4 shows as an example a histogram of the maximal deviations of all trials on the `Speech4` data with $N = 500$. As can be observed, we obtained a clear cluster of runs with high orthogonality. We observed worst performace of the GSC algorithm on the `c5signals` dataset. However, the dataset contains sub-Gaussian sources which in general can not be recovered by sparse coding approaches.

Discussion

The GSC algorithm falls into the class of standard SC algorithms. However, instead of a heavy-tail prior, it uses a spike-and-slab distribution. The algorithm has a distinguishing capability of taking the whole (potentially a multimodal) posterior into account for parameter optimization, which is in contrast to the MAP approximation of the posterior, which is a widely used approach for training SC models (see e.g., Lee et al., 2007; Mairal et al., 2009a) . Various sophisticated methods have been proposed to efficently find the MAP (e.g., Tibshirani, 1996). MAP based optimizations usually also require regularization parameters that have to be inferred (e.g., by means of cross-validation). Other approximations that take more aspects of the posterior into account are also being investigated actively (e.g., Seeger, 2008; Mohamed et al., 2010). However, approximations can introduce learning biases. For instance, MAP and Laplace approximations assume monomodality in posterior estimation, which is not always the case.

Closed-form EM learning of the GSC algorithm also comes with a cost, which is exponential w.r.t. the number of hidden dimensions H. This can be seen by considering (2.9) where the partition function requires a summation over all binary vectors $\vec{b}$ (similar for expectation values w.r.t. the posterior). Nevertheless,

[2]We obtained the reported results by diagonalizing the updated Σ in the M-step by setting $\Sigma = \sigma^2 I_D$, where $\sigma^2 = \mathrm{Tr}(\Sigma)/D$.

we show in numerical experiments that the algorithm is well applicable to the typical range of source separation tasks. In such domains the GSC algorithm can benefit from taking potentially a multimodal posterior into acount and infering a whole set of model parameters including the sparsity per latent dimension. For instance, when using a number of hidden dimensions larger than the number of actual sources, the model can discard dimensions by setting π_h parameters to zero. The studied model could thus be considered as treating parameter inference in a more Bayesian way than, e.g., SC with MAP estimate (compare Lee et al., 2007). The second line of research aims at a fully Bayesian description of sparse coding and emphasises a large flexibility using estimations of the number of hidden dimensions and by being applicable with a range of different noise models. The great challenge of these general models is the procedure of parameter estimation. For the model in Mohamed et al. (2010), for instance, the Bayesian methodology involves conjugate priors and hyperparameters in combination with Laplace approximation and different sampling schemes. While the aim, e.g., in West (2003); Knowles and Ghahramani (2007); Teh et al. (2007); Paisley and Carin (2009); Knowles and Ghahramani (2011); Mohamed et al. (2010) is a large flexibility, we aim at a generalization of sparse coding that maintains the closed-form EM solutions.

To summarize, we have studied a novel sparse coding algorithm and have shown its competitiveness on source separation benchmarks. Along with the reported results on source separation, the main contribution of this work is the derivation and numerical investigation of the (to the knowledge of the authors) first closed-form, exact EM algorithm for spkie-and-slab sparse coding.

Acknowledgement: J. Lücke is funded by the German Research Foundation (DFG), grant LU 1196/4-1; A.-S. Sheikh is funded by the German BMBF, grant 01GQ0840.

Appendix

Derivation of Closed-Form Posterior

Here we derive the closed-form solution we use in the E-step for computing the exact poterior (2.8) over Gaussian spike-and-slab latents. To keep the derivation valid for the generalized generative model (3.1) to (3.3) considered in Chapter 3, in the following we will assume that the Gaussian slab is parameterized by a mean vector $\vec{\mu} \in \mathbb{R}^H$ and a full rank covariance matrix $\Psi \in \mathbb{R}^{H \times H}$. Therefore the results (2.8) to (2.9) can be obtained by assuming $\vec{\mu} = 0$ and $\Psi = I_H$.

First let us write down the joint density of two normally distributed random variables $\vec{x} \sim \mathcal{N}(\langle \vec{x} \rangle, \Sigma_x)$ and $\vec{y} \sim \mathcal{N}(\langle \vec{y} \rangle, \Sigma_y)$:

$$\mathrm{p}(\begin{bmatrix} \vec{x} \\ \vec{y} \end{bmatrix}) = \mathcal{N}(\begin{bmatrix} \vec{x} \\ \vec{y} \end{bmatrix}; \begin{bmatrix} \langle \vec{x} \rangle \\ \langle \vec{y} \rangle \end{bmatrix}, \begin{bmatrix} \Sigma_x & \Sigma_{x,y} \\ \Sigma_{x,y}^T & \Sigma_y \end{bmatrix}),$$

where $\Sigma_{x,y}$ is a non-symmetric cross-covariance matrix. Now it turns out that the conditional distributions of the two variables are also Gaussians:

$$\vec{x} \mid \vec{y} \sim \mathcal{N}(\langle \vec{x} \rangle + \Sigma_{x,y} \Sigma_y^{-1} (\vec{y} - \langle \vec{y} \rangle), \Sigma_x - \Sigma_{x,y} \Sigma_y^{-1} \Sigma_{x,y}^T) \tag{2.13}$$

$$\vec{y} \mid \vec{x} \sim \mathcal{N}(\langle \vec{y} \rangle + \Sigma_{x,y}^T \Sigma_x^{-1} (\vec{x} - \langle \vec{x} \rangle), \Sigma_y - \Sigma_{x,y}^T \Sigma_x^{-1} \Sigma_{x,y}) \,. \tag{2.14}$$

In the GSC model (2.1) to (2.3), we have an observed vector $\vec{y} \in \mathbb{R}^D$ that depends on two latent variables, a Gaussian distributed vector $\vec{z} \in \mathbb{R}^H$ and a Bernoulli governed binary vector $\vec{b} \in \{0, 1\}^H$. It follows from (2.3) that:

$$\begin{aligned} \vec{y} &= \tilde{W}_{\vec{b}} \vec{z} + \epsilon \\ \Rightarrow \langle \vec{y} \rangle &= \langle \tilde{W}_{\vec{b}} \vec{z} + \epsilon \rangle \\ &= \tilde{W}_{\vec{b}} \langle \vec{z} \rangle + 0 \\ &= \tilde{W}_{\vec{b}} \vec{\mu} \end{aligned} \tag{2.15}$$

The covariance of $\vec{y}$ given $\vec{b}$ and $\vec{z}$ is:

$$\begin{aligned} \Sigma_{\vec{y}} &= \langle (\vec{y} - \langle \vec{y} \rangle)(\vec{y} - \langle \vec{y} \rangle)^T \rangle \\ &= \langle (\tilde{W}_{\vec{b}} \vec{z} + \epsilon - \tilde{W}_{\vec{b}} \vec{\mu})(\tilde{W}_{\vec{b}} \vec{z} + \epsilon - \tilde{W}_{\vec{b}} \vec{\mu})^T \rangle \\ &= \langle (\tilde{W}_{\vec{b}} (\vec{z} - \vec{\mu}) + \epsilon)(\tilde{W}_{\vec{b}} (\vec{z} - \vec{\mu}) + \epsilon)^T \rangle \\ &= \langle \tilde{W}_{\vec{b}} (\vec{z} - \vec{\mu})(\vec{z} - \vec{\mu})^T \tilde{W}_{\vec{b}}^T \rangle + \langle \epsilon \epsilon^T \rangle \\ &= \tilde{W}_{\vec{b}} \langle (\vec{z} - \vec{\mu})(\vec{z} - \vec{\mu})^T \rangle \tilde{W}_{\vec{b}}^T + \Sigma \\ &= \tilde{W}_{\vec{b}} \Psi \tilde{W}_{\vec{b}}^T + \Sigma \\ &= C_{\vec{b}} \end{aligned} \tag{2.16}$$

Furthermore, the cross-covariance between $\vec{z}$ and $\vec{y}$ given $\vec{b}$ is:

$$\begin{aligned}
\Sigma_{\vec{z},\vec{y}} &= \langle(\vec{z} - \langle\vec{z}\rangle)(\vec{y} - \langle\vec{y}\rangle)^T\rangle \\
&= \langle(\vec{z} - \vec{\mu})(\tilde{W}_{\vec{b}}\vec{z} + \epsilon - \tilde{W}_{\vec{b}}\vec{\mu})^T\rangle \\
&= \langle(\vec{z} - \vec{\mu})(\tilde{W}_{\vec{b}}(\vec{z} - \vec{\mu}) + \epsilon)^T\rangle \\
&= \langle(\tilde{W}_{\vec{b}}(\vec{z} - \vec{\mu})(\vec{z} - \vec{\mu}))^T\rangle + \langle((\vec{z} - \vec{\mu})\epsilon)^T\rangle \\
&= \tilde{W}_{\vec{b}}^T\langle(\vec{z} - \vec{\mu})(\vec{z} - \vec{\mu})^T\rangle + 0 \\
&= \Psi\tilde{W}_{\vec{b}}^T
\end{aligned} \tag{2.17}$$

We know from (2.13) that the conditional distribution of $\vec{z}$ given $\vec{y}$ and $\vec{b}$ is:

$$\begin{aligned}
&p(\vec{z}\,|\,\vec{b},\vec{y}) \\
&= \mathcal{N}(\langle\vec{b}\odot\vec{z}\rangle + \Sigma_{\vec{z},\vec{y}}\Sigma_{\vec{y}}^{-1}(\vec{y} - \langle\vec{y}\rangle), \Psi\big(diag(\vec{b})\big) - \Sigma_{\vec{z},\vec{y}}\Sigma_{\vec{y}}^{-1}\Sigma_{\vec{z},\vec{y}}^T) \\
&= \mathcal{N}(\langle\vec{b}\odot\vec{z}\rangle + \Sigma_{\vec{z},\vec{y}}\Sigma_{\vec{y}}^{-1}(\vec{y} - \langle\vec{y}\rangle), \Psi_{\vec{b}} - \Sigma_{\vec{z},\vec{y}}\Sigma_{\vec{y}}^{-1}\Sigma_{\vec{z},\vec{y}}^T)
\end{aligned}$$

where,

$$\begin{aligned}
&\langle\vec{z}\odot\vec{b}\rangle + \Sigma_{\vec{z},\vec{y}}\Sigma_{\vec{y}}^{-1}(\vec{y} - \langle\vec{y}\rangle) \\
&= (\langle\vec{z}\rangle\odot\vec{b}) + \Psi\tilde{W}_{\vec{b}}^T(\tilde{W}_{\vec{b}}\Psi\tilde{W}_{\vec{b}}^T + \langle\epsilon\epsilon^T\rangle)^{-1}(\vec{y} - \tilde{W}_{\vec{b}}\vec{\mu}) \\
&= (\vec{\mu}\odot\vec{s}) + \big(\underbrace{\tilde{W}_{\vec{b}}^T\Sigma^{-1}\tilde{W}_{\vec{b}} + \Psi^{-1}}_{M_{\vec{b}}}\big)^{-1}\tilde{W}_{\vec{b}}^T\Sigma^{-1}(\vec{y} - \tilde{W}_{\vec{b}}\vec{\mu}) \\
&= \vec{\kappa}_{\vec{b}}
\end{aligned} \tag{2.18}$$

and

$$\begin{aligned}
&\Psi_{\vec{b}} - \Sigma_{\vec{z},\vec{y}}\Sigma_{\vec{y}}^{-1}\Sigma_{\vec{z},\vec{y}}^T \\
&= \Psi_{\vec{b}} - \Psi\tilde{W}_{\vec{b}}^T\big(\tilde{W}_{\vec{b}}\Psi\tilde{W}_{\vec{b}}^T + \langle\epsilon\epsilon^T\rangle\big)^{-1}\tilde{W}_{\vec{b}}\Psi^T \\
&= \Psi_{\vec{b}} - \Psi_{\vec{b}}W^T\big(W\Psi_{\vec{b}}W^T + \langle\epsilon\epsilon^T\rangle\big)^{-1}W\Psi_{\vec{b}}^T \\
&= \Psi_{\vec{b}} - \big(W^T\Sigma^{-1}W + \Psi_{\vec{b}}^{-1}\big)^{-1}W^T\Sigma^{-1}W\Psi_{\vec{b}}^T \\
&= \big(\underbrace{W^T\Sigma^{-1}W + \Psi_{\vec{b}}^{-1}}_{M_{\vec{b}}}\big)^{-1} \\
&= \Lambda_{\vec{b}}
\end{aligned} \tag{2.19}$$

We use the results from Equations (2.15) to (2.19) to derive the full posterior $p(\vec{b},\vec{z}\,|\,\vec{y}^{(n)},\Theta)$ in closed form (2.8).

Chapter 3

Truncated Variational Inference in Spike-and-Slab Sparse Coding

In the last chapter we presented a closed-form analytical solution for the EM algorithm for spike-and-slab sparse coding. We saw that for the special case of a Gaussian slab, we could integrate out the continuous part of the discrete-continuous latent prior in order to exactly compute the E-step posteriors. The computation however still involved a sweep through the discrete part of the posterior latent space, which was comprised of exponentially many states with respect to the number of latents. This incurred a prohibitive computational cost for large-scale applications of the model.

In this chapter we will introduce a novel procedure for scalable inference and learning in linear spike-and-slab sparse coding. The approximate inference method is based on Lücke and Eggert (2010), which for a given observation subdivides the latent space into subspaces of relevant and irrelevant latents – hence by truncating the majority set of irrelevant latents, the posterior is efficiently estimated within a selected subspace of reduced dimensionality. The computational cost of the resulting algorithm no longer depends on the total number of latents, which allows for large-scale applications of the model.

We contrast our method with variational inference in spike-and-slab sparse coding (Zhou et al., 2009; Titsias and Lazaro-Gredilla, 2011; Goodfellow et al., 2013), where the posterior is fully factorized across the latents, i.e., their activations are taken to be a-posteriori independent of each other. The aim of our analysis is to identity any learning biases that maybe introduced by the independence assumption for posterior approximation. In a comparison on source separation benchmarks, we find that spike-and-slab sparse coding in general outperforms the established MAP based methods for standard sparse coding (Mairal et al., 2009a; Lee et al., 2007). The results align with the findings of Mohamed et al. (2012), which also support spike-and-slab models over the L_1-norm based approaches for sparse coding. In further experiments we demonstrate that our method can achieve state-of-the-art performance in image denoising on natural benchmark data.

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

Abdul-Saboor Sheikh[1], Jacquelyn A. Shelton[1] and Jörg Lücke[2]

[1]Faculty of Electrical Engineering and Computer Science
Technical University Berlin
Marchstr. 23, 10587 Berlin, Germany
{sheikh,shelton}@tu-berlin.de

[2]Cluster of Excellence Hearing4all and Faculty VI
University of Oldenburg
26115 Oldenburg, Germany
joerg.luecke@uni-oldenburg.de

Editor: Aapo Hyvärinen

Published in the Journal of Machine Learning Research, 15:2653–2687, 2014.

Abstract: We study inference and learning based on a sparse coding model with 'spike-and-slab' prior. As in standard sparse coding, the model used 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 EM approach and, for comparison, an approach based on standard factored variational distributions. The truncated approach can be regarded as a variational approach with truncated posteriors as 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 EM 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 EM approach improves on the factored variational approach. While the performance of the factored approach saturates with increasing numbers 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

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., 2013; 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). 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, making them desirable for many types of data. Algorithms based on spike-and-slab distributions have successfully been used, e.g., for deep learning and transfer learning (Goodfellow et al., 2013), regression (West, 2003; Carvalho et al., 2008; Carbonetto and Stephen, 2011; Titsias and Lazaro-Gredilla, 2011), or 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., 2013).

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). Figure 3.1 shows typical posterior distributions for spike-and-slab sparse coding (the model will be formally defined in the next section). The figure illustrates posterior examples 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., 2009a; 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., 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., 2013). While MCMC approaches are more general and more accurate given sufficient computational resources, variational approaches are usually more efficient. Especially in high dimensional hidden 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., 2013). The variational approaches applied to

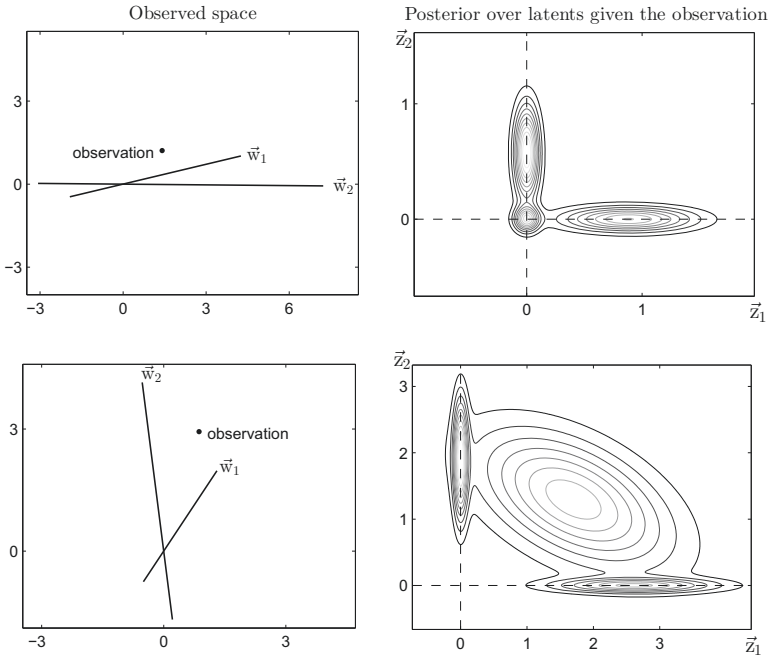

Figure 3.1: Left figures visualize observations generated by two different instantiations of the spike-and-slab sparse coding model (3.1) to (3.3). Solid lines are the generating bases vectors. Right figures illustrate the corresponding exact posteriors over latents computed using (3.13) and (3.15) given observations and generating model parameters. 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 (3.15).

spike-and-slab models thus far (see Rattray et al., 2009; Yoshida and West, 2010; Titsias and Lazaro-Gredilla, 2011; Goodfellow et al., 2013) assume a factorization of the posteriors over the latent dimensions, that is the hidden dimensions are assumed to be independent a-posteriori. This means that any dependencies such as explaining-away effects including correlations (compare Figure 3.1) are ignored and not accounted for. But what consequences does such a negligence 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, 2005; 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, 2005, for a detailed discussion). Therefore, the posterior independence assumption in general may result in suboptimal solutions.

In this work we study an approximate EM approach for spike-and-slab sparse coding which does not assume a-posteriori independence and which can model multiple modes. The novel approach can be considered as a variational EM approach but instead of using factored distributions or Gaussians, it is based on posterior distributions truncated to regions of high probability mass (Lücke and Eggert, 2010). Such truncated EM approaches have recently been applied to different models (see e.g., Puertas et al., 2010; Shelton et al., 2011; Dai and Lücke, 2012;

Bornschein et al., 2013). In contrast to the previously studied factored variational approaches (Titsias and Lazaro-Gredilla, 2011; Mohamed et al., 2012; Goodfellow et al., 2013), 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 approach exclusively to the binary part of the hidden space, we can still fully benefit from the analytical tractability of the continuous integrals.

In this study, 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., 2013), 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 3.II 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.III we derive the closed-form EM parameter update equations for the introduced spike-and-slab model. Based on these equations, in Section 3.IV we derive the truncated EM algorithm for efficient learning in high dimensions. In Section 3.V, we numerically evaluate the algorithm and compare it to factored variational and other approaches. Finally, in Section 3.VI we discuss the results. The Appendix present details of the derivations and experiments.

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 $\vec{b} \in \{0,1\}^H$ and a multivariate Gaussian prior for the continuous latent vector

$\vec{z} \in \mathbb{R}^H$:

$$p(\vec{b}\,|\Theta) = \mathcal{B}(\vec{b};\vec{\pi}) = \prod_{h=1}^{H} \pi_h^{b_h} \,(1-\pi_h)^{1-b_h}, \tag{3.1}$$

$$p(\vec{z}\,|\Theta) = \mathcal{N}(\vec{z};\,\vec{\mu},\Psi), \tag{3.2}$$

where π_h defines the probability of b_h being equal to one and where $\vec{\mu}$ and Ψ 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 (3.2) generalize the spike-and-slab model used in (Lücke and Sheikh, 2012). A point-wise multiplication of the two latent vectors, i.e., $(\vec{b}\odot\vec{z})_h = b_h\, z_h$ generates a 'spike-and-slab' distributed variable $(\vec{b}\odot\vec{z})$, which has either continuous values or exact zero entries. 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 by adding Gaussian noise:

$$p(\vec{y}\,|\,\vec{b},\vec{z},\Theta) = \mathcal{N}(\vec{y};\,W(\vec{b}\odot\vec{z}),\Sigma), \tag{3.3}$$

where each column of the matrix $W \in \mathbb{R}^{D\times H}$ is a basis function $W = (\vec{w}_1,\ldots,\vec{w}_H)$ and where the matrix $\Sigma \in \mathbb{R}^{D\times D}$ parameterizes the observation noise. Full rank covariances Σ can flexibly parametrize noise and have been found beneficial in noisy environments (Dalen and Gales, 2008; Ranzato and Hinton, 2010; Dalen and Gales, 2011). Nevertheless the model can also be constrained to have homoscedastic noise (i.e., $\Sigma = \sigma^2 I$). We use $\Theta = (W,\Sigma,\vec{\pi},\vec{\mu},\Psi)$ to denote all the model parameters. Having a spike-and-slab prior implies that for high levels of sparsity (low values of π_h) the latents assume exact zeros with a 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 (3.1) to (3.3). The factored variational approach (Multi–Task and Multiple Kernel Learning, MTMKL; Titsias and Lazaro-Gredilla, 2011) that we use for detailed comparison is based on a similar 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, $\Sigma = \mathrm{diag}(\sigma_1^2,\ldots,\sigma_D^2)$. On the other hand, it is a generalization by drawing the basis functions W from Gaussian processes. The model (3.1) to (3.3) 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 $p(\vec{b},\vec{z}\,|\,\vec{y})$ is approximated by a variational distribution $q_n(\vec{b},\vec{z};\,\Theta)$ which assumes the combined latents to be independent a-posteriori (compare Zhou et al., 2009; Titsias and Lazaro-Gredilla, 2011; Goodfellow et al., 2013):

$$q_n(\vec{b},\vec{z};\Theta) = \prod_{h=1}^{H} q_n^{(h)}(b_h,z_h;\Theta),$$

where $q_n^{(h)}$ are distributions only depending on b_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 later numerical experiments on the model, we used the source code provided along with that publication.[1] Further comparisons will include the spike-and-slab sparse coding model by Zhou et al. (2009). The generative model is similar to the spike-and-slab model in Equations (3.1) to (3.3) but uses a Beta process prior to parameterize the Bernoulli (the "spike") distribution and assumes homoscedastic observation noise. Inference in their model is based on factored variational EM or Gibbs sampling. As this model is closely related to ours, we use it as another instance for comparison in our numerical experiments in order to assess the influence of different inference method choices. This comparison allows us to explore differences of training the model with a sampling-based approach, as they yield many of the same benefits of our inference method (e.g., flexible representation of uncertainty), but where generally more computational resources are necessary.

Expectation Maximization for Parameter Optimization

In order to learn the model parameters Θ 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 log-likelihood given by:

$$\mathcal{F}(\Theta^{\text{old}}, \Theta) = \sum_{n=1}^{N} \left\langle \log p(\vec{y}^{(n)}, \vec{b}, \vec{z} \,|\, \Theta) \right\rangle_n + H(\Theta^{\text{old}}), \tag{3.4}$$

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

$$\left\langle f(\vec{b}, \vec{z}) \right\rangle_n = \sum_{\vec{b}} \int_{\vec{z}} p(\vec{b}, \vec{z} \,|\, \vec{y}^{(n)}, \Theta^{\text{old}})\, f(\vec{b}, \vec{z})\, \mathrm{d}\vec{z} \tag{3.5}$$

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

The EM algorithm iteratively optimizes the free-energy by alternating between two steps. First, in the E-step given the current parameters Θ^{old}, the relevant expectation values under the posterior $p(\vec{b}, \vec{z} \,|\, \vec{y}^{(n)}, \Theta^{\text{old}})$ are computed. Next, the M-step uses these posterior expectations and maximizes the free-energy $\mathcal{F}(\Theta^{\text{old}}, \Theta)$ w.r.t. Θ. 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 (3.5). The required expressions and approximations for these expectations (the E-step) will be derived afterwards.

[1] We downloaded the code from `http://www.well.ox.ac.uk/~mtitsias/code/varSparseCode.tar.gz`.

M-step Parameter Updates

The M-step parameter updates of the model are canonically obtained by setting the derivatives of the free-energy (4.3) w.r.t. the second argument to zero. Details of the derivations are given in Appendix 3.G and the resulting update equations are as follows:

$$W = \frac{\sum_{n=1}^{N} \vec{y}^{(n)} \big\langle \vec{b} \odot \vec{z} \big\rangle_n^{\mathrm{T}}}{\sum_{n=1}^{N} \big\langle (\vec{b} \odot \vec{z})(\vec{b} \odot \vec{z})^{\mathrm{T}} \big\rangle_n}, \tag{3.6}$$

$$\vec{\pi} = \frac{1}{N} \sum_{n=1}^{N} \big\langle \vec{b} \big\rangle_n, \tag{3.7}$$

$$\vec{\mu} = \frac{\sum_{n=1}^{N} \big\langle \vec{b} \odot \vec{z} \big\rangle_n}{\sum_{n=1}^{N} \big\langle \vec{b} \big\rangle_n}, \tag{3.8}$$

$$\Psi = \sum_{n=1}^{N} \Big[\big\langle (\vec{b} \odot \vec{z})(\vec{b} \odot \vec{z})^{\mathrm{T}} \big\rangle_n - \big\langle \vec{b}\,\vec{b}^{\mathrm{T}} \big\rangle_n \odot \vec{\mu}\vec{\mu}^{\mathrm{T}} \Big] \odot \Big(\sum_{n=1}^{N} \Big[\big\langle \vec{b}\,\vec{b}^{\mathrm{T}} \big\rangle_n \Big] \Big)^{-1} \tag{3.9}$$

$$\text{and} \quad \Sigma = \frac{1}{N} \sum_{n=1}^{N} \Big[\vec{y}^{(n)} (\vec{y}^{(n)})^{\mathrm{T}} - W \big[\big\langle (\vec{b} \odot \vec{z}) \big\rangle_n \big\langle (\vec{b} \odot \vec{z}) \big\rangle_n^{\mathrm{T}} \big] W^{\mathrm{T}} \Big]. \tag{3.10}$$

E-step Expectation Values

The M-step equations (3.6) to (3.10) 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/sums over the latent space. For the derivation of closed-form E-step equations it is useful to know that the discrete latent variable $\vec{b}$ can be combined with the basis function matrix W so that we can rewrite (3.3) as

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

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

Here the data likelihood $p(\vec{y} \,|\, \Theta)$ can be derived in closed-form after marginalizing the joint $p(\vec{y}, \vec{b}, \vec{z} \,|\, \Theta)$ over $\vec{z}$:

$$\begin{aligned} p(\vec{y}, \vec{b} | \, \Theta) &= \mathcal{B}(\vec{b}; \vec{\pi}) \int \mathcal{N}(\vec{y}; \tilde{W}_{\vec{b}}\, \vec{z}, \Sigma)\, \mathcal{N}(\vec{z}; \vec{\mu}, \Psi)\, \mathrm{d}\vec{z} \\ &= \mathcal{B}(\vec{b}; \vec{\pi})\; \mathcal{N}(\vec{y}; \tilde{W}_{\vec{b}}\, \vec{\mu}, C_{\vec{b}}), \end{aligned} \tag{3.11}$$

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

$$p(\vec{y} \,|\, \Theta) = \sum_{\vec{b}} \mathcal{B}(\vec{b}; \vec{\pi})\; \mathcal{N}(\vec{y}; \tilde{W}_{\vec{b}}\, \vec{\mu}, C_{\vec{b}}). \tag{3.12}$$

Thus, the marginal distribution takes the form of a Gaussian mixture model with 2^H mixture components indexed by $\vec{b}$. However, unlike in a standard Gaussian mixture model, 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 Equations (3.11) and (3.12) the posterior over the binary latents $p(\vec{b}\,|\,\vec{y},\Theta)$ is given by:

$$p(\vec{b}\,|\,\vec{y},\Theta) = \frac{p(\vec{b},\vec{y}\,|\,\Theta)}{p(\vec{y}\,|\,\Theta)} = \frac{\mathcal{B}(\vec{b};\vec{\pi})\;\mathcal{N}(\vec{y};\tilde{W}_{\vec{b}}\,\vec{\mu},C_{\vec{b}})}{\sum_{\vec{b}'}\mathcal{B}(\vec{b}';\vec{\pi})\;\mathcal{N}(\vec{y};\;\tilde{W}_{\vec{b}'}\vec{\mu},C_{\vec{b}'})}. \tag{3.13}$$

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

$$p(\vec{b},\vec{z}\,|\,\vec{y},\Theta) = p(\vec{b}\,|\,\vec{y},\Theta)\,p(\vec{z}\,|\,\vec{b},\vec{y},\Theta). \tag{3.14}$$

Like the first factor in (3.14), the second factor is also analytically tractable and given by:

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

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

$$\begin{aligned} \Lambda_{\vec{b}} &= (\tilde{W}_{\vec{b}}^{\mathrm{T}}\,\Sigma^{-1}\,\tilde{W}_{\vec{b}} \;+\; \Psi_{\vec{b}}^{-1})^{-1}, \\ \vec{\kappa}_{\vec{b}}^{\,(n)} &= (\vec{b}\odot\vec{\mu}) + \Lambda_{\vec{b}}\,\tilde{W}_{\vec{b}}^{\mathrm{T}}\,\Sigma^{-1}\,(\vec{y}^{\,(n)} - \tilde{W}_{\vec{b}}\,\vec{\mu}) \end{aligned} \tag{3.15}$$

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

$$p(\vec{b},\vec{z}\,|\,\vec{y}^{\,(n)},\Theta) \;\;=\;\; \frac{\mathcal{B}(\vec{b};\vec{\pi})\,\mathcal{N}(\vec{y}^{\,(n)};\tilde{W}_{\vec{b}}\,\vec{\mu},C_{\vec{b}})\,\mathcal{N}(\vec{z};\;\vec{\kappa}_{\vec{b}}^{\,(n)},\Lambda_{\vec{b}})}{\sum_{\vec{b}'}\mathcal{B}(\vec{b}';\vec{\pi})\,\mathcal{N}(\vec{y}^{\,(n)};\tilde{W}_{\vec{b}'}\,\vec{\mu},C_{\vec{b}'})}. \tag{3.16}$$

Equation (3.16) 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 $\vec{b}$ 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 expectation values required for the M-step are given as follows:

$$\big\langle\vec{b}\big\rangle_n \;\;=\;\; \sum_{\vec{b}} q_n(\vec{b};\Theta)\,\vec{b}, \tag{3.17}$$

$$\big\langle\vec{b}\,\vec{b}^{\mathrm{T}}\big\rangle_n \;\;=\;\; \sum_{\vec{b}} q_n(\vec{b};\Theta)\,\vec{b}\,\vec{b}^{\mathrm{T}}, \tag{3.18}$$

$$\big\langle\vec{b}\odot\vec{z}\big\rangle_n \;\;=\;\; \sum_{\vec{b}} q_n(\vec{b};\Theta)\,\vec{\kappa}_{\vec{b}}^{\,(n)}, \tag{3.19}$$

$$\text{and}\quad \big\langle(\vec{b}\odot\vec{z})(\vec{b}\odot\vec{z})^{\mathrm{T}}\big\rangle_n \;\;=\;\; \sum_{\vec{b}} q_n(\vec{b};\Theta)\,\big(\Lambda_{\vec{b}} + \vec{\kappa}_{\vec{b}}^{\,(n)}(\vec{\kappa}_{\vec{b}}^{\,(n)})^{\mathrm{T}}\big). \tag{3.20}$$

In all of the expressions above, the left-hand-sides are expectation values over the full latent space w.r.t. the posterior $p(\vec{b},\vec{z}\,|\,\vec{y}^{\,(n)},\Theta)$, whereas the right-hand-sides now take the form of expectation values only over the binary part w.r.t. the posterior $p(\vec{b}\,|\,\vec{y}^{\,(n)},\Theta)$ in Equation (3.13). The derivations of E-step equations

(3.17) to (3.20) are a generalization of the derivations by Lücke and Sheikh (2012). While Gaussian identities and marginalization have been used to obtain analytical results for mixture-of-Gaussians priors before (e.g. Moulines et al., 1997; Attias, 1999; Olshausen and Millman, 2000; Garrigues and Olshausen, 2007), 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 pointed out by Mohamed et al. (2012).

Iteratively computing the E-step equations (3.17) to (3.20) using the current parameters Θ and the M-step equations (3.6) to (3.10), represents a closed-form and exact EM algorithm which increases the data likelihood of the model to (possibly local) maxima.

Truncated 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 $p(\vec{b}\,|\,\vec{y},\Theta)$ above, which each require a summation over all binary vectors $\vec{b} \in \{0,1\}^H$. For tasks involving low dimensional hidden spaces, the exact algorithm is still applicable. For higher dimensional problems approximations are required, however. 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 $p(\vec{b},\vec{z}\,|\,\vec{y},\Theta)$, we use a truncated approximation to the posterior $p(\vec{b}\,|\,\vec{y}^{\,(n)},\Theta)$ in Equation (3.13). The truncated approximation is defined to be proportional to the true posteriors on subspaces of the latent space with high probability mass (compare *Expectation Truncation*, Lücke and Eggert, 2010). More concretely, a posterior distribution $p(\vec{b}\,|\,\vec{y}^{\,(n)},\Theta)$ is approximated by a distribution $q_n(\vec{b};\Theta)$ that only has support on a subset $\mathcal{K}_n \subseteq \{0,1\}^H$ of the state space:

$$q_n(\vec{b};\Theta) = \frac{p(\vec{b},\vec{y}^{\,(n)}\,|\,\Theta)}{\sum\limits_{\vec{b}\,'\in\mathcal{K}_n} p(\vec{b}\,',\vec{y}^{\,(n)}\,|\,\Theta)}\,\delta(\vec{b}\in\mathcal{K}_n), \tag{3.21}$$

where $\delta(\vec{b}\in\mathcal{K}_n)$ is an indicator function, i.e., $\delta(\vec{b}\in\mathcal{K}_n)=1$ if $\vec{b}\in\mathcal{K}_n$ and zero otherwise.

The basic assumption behind the approximation in (3.21) is that the posterior over the entire hidden space is concentrated in small volumes, which is represented by the reduced support of subset $\mathcal{K}_n$. When using a spike-and-slab sparse coding model to gain a generative understanding of the data, sparsity in the posterior distribution usually emerges naturally. We can see an illustration of this in Figure 3.2 (generation details in Section 3.v.4). Figure 3.2A shows (for three typical data points) how much posterior mass is carried by each of the $H = 10$ latent dimensions. Figure 3.2B shows (for the same data points) histograms of the posterior mass marginalized across the whole range of hyperplanes spanned by the 10–dimensional latent space. Figure 3.2A indicates that only a subset of the H latents is significantly relevant for encoding the posterior, while Figure 3.2B allows us

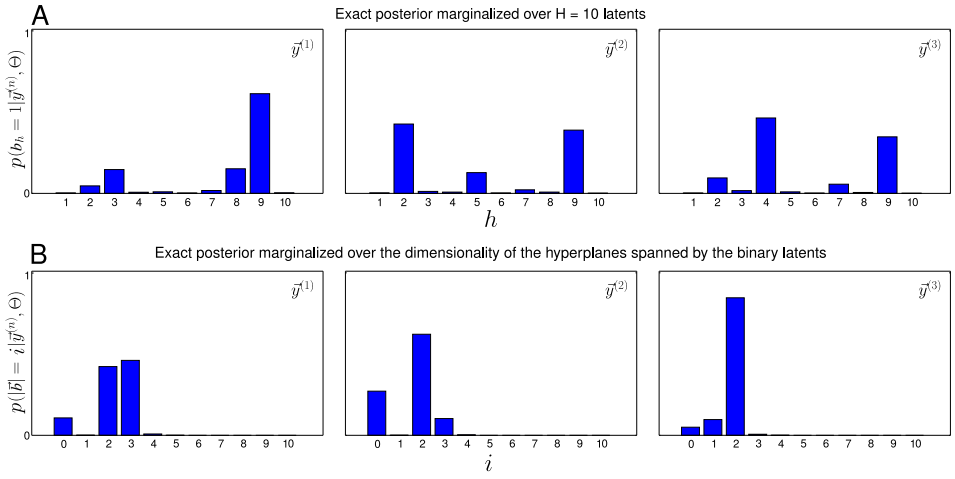

Figure 3.2: Visualization of the exact posterior probabilities of the spike-and-slab model with $H = 10$ latents, computed for three given data points $\vec{y}^{(n)}$. The model was trained on natural data (see Section 3.v.4 for more details). **A** Histograms of the posterior mass over the H latents: $p(b_h = 1 \,|\, \vec{y}^{(n)}, \Theta) = \sum_{\vec{b}_{\text{with}\, b_h=1}} p(\vec{b} \,|\, \vec{y}^{(n)}, \Theta) / \sum_{\vec{b}} p(\vec{b} \,|\, \vec{y}^{(n)}, \Theta)$. Low values for most h imply that these latents can be neglected (i.e., clamped to zero) for a posterior approximation. **B** Histograms of the posterior mass over the hyperplanes of increasing dimensionality i: $p(|\vec{b}| = i \,|\, \vec{y}^{(n)}, \Theta) = \sum_{\vec{b}_{\text{where}}|\vec{b}|=i} p(\vec{b} \,|\, \vec{y}^{(n)}, \Theta) / \sum_{i'=0}^{H} \sum_{\vec{b}_{\text{where}}|\vec{b}|=i'} p(\vec{b} \,|\, \vec{y}^{(n)}, \Theta)$. In case of all the three examples presented here, subspaces with $i > 4$ can be neglected as another approximation step for posterior estimation.

to observe that the posterior mass is primarily contained within low-dimensional hyperplanes of the H–dimensional hidden space. In other words, given a data point we find that most of the posterior mass is concentrated in low-dimensional subspaces spanned by $H' \ll H$ of the latent dimensions. The sparse nature of the posterior as illustrated by Figure 3.2 allows us to apply approximation (3.21), where we define the subsets $\mathcal{K}_n$ based on index sets $I_n \subseteq \{1, \ldots, H\}$, which contain the indices of H' most relevant sparse latents (compare Figure 3.2A) for the corresponding data points $\vec{y}^{(n)}$:

$$\mathcal{K}_n \;=\; \{\vec{b} \;|\; \textstyle\sum_h b_h \leq \gamma \text{ and } \forall h \notin I_n : b_h = 0\} \cup \mathcal{U}, \tag{3.22}$$

where the indices comprising I_n have the highest value of a selection (or scoring) function $\mathcal{S}_h(\vec{y}^{(n)}, \Theta)$ (which we define later). The set $\mathcal{U}$ is defined as $\mathcal{U} = \{\vec{b} \;|\; \sum_h b_h = 1\}$ 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 γ non-zero entries and with non-zero entries only permitted for $h \in I_n$. The parameter $\gamma \leqslant H'$ sets the maximal dimensionality of the considered hyper-planes (compare Figure 3.2B). It was empirically 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.

If we now use the concrete expressions of the sparse spike-and-slab model (Equations (3.1) to (3.3)) for the variational distribution in Equation (3.21), the

truncated approximation is given by:

$$p(\vec{b}\,|\,\vec{y}^{\,(n)},\Theta) \;\approx\; q_n(\vec{b};\Theta) \;=\; \frac{\mathcal{N}(\vec{y}^{\,(n)};\tilde{W}_{\vec{b}}\,\vec{\mu},C_{\vec{b}})\,\mathcal{B}(\vec{b};\vec{\pi})}{\sum_{\vec{b}'\in\mathcal{K}_n}\mathcal{N}(\vec{y}^{\,(n)};\tilde{W}_{\vec{b}'}\vec{\mu},C_{\vec{b}'})\,\mathcal{B}(\vec{b}';\vec{\pi})}\,\delta(\vec{b}\in\mathcal{K}_n) \tag{3.23}$$

The approximation can now be used to compute the expectation values which are required for the M-step equations. If we use the variational distributions in Equation (3.23) for $q_n(\vec{b};\Theta)$ on the right-hand-sides of Equations (3.17) to (3.20), we obtain:

$$\sum_{\vec{b}} q_n(\vec{b};\Theta)\,f(\vec{b}) \;=\; \frac{\sum_{\vec{b}\in\mathcal{K}_n}\mathcal{N}(\vec{y}^{\,(n)};\tilde{W}_{\vec{b}}\,\vec{\mu},C_{\vec{b}})\,\mathcal{B}(\vec{b};\vec{\pi})\,f(\vec{b})}{\sum_{\vec{b}'\in\mathcal{K}_n}\mathcal{N}(\vec{y}^{\,(n)};\tilde{W}_{\vec{b}'}\vec{\mu},C_{\vec{b}'})\,\mathcal{B}(\vec{b}';\vec{\pi})}, \tag{3.24}$$

where $f(\vec{b})$ denotes any of the (possibly parameter dependent) functions of (3.17) to (3.20). 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 $\mathcal{K}_n$.

Note that the definition of $q_n(\vec{b};\Theta)$ in Equation (3.21) neither assumes unimodality like MAP approximations (Mairal et al., 2009a; 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., 2013; Titsias and Lazaro-Gredilla, 2011). The approximation scheme we have introduced here exploits the inherent property of the sparse spike-and-slab model to have posterior probabilities concentrated in low-dimensional subspaces. The quality of our approximated posterior $q_n(\vec{b};\Theta)$ primarily depends on an appropriate selection of the relevant subspaces $\mathcal{K}_n$ (see Section 3.iv.2 below).

The truncated approximation is similar to factored variational approximations or MAP approximations in the sense that it can be formulated as an approximate distribution $q_n(\vec{b};\Theta)$ within the free-energy formulation by Neal and Hinton (1998). Within this formulation, $q_n(\vec{b};\Theta)$ is often referred to as *variational* approximation, and we therefore refer to our approximation as *truncated variational EM*. Like factored variational approaches, we here aim to minimize the KL-divergence between the true posterior and the approximation in Equation (3.21). However, we do not use variational parameters and a gradient based optimization of such parameters for the minimization. Our approach is therefore not a variational approach in the sense of classical variational calculus.

Computational Complexity

The truncated E-step defined by (3.17) to (3.20) with (3.24) scales with the approximation parameters γ and H' which can be defined independently of the latent dimensionality H. The complexity scales as $\mathcal{O}\big(N\sum_{\gamma'=0}^{\gamma}\binom{H'}{\gamma'}(D+\gamma')^3\big)$, where the D^3 term can be dropped from the cubic expansion if the observed noise Σ is considered to be diagonal or homoscedastic. Also the truncated approximation yields sparse matrices in Equations (3.21) and (3.23) 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 trun-

cated approach to achieve significantly improved runtime performances. Clustering the data points using the index sets I_n saves us from redundantly performing various computationally expensive operations involved in Equations (3.15) and (3.23), that given a state $\vec{b} \in \mathcal{K}_n$ 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 3.I for details). Using the batch execution mode we have observed an average runtime speedup of up to an order of magnitude.

Selection Function

To compose appropriate subspaces $\mathcal{K}_n$ a selection function $\mathcal{S}_h(\vec{y}^{\,(n)}, \Theta)$ is defined, which prior to each E-step allows us to select the relevant H' hidden dimensions (i.e., the elements of the index sets I_n) for a given observation $\vec{y}^{\,(n)}$. A selection function is essentially a heuristic-based scoring mechanism, that ranks all the latents based on their potential for being among the generating causes of a given observation. Selection functions can be based on upper bounds for probabilities $p(b_h = 1 \,|\, \vec{y}^{\,(n)}, \Theta)$ (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; Bornschein et al., 2013).

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

$$\mathcal{S}_h(\vec{y}^{\,(n)}, \Theta) = \mathcal{N}(\vec{y}^{\,(n)}; \tilde{W}_{\vec{b}_h}\vec{\mu}, C_{\vec{b}_h}) \propto p(\vec{y}^{\,(n)} \,|\, \vec{b} = \vec{b}_h, \Theta), \tag{3.25}$$

where $\vec{b}_h$ represents a singleton state in which only the entry h is non-zero. The selection function (3.25) is basically the data likelihood given a singleton state $\vec{b}_h$. The function does not take into account the probability of the state itself (i.e., $p(\vec{b}_h \,|\, \Theta)$), as this may introduce a bias against less active latent dimensions. Similar to previously used selection functions (compare e.g., Lücke and Eggert, 2010; Puertas et al., 2010), in order to maintain a linear scaling behavior w.r.t. the number of latents, the selection function introduced here avoids computationally demanding higher-order combinatorics of the latents by only assessing one-to-one correspondences between individual latents and an observed data point. In the next section we empirically evaluate the efficacy of our selection function by means of numerical experiments that are based on the KL-divergence between the exact and the approximated posteriors computed from the subspaces $\mathcal{K}_n$.

Equations (3.21) to (3.23) 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.

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 in parallel on multiple CPUs with no dependency on their arrangement as physically collocated arrays with shared memory or distributed among multiple compute nodes (see Bornschein et al., 2010, for more details). We further extended the basic technique to make our implementation more efficient and suitable for parallelization by applying the batch execution (the observation discussed in Section 3.iv.1 on Computational Complexity and Appendix 3.I). In all the experiments, the GSC model parameters were randomly initialized.[2] The choice of GSC truncation parameters H' and γ 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 truncation parameters are therefore capped by the available computational resources. However, empirically we observed that often much smaller values were sufficient than those that are maximally affordable.[3] Note that factored variational approaches do not usually offer such a trade-off between the exactness and computational demand of their inference schemes by means of a simple parameter adjustment.

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 (3.13) and truncated (3.23) 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 $\mathcal{K}_n$ and the overall posterior mass (compare Lücke and Eggert, 2010):

$$Q^{(n)} \;=\; \frac{\sum_{\vec{b}\in\mathcal{K}_n}\int_{\vec{z}} p(\vec{b},\vec{z}\,|\,\vec{y}^{\,(n)},\Theta)\,\mathrm{d}\vec{z}}{\sum_{\vec{b}'}\int_{\vec{z}'} p(\vec{b}\,',\vec{z}\,'\,|\,\vec{y}^{\,(n)},\Theta)\,\mathrm{d}\vec{z}\,'} \;=\; \frac{\sum_{\vec{b}\in\mathcal{K}_n}\mathcal{B}(\vec{b};\vec{\pi})\,\mathcal{N}(\vec{y}^{\,(n)};\tilde{W}_{\vec{b}}\,\vec{\mu},C_{\vec{b}})}{\sum_{\vec{b}'}\mathcal{B}(\vec{b}\,';\vec{\pi})\,\mathcal{N}(\vec{y}^{\,(n)};\tilde{W}_{\vec{b}'}\vec{\mu},C_{\vec{b}'})}\,, \tag{3.26}$$

where the integrals over the latent $\vec{z}$ in (3.26) are again given in closed-form. The metric $Q^{(n)}$ ranges from zero to one and is directly related to the KL-divergence between the approximation q_n in Equation (3.23) and the true posterior:

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

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^{gen}. Each bar is then randomly assigned

[2] We randomly and uniformly initialized the π_h between 0.05 and 0.95. $\vec{\mu}$ was initialized with normally distributed random values and the diagonal of Ψ was initialized with strictly positive uniformly distributed random values. We set Σ 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.

[3] Compare Appendix 3.H for trade-off between complexity and accuracy of the truncated EM approach.

either a positive or negative value with magnitude 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[4] observed noise $\Sigma^{\text{gen}} = \sigma^2 I_D$, where $\sigma^2 = 2.0$. 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 consists of 50 EM iterations and after each run we compute 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 (3.22) constructed from the H' latents chosen through the selection function (3.25) contain almost the entire posterior probability mass in this case. The small fraction of remaining posterior mass lies in other discrete subspaces and its principle form is known to not contain any heavy tails (see Equation (3.16)). The contribution of the truncated posterior mass to parameter updates can therefore be considered negligible.

Consistency

Prior to delving into a comparative analysis of GSC with other methods, we assess the consistency of the approach by applying both its exact and truncated variational inference schemes on the task of recovering sparse latent directions w.r.t. increasing numbers of training data. For this experiment we work with synthetic data generated by the GSC model itself. Moreover, we also apply the truncated variational inference on standard sparse coding data generated with a standard Laplace prior(Olshausen and Field, 1996; Lee et al., 2007). Taking into account the computational demands of the exact inference, we set both the hidden as well as observed dimensions (H and D respectively) to 10. For the experiment we exponentially increase N from 1000 to 512000. For each trial in the experiment we generate a new ground-truth mixing matrix $W^{\text{gen}} \in \mathbb{R}^{D \times H}$ by randomly generating a set of H orthogonal bases and perturbing them with a Gaussian noise with zero mean and a variance of 2.0. We set the sparsity parameters π_h to $1/H$, while the observed noise is assumed to be homoscedastic with $\sigma = 1.0$. When generating data with a spike-and-slab prior, the slab is considered to have its mean at zero with an identity covariance matrix, i.e., $\mu_h = 0.0$ for all $h \in H$ and $\Psi^{\text{gen}} = I_H$, respectively. In each trial after performing 100 EM iterations and inferring the whole set of GSC parameters Θ, we quantify the quality of the inference in terms of how well the inferred bases W align with the corresponding ground truth bases W^{gen}. As a measure of discrepancy between the generating and the recovered bases we use the Amari index (Amari et al., 1995):

$$A(W) \;=\; \frac{1}{2H(H-1)} \sum_{h,h'=1}^{H} \Big(\frac{|O_{hh'}|}{\max_{h''} |O_{hh''}|} + \frac{|O_{hh'}|}{\max_{h''} |O_{h''h'}|} \Big) - \frac{1}{H-1} \tag{3.27}$$

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

[4] To infer homoscedastic noise we set in the M-step the updated noise matrix Σ to $\sigma^2 I_D$ where $\sigma^2 = \text{Tr}(\Sigma)/D$. This is equivalent to parameter update for σ^2 if the model originally assumes homoscedastic noise.

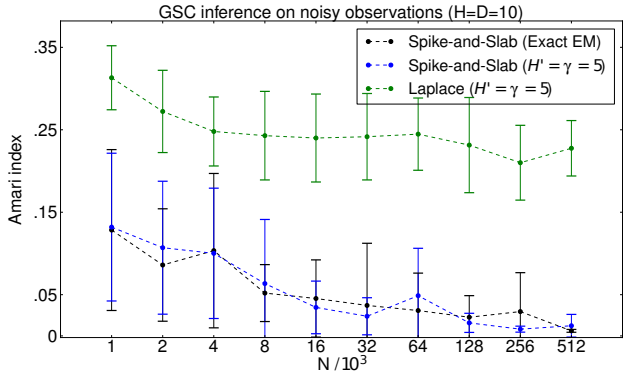

Figure 3.3: Numerical experiment investigating the consistency of the exact as well as the truncated variational GSC algorithm for increasing numbers of data points. The curves show results for the recovery of sparse directions for different numbers of data points. Data points were generated by both the spike-and-slab generative model (black and blue) and a standard sparse coding model with Laplace prior (green). The curves show the mean Amari index and standard deviations computed based on 15 repetitions of the learning algorithm.

Figure 3.3 summarizes the results of the experiment. Each error bar in the plot extends one standard deviation on both sides of its corresponding mean Amari index, which is computed from 15 repetitions. The black curve shows the results of the exact GSC inference on spike-and-slab generated data, while the blue and green curves illustrate the results of the truncated variational inference ($H' = \gamma = 5$) on data generated by spike-and-slab and Laplace priors respectively. For data generated with the spike-and-slab prior, we observe a gradually more accurate recovery of the sparse directions, as the mean Amari indices gradually converge towards the minimum value of zero for increasing numbers of training data. The minimum Amari index values that we obtain for the black and blue curves for $N \in \{128K, 256K, 512K\}$ are all below 6×10^{-3}. For the standard sparse coding data, we also see an improvement in performance with more data; however, higher mean values of the Amari index in this case can presumably be attributed to the model mismatch.

Recovery of Sparse Directions on Synthetic Data

In our first comparison with other methods, we measure the performances of GSC (using the truncated variational approximation) and MTMKL (which uses a factored variational approximation) approaches on the sparse latent direction recovery task given 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 another set of experiments we use the standard Laplace distribution as a prior (Olshausen and Field, 1996; Lee et al., 2007). For each trial in the experiments a new mixing matrix W^{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 repeat 15 trials per given dimensionality. For each trial we randomly generated a new data set of $N = 5000$ noisy observations with $\Sigma^{\text{gen}} = I_D$. Per trial, we perform 50 iterations of both algorithms. The GSC truncation parameters H' and γ were set to $\frac{H}{10}$. We assess

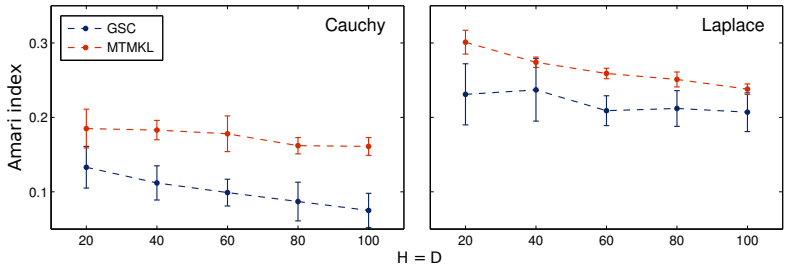

Figure 3.4: Performance of GSC (with $H' = \gamma = \frac{H}{10}$) vs. MTMKL on data generated by standard sparse coding models both with Cauchy and Laplace priors. Performance compared on the Amari index (3.27).

the performances of the algorithms w.r.t. the Amari index (3.27).

The results for GSC and MTMKL in Figure 3.4 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.

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. Figure 3.5 shows the performance of both the methods based on three different source separation benchmarks obtained from (ICALAB; Cichocki et al., 2007). The error bars show two standard deviations estimated based on 15 trials per experiment. The x-axis in the figure represents the degree of orthogonality of the ground truth mixing bases W^{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 $\sigma \in \{4, 10, 20\}$, respectively. For Figure 3.5 no observation noise was added to the mixed sources. For both the algorithms we performed 100 iterations per run.[5] The GSC truncation parameters H' and γ 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 data set 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 data sets. Figure 3.6 shows the performance in the case of *Speech*20 (with added Gaussian noise with $\sigma = 2.0$), for instance. Along

[5]For the MTMKL algorithm we observed convergence after 100 iterations while the GSC algorithm continued to improve with more iterations. However, allowing the same number of iterations to both the algorithms, the reported results are obtained with 100 iterations.

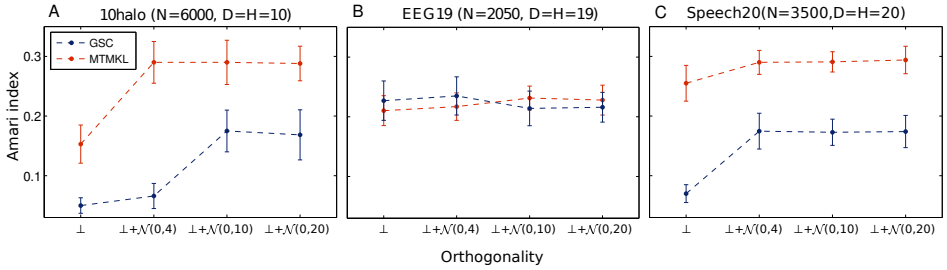

Figure 3.5: Performance of GSC vs. MTMKL on source separation benchmarks with varying degrees of orthogonality of the mixing bases. The orthogonality on the x-axis varies from being orthogonal $\perp$ to increasingly non-orthogonal mixing as randomly generated orthogonal bases are perturbed by adding Gaussian noise $\mathcal{N}(0, \sigma)$ to them. No observation noise was assumed for these experiments. Performances are compared on the Amari index (3.27).

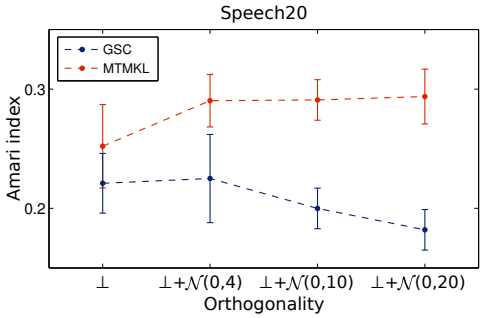

Figure 3.6: Performance of GSC vs. MTMKL in terms of the Amari index (3.27) on the Speech20 benchmark with varying degrees of orthogonality of the mixing bases and Gaussian noise (with $\sigma = 2$) added to observed data. The orthogonality on the x-axis varies from being orthogonal $\perp$ to increasingly non-orthogonal mixing as randomly generated orthogonal bases are perturbed by adding Gaussian noise $\mathcal{N}(0, \sigma)$ to them.

the x-axis orthogonality decreases, again. While the performance of MTMKL decreases with decreasing orthogonality, performance of GSC increases in this case. For other data sets increased observation noise may not have such effects, however (see Appendix, Figure 3.12 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 (SPAMS; Mairal et al., 2009a) and the efficient sparse coding algorithms (ESCA; Lee et al., 2007). These methods are based on linear regression with lasso regularization, where sparsity is induced by introducing a parameter-regulated penalty term in the objective function,[6] which penalizes the L_1-norm of regressors (or latent variables). In a probabilistic context this is equivalent to assuming a Laplace prior on the regressors. In this experiment we test the performance 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

[6] For both the algorithms compared here, optimal values for sparsity controlling regularization parameters were chosen through cross-validation.

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

Table 3.1: Performance of GSC, MTMKL and other publicly available sparse coding algorithms on benchmarks for source separation. Performances are compared based on the Amari index (3.27). Bold values highlight the best performing algorithm(s).

new randomly generated orthogonal data mixing matrix W^{gen} and new parameter initialization in each trial. The GSC inference was exact for these experiments with better results obtained with observed noise constrained to be homoscedastic. 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.

Table 3.1 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 comparably 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.[7] Further improvements on some settings in Table 3.1 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.

Figure 3.2 was generated in a similar fashion on the *10halo* data set. There we computed the exact posterior (3.13) over $H = 10$ latent dimensions, thus the approximation parameters were $\gamma = H' = H$ (exact E-step). After performing 50 EM iterations and learning all the model parameters, we then visualized marginalized posteriors for a given data point along each column of the figure. The top row of the figure allows us get an idea of how concentrated and sparse a data point is in terms of the latents contributing to its posterior mass. The bottom row of the figure on the other hand allows us to observe the sparsity in the posterior w.r.t. the dimensionality of the hyperplanes spanned by the latents, with a posterior mass accumulation in low-dimensional hyperplanes.

Computational Complexity vs. Performance

In terms of computational complexity, GSC and MTMKL algorithms are significantly different, so we also looked at the trade-off between their computational

[7] In Table 3.1 the results do not necessarily improve with an increased number of data points. However, the data points considered here 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.

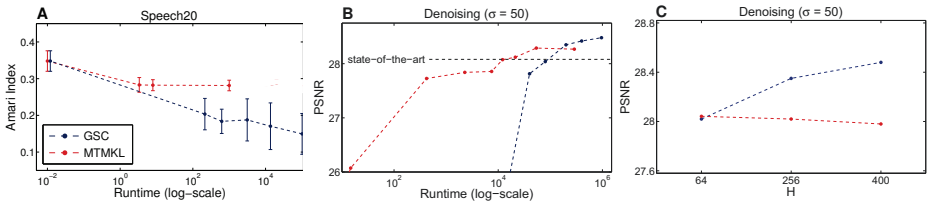

Figure 3.7: **A,B:** Runtime vs. performance comparison of GSC (blue) and MTMKL (red) on source separation and denoising tasks. Source separation is compared on the Amari index (the lower the better) while the denoising is compared on the peak signal-to-noise (PSNR) ratio (the higher the better). **C**: Performance of GSC (blue) and MTMKL (red) on the denoising task against an increasing number of latents.

costs versus performance. Subfigures A and B in Figure 3.7 show 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 and the number of iterations going up to 12K. The GSC plot was generated from $H = 400$ latents with H' and γ being 10 and 5 respectively. The last point on the GSC (blue) curve corresponds to the $120th$ EM iteration. 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.

For the denoising task we also compared the performance of both the algorithms against an increasing number of latents H. While the computational cost of the MTMKL algorithm increases linearly w.r.t. H, the runtime cost of the truncated variational GSC remains virtually unaffected by it, since it scales w.r.t. the parameters H' and γ (see Section 3.iv.1). In this experiment we performed 65 iterations of the GSC algorithm for $H \in \{64, 256\}$ and up to 120 iterations for $H = 400$. For MTMKL we performed up to 120 iterations for each given H. Figure 3.7C summarizes the results of this experiment. In the figure we can see a constant performance increase for GSC, while for MTMKL we actually observe a slight decrease in performance. This is in conformity with what Titsias and Lazaro-Gredilla (2011) report in their work that for the denoising task they observed no performance improvements for larger number of latents.

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). We also compare with the approach by Zhou et al. (2009) as a representative sampling-based optimization scheme. For the task a noisy input image is generated by adding Gaussian noise (with zero mean and standard deviation determining the noise level) to the

256×256 image (see Figure 4.4). Following the previous studies, we generated $62,001$ overlapping (shifted by 1 pixel) 8×8 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 γ for each run are listed in Table 3.2. We assumed homoscedastic observed noise with a priori unknown variance in all these experiments (as the MTMKL model).

A comprehensive comparison of the denoising results of the various algorithms is shown in Table 3.2, where performance is measured 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 Figure 4.4 we show our result for noise level $\sigma = 25$. 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 probabilities to low ones). We also applied GSC with higher numbers of latent dimensions: Although for low noise levels of $\sigma = 15$ and $\sigma = 25$ we did not measure significant improvements, we observed a further increase for $\sigma = 50$. For instance, with $H = 400$, $H' = 10$ and $\gamma = 8$, we obtained for $\sigma = 50$ a PSNR of 28.48dB.

As for source separation described in Section 3.v.5, we also compared performance vs. computational demand of both algorithms for the task of image denoising. As illustrated in A and B of Figure 3.7, MTMKL performs better when computational resources are relatively limited. However, when increasingly more computational resources are made available, MTMKL does not improve much further on its performance while GSC performance continuously increases and eventually outperforms MTMKL on this task.

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

Table 3.2: 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 Beta process (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).

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

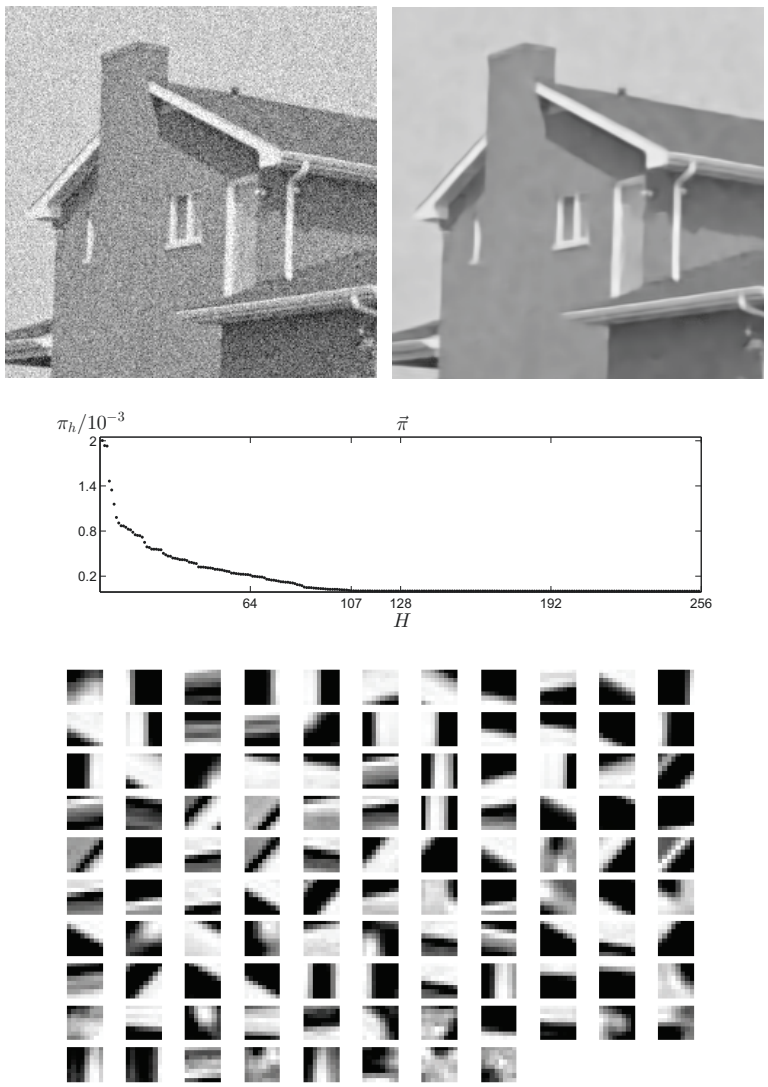

Figure 3.8: Top left: Noisy "house" image with $\sigma = 25$. Top right: GSC denoised image. Middle: Inferred sparsity values π_h in descending order indicate that finally around 107 of in total 256 latent dimensions significantly contribute to model the data. Bottom: Basis functions (ordered from left to right, top to bottom) corresponding to the first 107 latent dimensions sorted w.r.t. the decreasing sparsity values π_h .

efficient and precise. Many sparse coding algorithms based on the MAP estimation are continuously being developed and are successfully applied in a variety of settings (e.g., Lee et al., 2007; Mairal et al., 2009a). 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 by Mohamed et al. (2012) uses Bayesian methodology involving conjugate priors and hyperparameters 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., 2013). While these approaches are all based on spike-and-slab generalizations of sparse coding (like fully Bayesian approaches), they maintain deterministic approximation procedures for parameter optimization. Variational approximations allow for applications to large hidden spaces which pose 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 Figure 3.10 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. (2013) 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 (MacKay, 2001; Ilin and Valpola, 2005; 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, 2005) 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 (Figure 3.4). The same applies for experiments for unmixing real signals in which we increased the non-orthogonality (Figure 3.5A,C; suppl. Figure 3.12); although for some data performance is very similar (Figure 3.5B). Also if sources are mixed orthogonally, we usually observe better performance of the truncated variational approach (Table 3.1), 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 data sets 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 Table 3.1, 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., 2009a) and ESCA (Lee et al., 2007) (see Table 3.1). This may be taken as evidence for the better suitability of a spike-and-slab prior for such types of data.

For source separation our approach (like conventional sparse coding or ICA) seeks to infer sparse directions by capturing the sparse, latent structures from the spatial domain of the input signals. However, when dealing with data that also carry a temporal structure (e.g., speech or EEG recordings), other approaches which explicitly model temporal regularities such as Hidden Markov Models (HMMs) may as well be a more natural and (depending on the task) a more suitable choice. Such methodologies can in principle be combined with the sparse coding approaches studied and compared here to form more comprehensive models for spatio-temporal data, which can yield improved performance on blind source separation tasks (compare e.g., Gael et al., 2008; Mysore et al., 2010).

In the last experiment of this study, we finally compared the performance of factored and truncated variational approximations on a standard image denoising task (see Table 3.2). 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 Table 3.2 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 over-complete 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 $H = 400$). As we assume an independent Bernoulli prior per latent dimension, GSC can also prune out latent dimensions by inferring very low values of π_h for the bases that make negligible contribution in the inference procedure. This can be observed in Figure 4.4, 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. 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 Table 3.2) or when relatively limited computational resources are available (see Figure 3.7).

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 EM approach is a competitive method. It shows that posterior dependencies and multi-modality can be captured by a scalable deterministic approximation. Furthermore, the direct comparison with a factored variational approach in source separation experiments 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 well 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 and learning in such models.

Acknowledgements: We acknowledge funding by the German Research Foundation (DFG), grant LU 1196/4-2, and by the German Ministry of Research and Education (BMBF), grant 01GQ0840 (BFNT Frankfurt). Furthermore, we acknowledge support by the Frankfurt Center for Scientific Computing (CSC Frankfurt).

Appendix

Derivation of M-step Equations

Our goal is to optimize the free-energy w.r.t. Θ:

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

where

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

and

$$\log\big(p(\vec{b}\,|\Theta)\big) = \sum_{h=1}^{H} \log\Big(\pi_h^{b_h}\,(1-\pi_h)^{1-b_h}\Big) \,.$$

The free-energy thus takes the form:

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

where $q_n(\vec{b}, \vec{z}; \Theta^{\text{old}})$ denotes the posterior $p(\vec{b}, \vec{z} \,|\, \vec{y}^{(n)}, \Theta^{\text{old}})$. Now we can derive the M-step equations (3.6) to (3.10) by canonically setting the derivatives of the free-energy above w.r.t. each parameter in Θ to zero.

Optimization of the Data Noise

Let us start with the derivation of the M-step equation for Σ:

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

$$\begin{aligned}
\Rightarrow \Sigma &= \frac{1}{N}\sum_{n=1}^{N}\sum_{\vec{b}}\int_{\vec{z}} q_n(\vec{b},\vec{z};\Theta^{\text{old}})\Big[\left(\vec{y}^{(n)} - W(\vec{b}\odot\vec{z})\right)\left(\vec{y}^{(n)} - W(\vec{b}\odot\vec{z})\right)^{\mathrm{T}}\Big]\mathrm{d}\vec{z} \\
&= \frac{1}{N}\sum_{n=1}^{N}\Big[\Big(\vec{y}^{(n)} - W\big\langle(\vec{b}\odot\vec{z})\big\rangle_n\Big)\Big(\vec{y}^{(n)} - W\big\langle(\vec{b}\odot\vec{z})\big\rangle_n\Big)^{\mathrm{T}} \\
&\qquad\qquad + W\big[\big\langle(\vec{b}\odot\vec{z})(\vec{b}\odot\vec{z})^{\mathrm{T}}\big\rangle_n - \big\langle(\vec{b}\odot\vec{z})\big\rangle_n\big\langle(\vec{b}\odot\vec{z})\big\rangle_n^{\mathrm{T}}\big]W^{\mathrm{T}}\Big] \\
&= \frac{1}{N}\sum_{n=1}^{N}\Big[\vec{y}^{(n)}(\vec{y}^{(n)})^{\mathrm{T}} - W\big[\big\langle(\vec{b}\odot\vec{z})\big\rangle_n\big\langle(\vec{b}\odot\vec{z})\big\rangle_n^{\mathrm{T}}\big]W^{\mathrm{T}}\Big],
\end{aligned}$$

where $\langle\cdot\rangle_n$ denotes the expectation value in Equation (3.5).

Optimization of the Bases

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

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

$$\Rightarrow W = \frac{\sum_{n=1}^{N}\vec{y}^{(n)}\big\langle\vec{b}\odot\vec{z}\big\rangle_n^{\mathrm{T}}}{\sum_{n=1}^{N}\big\langle(\vec{b}\odot\vec{z})(\vec{b}\odot\vec{z})^{\mathrm{T}}\big\rangle_n}.$$

Optimization of the Sparsity Parameter

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

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

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

Optimization of the Latent Mean

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

$$\begin{aligned}\frac{\partial}{\partial\vec{\mu}}\mathcal{F}(\Theta^{\text{old}},\Theta) &= \sum_{n=1}^{N}\sum_{\vec{b}}\int_{\vec{z}} q_n(\vec{b},\vec{z};\Theta^{\text{old}})\\ &\qquad\Big[-\tfrac{1}{2}\tfrac{\partial}{\partial\vec{\mu}}\big((\vec{z}-\vec{\mu})\odot\vec{b}\,\big)^{\mathrm{T}}\big(\Psi\odot\vec{b}\vec{b}^{\mathrm{T}}\big)^{-1}\big((\vec{z}-\vec{\mu})\odot\vec{b}\,\big)\Big]\mathrm{d}\vec{z}\\ &= \sum_{n=1}^{N}\sum_{\vec{b}}\int_{\vec{z}} q_n(\vec{b},\vec{z};\Theta^{\text{old}})\Big[\big(\Psi\odot\vec{b}\vec{b}^{\mathrm{T}}\big)^{-1}\Big((\vec{z}-\vec{\mu})\odot\vec{b}\,\Big)\Big]\mathrm{d}\vec{z}\overset{!}{=}0\end{aligned}$$

$$\Rightarrow \vec{\mu} = \frac{\sum_{n=1}^{N}\big\langle\vec{b}\odot\vec{z}\big\rangle_n}{\sum_{n=1}^{N}\big\langle\vec{b}\big\rangle_n}.$$

Optimization of the Latent Covariance

Lastly we derive the M-step update for the latent covariance Ψ:

$$\begin{aligned}&\frac{\partial}{\partial\Psi}\mathcal{F}(\Theta^{\text{old}},\Theta)\\ &\quad= \sum_{n=1}^{N}\sum_{\vec{b}}\int_{\vec{z}} q_n(\vec{b},\vec{z};\Theta^{\text{old}})\\ &\qquad\Big[-\tfrac{1}{2}\tfrac{\partial}{\partial\Psi}\Big(\log|\Psi\odot\vec{b}\vec{b}^{\mathrm{T}}|\Big)-\tfrac{1}{2}\tfrac{\partial}{\partial\Psi}\big((\vec{z}-\vec{\mu})\odot\vec{b}\,\big)^{\mathrm{T}}\big(\Psi\odot\vec{b}\vec{b}^{\mathrm{T}}\big)^{-1}\big((\vec{z}-\vec{\mu})\odot\vec{b}\,\big)\Big]\mathrm{d}\vec{z}\\ &\quad= \sum_{n=1}^{N}\sum_{\vec{b}}\int_{\vec{z}} q_n(\vec{b},\vec{z};\Theta^{\text{old}})\\ &\qquad\Big[-\tfrac{1}{2}\big(\Psi\odot\vec{b}\vec{b}^{\mathrm{T}}\big)^{-1}+\tfrac{1}{2}\big(\Psi\odot\vec{b}\vec{b}^{\mathrm{T}}\big)^{-2}\big((\vec{z}-\vec{\mu})\odot\vec{b}\,\big)\big((\vec{z}-\vec{\mu})\odot\vec{b}\,\big)^{\mathrm{T}}\Big]\mathrm{d}\vec{z}\overset{!}{=}0\end{aligned}$$

$$\begin{aligned}\Rightarrow\Psi &= \sum_{n=1}^{N}\Big[\big\langle(\vec{z}-\vec{\mu})(\vec{z}-\vec{\mu})^{\mathrm{T}}\odot\vec{b}\vec{b}^{\mathrm{T}}\big\rangle_n\Big]\odot\Big(\sum_{n=1}^{N}\Big[\big\langle\vec{b}\vec{b}^{\mathrm{T}}\big\rangle_n\Big]\Big)^{-1}\\ &= \sum_{n=1}^{N}\Big[\big\langle(\vec{b}\odot\vec{z})(\vec{b}\odot\vec{z})^{\mathrm{T}}\big\rangle_n-\big\langle\vec{b}\vec{b}^{\mathrm{T}}\big\rangle_n\odot\vec{\mu}\vec{\mu}^{\mathrm{T}}\Big]\odot\Big(\sum_{n=1}^{N}\Big[\big\langle\vec{b}\vec{b}^{\mathrm{T}}\big\rangle_n\Big]\Big)^{-1}.\end{aligned}$$

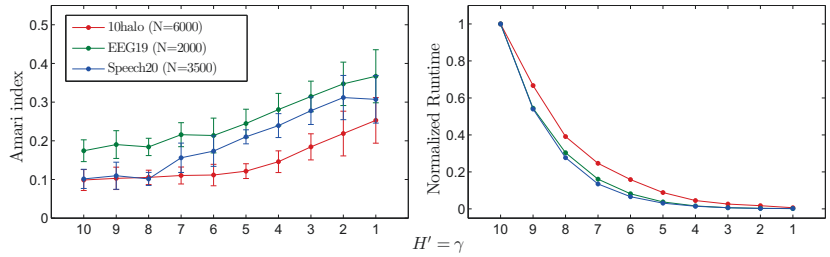

Figure 3.9: Performance of the GSC on 10halo, EEG19 and Speech20 benchmarks for decreasing truncation parameters H' and γ. The right plot shows how the computational demand of the truncated variational algorithm decreases with decreasing values of the truncation parameters. The runtime plots are normalized by the runtime value obtained for $H' = \gamma = 10$ for each of the benchmarks.

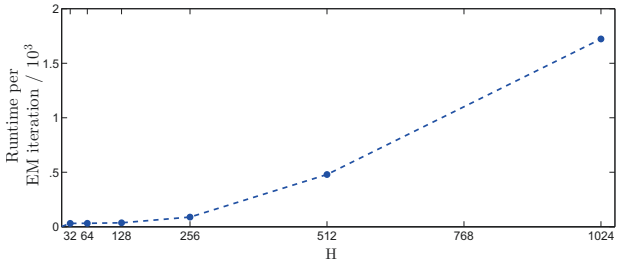

Figure 3.10: Time scaling behavior of GSC for increasing latent dimensions H and fixed truncation parameters H' and γ.

Performance vs. Complexity Trade-Off

If the approximation parameters H' and γ are held constant, the computational cost of the algorithm scales with the computational cost of the selection function. If the latter cost scales linearly 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 Figure 3.10). In experiments with H increasing from 16 to 1024, we observed a, finally, close to linear increase of computational costs. However, a larger H implies a larger number of parameters, and thus may require more data points to prevent over-fitting. Although a larger data set increases computational demand, our truncated approximation algorithm allows us to take advantage of parallel computing architecture in order to more efficiently deal with large data sets (see Appendix 3.I 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). Figure 3.9 shows that a high accuracy can still be achieved for relatively small values of H' γ which, at the same time, results in strongly reduced computational demands.

Dynamic Data Repartitioning for Batch/Parallel Processing

As described in Section 3.IV, the truncated variational approach deterministically selects the most likely H' causes of a given observation $\vec{y}$ for efficiently approximating the posterior distribution over a truncated latent space. In practice one can also use the selected latent causes for applying clustering to the observed data, which allows for an efficient and parallelizable batch-mode implementation of the E-step of the truncated variational EM algorithm.

In the batch processing mode, prior to each E-step the observed data can be partitioned by clustering together the data points w.r.t. their selected latent causes. The resulting clusters can then be processed individually (e.g., on multiple compute cores) to perform the E-step (Equations (3.21) to (3.23)) for all data points in a given cluster. This approach not only pursues a natural partitioning of data, but in a parallel execution environment, it can prove to be more efficient than uniformly distributing data (as in Bornschein et al., 2010) among multiple processing units. By maximizing the similarity (in latent space) of individual data points assigned to each of the processing units, we can overall minimize the number of redundant computations involved in Equations (3.15) and (3.23), that are tied to specific states of the latents. This can be observed by considering Equation (3.21), which is as follows:

$$p(\vec{b},\vec{z}\,|\,\vec{y}^{\,(n)},\Theta) \;\approx\; \frac{\mathcal{N}(\vec{y}^{\,(n)};\vec{\mu}_{\vec{b}},C_{\vec{b}})\,\mathcal{B}(\vec{b};\vec{\pi})\,\mathcal{N}(\vec{z};\,\vec{\kappa}^{\,(n)}_{\vec{b}},\Lambda_{\vec{b}})}{\sum_{\vec{b}'\in\mathcal{K}_n}\mathcal{N}(\vec{y}^{\,(n)};\vec{\mu}_{\vec{b}'},C_{\vec{b}'})\,\mathcal{B}(\vec{b}';\vec{\pi})}\,\delta(\vec{b}\in\mathcal{K}_n). \quad (3.28)$$

Here the parameters $\vec{\mu}_{\vec{b}}, C_{\vec{b}}$ and $\Lambda_{\vec{b}}$ entirely depend on a particular latent state $\vec{b}$. Also, $\vec{\kappa}^{\,(n)}_{\vec{b}}$ takes prefactors that can be precomputed given the $\vec{b}$. It turns out that to compute (3.28) our clustering-based, dynamic data repartitioning and redistribution strategy is more efficient than the uniform data distribution approach of Bornschein et al. (2010). This is illustrated in Figure 3.11, which shows empirical E-step speedup over the latter approach 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.[8] Each trial was run in parallel on 24 computing nodes. The red 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 every E-step. All the reported results here also take into account the cost of data clustering and repartitioning.

In a parallel setup, we perform the data clustering process by having each processing unit cluster its own data locally and then merging the resulting clusters globally. In order to avoid uneven data distribution, we also bound the maximum size of a cluster. Currently we pick (per iteration) top α percentile of occurring

[8] The observed and the latent dimensions of the GSC model were 25 and 20 respectively. The truncation approximation parameters H' and γ (maximum number of active causes in a given latent state) were 8 and 5 respectively.

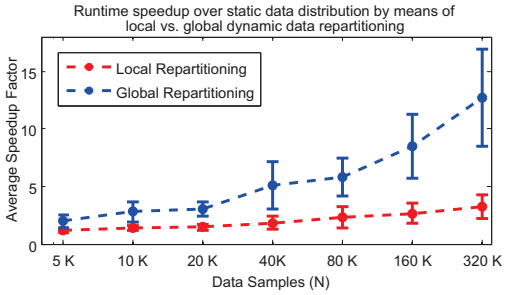

Figure 3.11: Runtime speedup of the truncated variational E-step (Equations (3.21) to (3.23)) with the static data distribution strategy taken as a baseline. The red plot shows the speedup when initially uniformly distributed data samples were only clustered locally by each processing unit, while the 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.

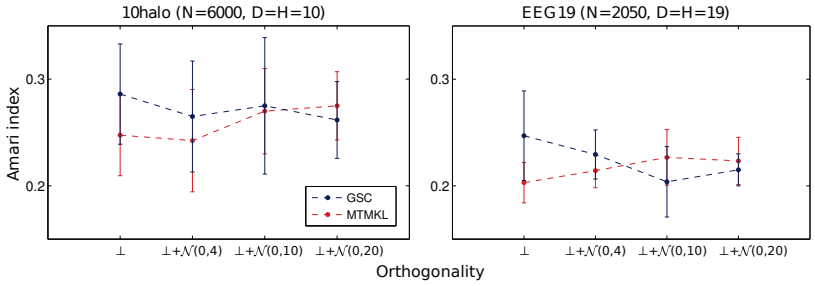

Figure 3.12: Source separation with observation noise. Performance of GSC vs. MTMKL on 10halo and EEG19 benchmarks with varying degrees of orthogonality of the mixing bases and Gaussian noise added to observations. Performance of GSC vs. MTMKL on the Speech20 benchmark with varying degrees of orthogonality 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 (3.27).

cluster sizes as the threshold.[9] Any cluster larger than α is evenly broken into smaller clusters of maximum size α. Moreover, to minimize communication overhead among computational units, we actually only cluster and redistribute the data indices. This entails that the actual data must reside in a shared memory structure which is efficiently and dynamically accessible by all the computational units. Alternatively, all the units require their own copy of the whole data set.

Here we have introduced and illustrated the gains of dynamic data repartitioning technique in the context of a specific sparse coding model, which in fact involves computationally expensive, state-dependent operations for computing posterior distributions. The technique however is inherently generic and can be straightforwardly employed for other types of multi-causal models.

[9]The α for the reported experiments was 5.

Chapter 4

A Truncated Sampler for Efficient Inference in Spike-and-Slab Sparse Coding

Parts of this chapter appeared in Proc. NIPS, pp. 3927-3935, 2016.

In previous chapters we have seen that posterior inference in spike-and-slab sparse coding has to take into account highly multi-modal probability landscapes. Although for a Gaussian slab it is possible to compute the exact posteriors (see e.g., Chapter 2), it is the exponential scaling of the number of probability modes with the number of latents which makes it computationally infeasible to perform the exact inference on large scales. The scalability of spike-and-slab sparse coding is therefore contingent on approximate posterior inference, although the posterior multimodality implies that a number of established approximation techniques can be relatively coarse for assuming uni-modality for posterior estimation (e.g., Ribeiro and Opper, 2011; Yoshida and West, 2010; Mairal et al., 2009a; Zhou et al., 2009; Seeger, 2008; Lee et al., 2007; Olshausen and Field, 1997).

Several approximate inference methods however have been proposed for Gaussian spike-and-slab sparse coding. They exploit the Gaussian slab to deal with the multi-modal nature of the posterior inference while incurring an amenable computational cost. For instance in Chapter 3, building upon the idea of truncating the exact posterior as an approximation (Lücke and Eggert, 2010), we introduce a truncated variational method for Gaussian spike-and-slab sparse coding. The method is capable of capturing multiple modes and its computational cost is independent of the number of latent components. We categorize the approach as variational since for a given observation, it essentially factorizes the posterior distribution by shrinking its domain to a selected latent subspace of concentrated posterior probability, while truncating the rest for carrying a negligible posterior mass in support of

the observation. In the same chapter, we also compared with a factored variational approach for spike-and-slab sparse coding (Goodfellow et al., 2013; Titsias and Lazaro-Gredilla, 2011). The method simplifies the exact posterior through factorization, where each factor is composed of both the binary and the continuous spike-and-slab variables associated with a latent. The method uses multiple modes by sequentially conditioning each factor on the posterior parameter values of the others, while the parameters of the current factor are optimized to decrease the KL divergence of the approximated factored posterior. The method has a linear cost with respect to the number of latents and by applying another approximation step of simultaneously updating all the factors in parallel, Goodfellow et al. (2013) have scaled-up the method to very large latent dimensions (see e.g., Figure 4.6).

Sampling based inference has also been applied for the Gaussian slab in spike-and-slab sparse coding (Titsias and Lazaro-Gredilla, 2011; Mohamed et al., 2012). By factorizing the posterior in the same manner as the factored variational method, samples of both discrete and continuous spike-and-slab variables are jointly drawn from the factor associated with the paired latent variables while conditioning on an observation and the current values of all the other latents. Sequentially drawing values from each factor constructs a Gibbs sampling routine such that a single pass through all the factors results in one complete sample from the multi-modal posterior.

Despite sharing a common computational structure with the variational method, the sequential core of the Gibbs sampling procedure does not allow for a straight-forward parallel extension as implemented by Goodfellow et al. (2013) for the variational inference. On the other hand, sampling based inference is considered to be more accurate when performed under sufficient computational budget, whereas factored variational approaches are known to introduce inference biases (MacKay, 2001; Ilin and Valpola, 2005; Turner and Sahani, 2011). However, computational demands of sampling based methods indeed pose a major challenge against their scalability. Both Titsias and Lazaro-Gredilla (2011) and Mohamed et al. (2012) apply the sampling based inference for Gaussian spike-and-slab sparse coding, but on a relatively much smaller scale with the largest reported application inferring less than 250 latents. As a comparison, Goodfellow et al. (2013) have scaled-up to learn thousands of latent components from many hundreds of thousand data points. Here our focus will be on the development of a highly scalable inference scheme for Gaussian spike-and-slab sparse coding that will allow for sampling to remain partially applicable in even very large latent spaces.

The Select and Sample Framework

We base our method on the hybrid framework of Shelton et al. (2011), who combined the latent preselection of Lücke and Eggert (2010) with sampling to propose a computational

model of efficient multi-modal inference in neural processing. Shelton et al. (2011) demonstrated the effectiveness of the framework on a scaled-up version of a linear sparse coding model with binary latents (Henniges et al., 2010). Later they also applied the framework for a nonlinear spike-and-slab sparse coding model (Shelton et al., 2012, 2015).

Latent Preselection

Select and sample is a framework for efficiently approximating the true latent posterior $p(\vec{s}\,|\,\vec{y}^{\,(n)}, \Theta)$ with another distribution conventionally referred to as $q_n(\vec{s}; \Theta)$, i.e.:

$$q_n(\vec{s}; \Theta) \approx p(\vec{s}\,|\,\vec{y}^{\,(n)}, \Theta) = \frac{p(\vec{y}^{\,(n)}\,|\,\vec{s}, \Theta)\, p(\vec{s}\,|\,\Theta)}{\int p(\vec{y}^{\,(n)}\,|\,\vec{s}\,', \Theta)\, p(\vec{s}\,'\,|\,\Theta) \mathrm{d}\vec{s}\,'}.$$

According to Lücke and Eggert (2010), if we can expect the latent acitivity $s_h, h \in \{1 \ldots H\}$ to be intrinsically sparse for any given observation $\vec{y}^{\,(n)}$, we can dramatically reduce the computational complexity of $q_n(\vec{s}; \Theta)$ by limiting its domain to the most relevant latents with respect to $\vec{y}^{\,(n)}$. To that end, $q_n(\vec{s}; \Theta)$ can be defined by truncating the true posterior $p(\vec{s}\,|\,\vec{y}^{\,(n)}, \Theta)$ such that it only has support on a subset $\mathcal{K}_n$ of the entire latent state space:

$$q_n(\vec{s}; \Theta) = \frac{p(\vec{s}\,|\,\vec{y}^{\,(n)}, \Theta)}{\displaystyle\int_{\vec{s}\,' \in \mathcal{K}_n} p(\vec{s}\,'\,|\,\vec{y}^{\,(n)}, \Theta)}\; \delta(\vec{s} \in \mathcal{K}_n), \tag{4.1}$$

where $\delta(\vec{s} \in \mathcal{K}_n)$ is an indicator function, taking the value $\delta(\vec{s} \in \mathcal{K}_n) = 1$ only if $\vec{s} \in \mathcal{K}_n$ and zero otherwise. For a discrete distribution the integral in (4.1) is replaced by a summation. $\mathcal{K}_n$ is defined on index sets $I_n \subseteq \{1, \ldots, H\}$, which hold the indices of $H' \ll H$ latents that are deemed to be the most relevant causes of $\vec{y}^{\,(n)}$:

$$\mathcal{K}_n \;=\; \{\vec{s}\;|\;\forall h \notin I_n,\, s_h = 0\}. \tag{4.2}$$

The indices contained by I_n have the highest values of a *selection function* $\mathcal{S}_h(\vec{y}^{\,(n)}, \Theta)$. The set Θ may contain model as well as other parameters that are specific to the selection function. A simple example of a selection function is cosine similarity between a data point and the basis vectors $\vec{W}^{(h)}$ as for instance used by Shelton et al. (2011). Other kinds of selection functions are employed by e.g., Bornschein et al. (2010); Dai and Lücke (2012); Sheikh et al. (2014).

Sampling

In the EM based optimization of LVMs, the latent posteriors $p(\vec{s}\,|\,\vec{y}^{\,(n)}, \Theta)$ in the E-step are essentially computed for finding sufficient statistics, which in general can be defined as

expected values $\langle \cdot \rangle_n$ of a function of latent variables, i.e.:

$$\langle f(\vec{s}) \rangle_n = \int f(\vec{s})\, p(\vec{s} \,|\, \vec{y}^{(n)}, \Theta) \mathrm{d}\vec{s}. \tag{4.3}$$

However, as (4.3) often turns out to be either analytically or computationally intractable, one can instead approximate the statistic as:

$$\langle f(\vec{s}) \rangle_n \approx \frac{1}{M} \sum_{m=1}^{M} f(\vec{s}^{\,(m)}), \tag{4.4}$$

where according to the law of large numbers, for randomly drawn $\vec{s}^{\,(m)} \sim p(\vec{s} \,|\, \vec{y}^{(n)}, \Theta)$ the approximation (4.4) converges to (4.3) as $M \to \infty$. Hence the *Monte Carlo* method (4.4) gives us a feasible means of approximating the solution to an otherwise intractable computational problem. In practice however, there are many issues involved with sampling which can be as basic as how many samples maybe sufficient or more convoluted such as no closed-form of the posterior making it difficult to draw samples from it. For cases where it is not possible to directly sample from the target distribution, there are methods such as *Acceptance sampling* or *Importance sampling*, which draw samples from a surrogate proposal distribution (e.g., a uniform distribution). The drawn samples are then accepted or weighted according to the actual distribution to compute the target expectations. Efficiency of such methods naturally depends on the proximity of the target and the proposal distributions, hence they may perform rather poorly for complex and high-dimensional target densities. Another approach which is better suited for sampling from complex distributions is the *Markov Chain Monte Carlo* (MCMC) algorithm. MCMC based samplers define the target distribution to be the stationary distribution of a *Markov chain*, i.e.:

$$p(x^{\,(m)}) = \int_{x^{\,(m-1)}} p(x^{\,(m)} \,|\, x^{\,(m-1)})\, p(x^{\,(m-1)})\, \mathrm{d}x.$$

At each step i, by drawing correlated samples based on a proposal distribution $p(x^{\,(m)} \,|\, x^{\,(m-1)})$, MCMC samplers take more informed steps by avoiding a completely random walk of the target density space. Independent of the starting point $x^{\,(0)}$, after taking enough steps a Markov chain converges to the target distribution as its stationary distribution provided that $p(x^{\,(m)} \,|\, x^{\,(m-1)})$ is fully stochastic and non-zero over its entire domain, hence exhibiting for the chain the desired properties of *aperiodicity* and *irreducibility* respectively. To ensure convergence of a Markov chain to the target distribution, MCMC samplers are often designed to satisfy the sufficient but not necessary condition of reversibility or detailed balance:

$$p(x^{\,(m-1)} \,|\, x^{\,(m)})\, p(x^{\,(m)}) = p(x^{\,(m)} \,|\, x^{\,(m-1)})\, p(x^{\,(m-1)}).$$

Even though reversibility allows the chain to eventually converge to the desired distribution, starting from a random initial point it is hard to determine the number of steps needed till the convergence; MCMC samplers therefore discard a number of initial steps as *burn-in* samples. Although the samples drawn from a Markov chain are correlated, the ergodicity of the chain that follows from its aperiodicity and irreducibility, still allow for the samples to be used for computing (4.4). Most common examples of the MCMC algorithm are *Metropolis-Hastings*, *Gibbs sampling* and *Slice sampling*. We refer to the works of Neal (1993); Andrieu et al. (2003) for a detailed discussion on MCMC methods.

Combining Latent Preselection with Sampling

The select and sample framework fuses the two varied ideas of posterior approximation (4.1) and (4.4) to efficiently estimate expectations as:

$$\langle\, f(\vec{s})\,\rangle_n \;\approx\; \langle\, f(\vec{s})\,\rangle_{q_n} \;\approx\; \frac{1}{M}\sum_{m=1}^{M} f(\vec{s}^{\,(m)}), \tag{4.5}$$

where $\vec{s}^{\,(m)} \sim q_n(\vec{s};\Theta)$. The free parameters of the framework are: the number of H' latents to be selected, the number of samples to be drawn and the number of burn-in samples for an MCMC based sampler. Figure 4.1 (adapted from Shelton et al., 2011) illustrates the select and sample framework in contrast with other inference methods.

Spike-and-Slab Sparse Coding Model

The Gaussian spike-and-slab sparse coding that we study here assumes a Bernoulli prior over all H components of the the binary latent vector $\vec{b} \in \{0,1\}^H$, with a multivariate Gaussian prior for the continuous latent vector $\vec{z} \in \mathbb{R}^H$:

$$p(b_h \,|\, \Theta) = \mathcal{B}(b_h;\pi) = \pi^{b_h}\,(1-\pi)^{1-b_h} \tag{4.6}$$

$$p(z_h \,|\, \Theta) = \mathcal{N}(z_h;\, \mu_h, \psi_h^2), \tag{4.7}$$

where π defines the probability of b_h being equal to one and where μ_h and ψ_h^2 parameterize the Gaussian slab. A 'spike-and-slab' latent variable $\vec{s}$ is generated by point-wise multiplying $\vec{b}$ and $\vec{z}$, i.e., $s_h = (\vec{b} \odot \vec{z})_h = b_h\, z_h$. The resulting variable $\vec{s}$ has either continuous or exact zero entries. Given such a latent vector, a generative process is defined as:

$$p(y_d \,|\, \vec{s}, \Theta) = \mathcal{N}(y_d;\, W_{dh}\vec{s}_h, \sigma_d^2), \tag{4.8}$$

where $\vec{\sigma} \in \mathbb{R}^D$ is the observation noise. The columns of the matrix $W \in \mathbb{R}^{D\times H}$ are the latent components $\vec{W}^{(h)}$, each associated with a latent variable b_h. Together we use $\Theta =$

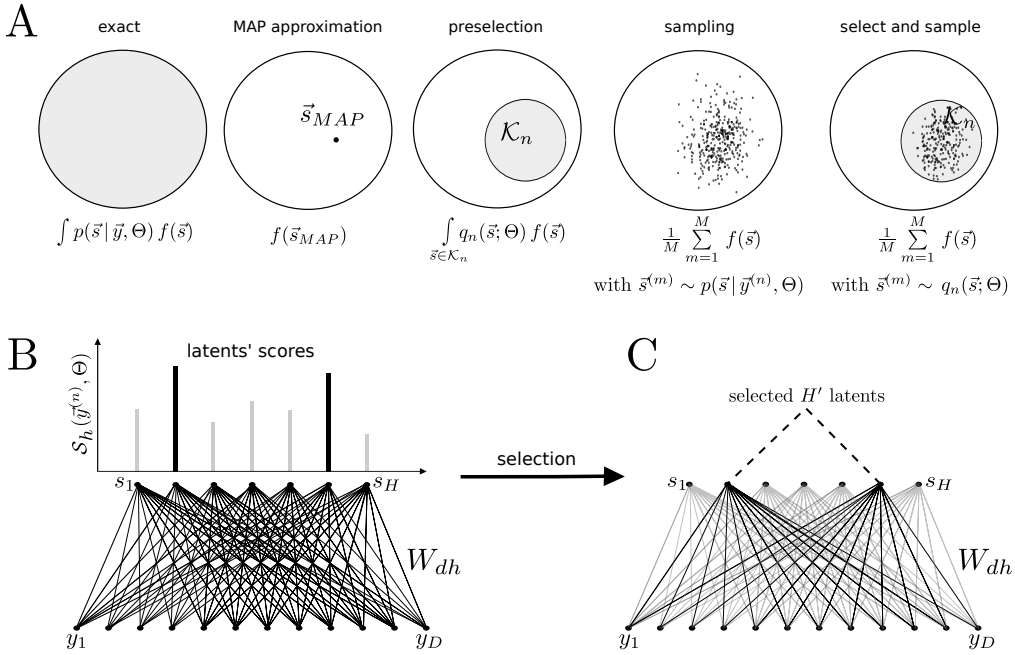

Figure 4.1: **A** A comparative illustration of different inference methods for estimating expectations with respect to the posterior over the latent space. **B** In a fully connected latent variable model, latent components $\vec{W}^{(h)}$ are scored by a given selection function $\mathcal{S}_h(\vec{y}^{(n)},\Theta)$. **C** The selection of H' highest scoring latent components reduces the posterior latent space to $\mathcal{K}_n$ (under 'preselection' in A). The select and sample framework further combines the preselection with sampling (the right-most illustration in A). The figure is adapted from Shelton et al. (2011).

$(W,\vec{\sigma},\vec{\pi},\vec{\mu},\vec{\psi})$ to mention all model parameters.

Parameter Optimization

To estimate the model parameters Θ given a set of N independent data points $\{\vec{y}^{(n)}\}_{n=1,\ldots,N}$, we apply expectation maximization (EM) algorithm as discussed in Chapter 1. The EM algorithm indirectly maximizes the data likelihood $\mathcal{L} = \prod_{n=1}^{N} p(\vec{y}^{(n)}\,|\,\Theta)$ by maximizing the so-called free energy, a lower bound of the log-likelihood, given by:

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

where $\langle\,\rangle_n$ denotes expectation under the posterior (4.3) and $H(\Theta^{\text{old}})$ is an entropy term only depending on parameter values held fixed during the optimization of $\mathcal{F}$ w.r.t. Θ. To recall, the EM algorithm iterates over two steps to optimize the free-energy: First, given the current parameters Θ^{old}, the relevant expectation values under the posterior are computed in the E-step; given these posterior expectations, $\mathcal{F}(\Theta^{\text{old}},\Theta)$ is maximized w.r.t. Θ in the M-step. Each iteration over the E- and M-steps locally maximizes the data likelihood.

The M-step optimization of free-energy entails updates of model parameters Θ, for which the update equations are derived by solving partial derivatives of the free-energy with respect to individual parameters. To that end, we expand the free-energy $\mathcal{F}(\Theta^{\text{old}}, \Theta)$ for the spike-and-slab model (4.6) to (4.8) as follows:

$$\begin{aligned}\mathcal{F}(\Theta^{\text{old}}, \Theta) \;=\; & \sum_{n=1}^{N} \sum_{\vec{b}} \int_{\vec{z}} p(\vec{b}, \vec{z} \,|\, \vec{y}^{\,(n)}, \Theta^{\text{old}}) \\ & \Big(-\tfrac{1}{2} \Big) \Bigg[\log(2\pi^D) + \sum_d \Big(2 \, \log(\sigma_d) + \tfrac{1}{\sigma_d^2} \Big(\vec{y}^{\,(n)} \, - \, W(\vec{b} \odot \vec{z}) \Big)_d^2 \Big) \\ & + \log(2\pi^{|\vec{b}|}) + \sum_h \Big(2 \, \log(\psi_h) \, b_h + \tfrac{1}{\psi_h^2} \big(\, (z_h \, - \, \mu_h) \, b_h \big)^2 - 2 \log(\pi_h^{b_h} \, (1 - \pi_h)^{1 - b_h}) \Big) \Bigg] \, \mathrm{d}\vec{z} \\ & + H(\Theta^{\text{old}}) \, . \end{aligned} \tag{4.9}$$

We then obtain the following parameter update equations by setting the partial derivatives of (4.9) with respect to the parameters to zero:

$$W = \frac{\sum_n \vec{y}^{\,(n)} \big\langle \vec{s} \big\rangle_n^{\mathrm{T}}}{\sum_n \big\langle \vec{s}\vec{s}^{\,\mathrm{T}} \big\rangle_n}, \tag{4.10}$$

$$\vec{\pi} = \frac{1}{N} \sum_n \big\langle \vec{b} \big\rangle_n \tag{4.11}$$

$$\vec{\mu} = \frac{\sum_n \big\langle \vec{s} \big\rangle_n}{\sum_n \big\langle \vec{b} \big\rangle_n}, \tag{4.12}$$

$$\psi_h^2 = \frac{\sum_n \big\langle (s_h - \mu_h b_h)^2 \big\rangle_n}{\sum_n \big\langle b_h \big\rangle_n}, \tag{4.13}$$

$$\text{and} \qquad \sigma_d^2 = \frac{1}{N} \sum_n \big\langle (\sum_h W_{dh} s_h - y_d^{(n)})^2 \big\rangle_n, \tag{4.14}$$

where $s_h = b_h \, z_h$ jointly denotes both the binary and the continuous parts of the spike-and-slab latent variable.

Preselection of Latents and Exact Gibbs Sampling

In order to efficiently estimate in the E-step the expected values $\langle \, . \, \rangle_n$ required for parameter updates (4.10) to (4.14), we take the approximate inference approach 'Select and Sample' (Shelton et al., 2011), computing the values as given by (4.5). The approximation scheme was later also applied to a non-linear spike-and-slab sparse coding model (Shelton et al., 2015). Following the assumption that the observed data is generated by a very sparse set of latent components, for a given data point $\vec{y}^{\,(n)}$ we first reduce its latent space by selecting $H' \ll H$ components that seem to be its most likely causes. This results in an approximation of the true posterior $p(\vec{s} \,|\, \vec{y}^{\,(n)}, \Theta)$ by a truncated distribution $q_n(\vec{s}; \Theta)$ described earlier as (4.1). To score the latent components for their preselection, we adapt

the selection function $\mathcal{S}_h(\vec{y}^{(n)}, \Theta)$ used in Chapter 3 for the spike-and-slab sparse coding model (4.6) to (4.8):

$$\mathcal{S}_h(\vec{y}^{(n)}, \Theta) \;=\; \sum_d \mathcal{N}(\vec{y}_d^{(n)};\, W_{dh}\mu_h, \sigma_d + W_{dh}^2/\psi_h) \propto p(\vec{y}^{(n)} \,|\, \vec{b} = \vec{b}_h, \Theta), \tag{4.15}$$

where $\vec{b}_h$ represents a singleton state with only component h being active.

After the preselection of latents, we do exact Gibbs sampling from the H'-dimensional posterior as the second step of the inference procedure. While the preselection step may not be needed for a small H, it becomes absolutely necessary to deal with the computational intractability faced in high dimensions. Previous works employing Gibbs sampling for spike-and-slab based models include (Mohamed et al., 2012; Zhou et al., 2009; Olshausen and Millman, 2000; Shelton et al., 2012), while Tan et al. (2010) for instance apply Gibbs sampling for efficient inference in sparse Bayesian modeling. A Gibbs sampler is a type of MCMC sampler, which sequentially samples each of the components of the variable of interest. For sampling a component the sampler constructs a posterior distribution that is conditioned on the current values of all the other components as well as any other given (i.e., observed) variables and model parameters. The value drawn given the conditional posterior distribution then replaces the current value of the component. A full Gibbs sample is returned after making a complete pass through all the components. It can be shown that by sequentially sampling each component given their conditional distributions, the samples returned by a Gibbs sampler are indeed drawn from the joint distribution of all the components.

To construct a Gibbs sampler for a-posteriori latent variable sampling in the spike-and-slab model (4.6) to (4.8), we can devise a latent variable Markov chain such that its target density is given by the following conditional posterior distribution:

$$\begin{aligned} p(s_h|\vec{s}_{H\setminus h}, \vec{y}, \theta) \;&\propto\; p(s_h|\theta) \prod_d p(y_d|s_h, \vec{s}_{H\setminus h}, \theta) \\ &= \Big((1-\pi)\,\tilde{\delta}(s_h) + \pi\,\mathcal{N}(s_h;\,\mu_h, \psi_h^2)\Big) \prod_d \mathcal{N}(s_h;\,\nu_d, \varphi_d^2)\,, \end{aligned} \tag{4.16}$$

where the Dirac delta $\tilde{\delta}(.)$ represents a spike at zero and where $\nu_d = (y_d - \sum_{h'\setminus h} W_{dh'}s_{h'})/W_{dh}$ and $\varphi_d^2 = \sigma_d^2/W_{dh}^2$. Here due to the special case of a Gaussian slab, we can apply standard Gaussian identities (also applied by Titsias and Lazaro-Gredilla (2011); Mohamed et al. (2012); Goodfellow et al. (2013) and in Chapters 2 and 3) to further simplify (4.16) as follows:

$$p(s_h|\vec{s}_{H\setminus h}, \vec{y}, \theta) \;\propto \Big((1-\pi)\,\mathcal{N}(s_h;\,\upsilon, \phi^2)\,\tilde{\delta}(s_h) + \pi\,\mathcal{N}(s_h;\tau, \omega^2)\Big)\,, \tag{4.17}$$

where $\upsilon = \phi^2 \sum_d \nu_d/\varphi_d^2$ and $\phi^2 = (\sum_d 1/\varphi_d^2)^{-1}$, whereas $\tau = \omega^2 (\upsilon/\phi^2 + \mu_h/\psi_h^2)$ and $\omega^2 = (1/\phi^2 + 1/\psi_h^2)^{-1}$. We can observe that the conditional posterior (4.17) of the latent variable s_h retains the form of a spike-and-slab distribution. We can therefore simply compute the cumulative distribution function (CDF) of (4.17) to simulate s_h from the exact conditional distribution ($s_h \sim p(s_h|\vec{s}_{H\setminus h}, \vec{y}, \theta)$) by means of inverse transform sampling. To draw a full sample, the sampler simulates in a random order each s_h such that $h \in I_n : |I_n| = H'$, while $\forall h \notin I_n$, $s_h = 0$.

With preselection, the computational cost of the sampler becomes independent of the total number of latent components H. It only scales linearly with H' and the number of observed dimensions D. Therefore the total cost of performing the E-step is $\mathcal{O}\big(N\,D\,H'\big)$.

Numerical Experiments

The implementation of parameter optimization scheme for the model was done within the framework proposed by Bornschein et al. (2010) for EM based optimization on a multi-core architecture. All the simulations in this section were therefore executed on arrays of CPU cores on large compute clusters. For the sampler, the number of samples to be drawn was primarily chosen to optimize the computational costs; however without significantly compromising the performance. For burn-in, we always discarded the initial 1/2 of the total number of drawn samples. The preselection parameter H' was also chosen to constraint the overall computational budget of the algorithm, but any prior knowledge about the expected sparsity of the latent causes was also taken into account to appropriately set the value of the parameter. In all the experiments, the initial values of π_h were randomly and uniformly chosen between 0.1 and 0.5. $\vec{\mu}$ was initialized with normally distributed random values, ψ_h was set to 1 and σ_d was initialized with the standard deviation of $\vec{y}_d$. The elements of W were i.i.d. drawn from a normal distribution with zero mean and a standard deviation of 5. For brevity, we will refer to our proposed select and sample spike-and-slab sparse coding as S5C in the following sections.

Method Verification

To verify our parameter optimization scheme, we first apply our method to small-scale synthetic data generated by the model. The small-scale of the experiment allows us to compute the exact data log-likelihood given the learned parameters (see Section 3.iii.2 in Chapter 3 for instance). The log-likelihood can then be traced against its ground-truth value, which can be computed from the known generative parameters used to generate the synthetic data. Such a comparison provides us a theoretically grounded measure for assessing the performance of S5C; the closer the gap between the log-likelihood values, the better our

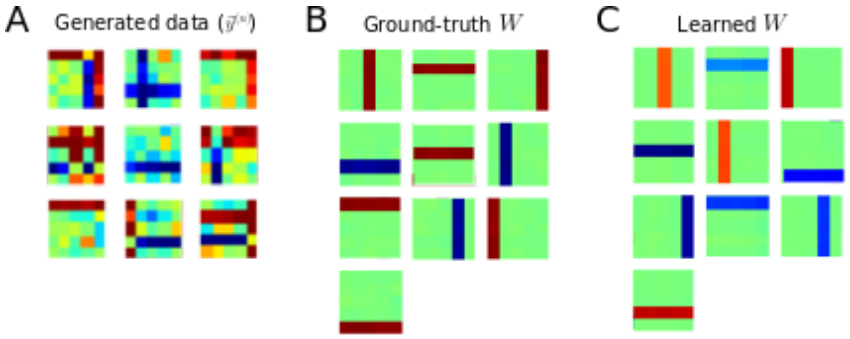

Figure 4.2: **A** Samples of observed data $\vec{y}^{(n)}$ generated by the spike-and-slab sparse coding model (4.6) to (4.8). **B** An example of ground-truth latent components $\vec{W}^{(h)}$ defined as vertical or horizontal bars with randomly assigned negative or positive values (here depicted in blue and red colors respectively). **C** An example of latent components learned by the select and sample spike-and-slab sparse coding (S5C) algorithm from a batch of observed data shown in **A**. The different colors of bars reflect variations in their inferred values, which is expected due to the multiplicative degeneracy between the latent components $\vec{W}^{(h)}$ and the latent mean values μ_h in the generative process (4.8).

method is at finding (globally) optimal model parameters to fit the observed data at hand.

For the verification of our method, we choose to perform the so called bars test (see e.g., Hoyer, 2002). The test defines the latent components $\vec{W}^{(h)}$ to be horizontal or vertical bars on a 2−dimensional grid. Figure 4.2B for instance defines latent components on a 5×5 pixel grid with 5 vertical and 5 horizontal bars. We apply our algorithm to $D = 5 \times 5$ bars data as shown in Figure 4.2A. To generate the data, we create $H = 10$ latent components as illustrated in Figure 4.2B. We then randomly make each of the 5 vertical and 5 horizontal bars positive or negative by assigning them a value of 5 or -5, while the non-bar pixels are assigned zero value. We generate the data points according to the generative model (4.6) to (4.8). To generate the data, the parameters of the latent slabs μ_h and ψ_h are set to 0.0 and 1.0 respectively. The values of π_h are set to $2.0/H$ such that there are on average two active latent components per generated observation. Finally the generated data is perturbed with the observation noise with $\sigma_d = 1.0$.

We run the bars test under three different settings of the preselection parameter H': once with $H' = H = 10$ (i.e., without preselection) and two further runs with $H' = 5$ and 4. For each setting we generate $N = 5000$ data points and perform 10 trials of the S5C algorithm with a random parameter initialization every trial. Including burn-in, we draw 40 samples per data point. We log the average log-likelihood value per EM iteration to track the convergence of the algorithm over different trials for each of the three different H' settings. Figure 4.3 shows the convergence of average log-likelihood over EM iterations for each trial of the S5C algorithm under different H' settings. Here we can observe that when the preselection is turned off (Figure 4.3A), the S5C algorithm shows rather slow and poor convergence behavior (dashed blue lines) towards the ground-truth (plotted as a

solid green line). We can see that in none of the 10 trials the likelihood given the learned parameters converge to the ground-truth value even after performing 150 EM iterations. Also in only 2 out of 10 runs, the 10 bar components are fully recovered. We observe a similar performance of the algorithm for the number of samples increased up to 50 and 60. However in comparison, we find that for $H' = 5$ and 4 (Figure 4.3B and C), the likelihood of the learned parameters converges to the ground-truth in 9 out of 10 trials within 50 EM iterations. Figure 4.2C shows an example of the components recovered by the S5C algorithm after 50 EM iterations for $H' = 5$. For both H' settings, we recover the 10 bars in 9 out of 10 trials. With increased number of samples for both $H' = 5$ and 4, we do not notice any significant change in the performance of the algorithm.

The results in Figure 4.3 suggest that by picking relevant components for posterior inference, preselection facilitates the sampler to directly navigate to the subspaces of dense posterior mass, while also dramatically reducing the computational complexity of the inference procedure. On the other hand, for a given observation the latent space without any preselection is likely to be too vast and cluttered with mostly irrelevant dimensions. This keeps the sampler from quickly converging to the volumes of concentrated posterior mass, consequently requiring the MCMC chain to run for (prohibitively) longer periods before it finds the desired regions of posterior space. Preselection has also been noticed previously for avoiding local optima in an expectation truncation based inference scheme for discrete latent spaces (Exarchakis et al., 2011), where in comparison to the exact inference, latent preselection led to a more accurate convergence of model parameters.

Image Denoising

Next we test the performance of S5C on the task of image denoising. Here we use the standard "house" benchmark for the task, which we have also used previously to evaluate spike-and-slab based (i.e., Zhou et al., 2009; Titsias and Lazaro-Gredilla, 2011) and other competing approaches (e.g., Li and Liu, 2009). The spike-and-slab approach by Zhou et al. (2009) employs a sampling-based optimization scheme. Titsias and Lazaro-Gredilla (2011) on the other hand apply variational inference for parameter optimization. As done in Section 4.3.1 and Chapters 2 and 3, Titsias and Lazaro-Gredilla (2011) also exploit the standard identities for the Gaussian slab to derive their method (the approach of Goodfellow et al. (2013) follows along the same lines. Moreover, Mohamed et al. (2012) resort to the same identities to derive a part of their inference procedure for their fully Bayesian Gaussian spike-and-slab sparse coding.).

To carry out the task of denoising, we first generate a noisy image by adding Gaussian noise to the house image. The standard deviation σ of the Gaussian function determines the level of added noise. Following the previous works, we consider three noise levels: $15, 25$

and 50. To generate input for the algorithm, we extract 8×8 patches from 256×256 noisy image, visiting a whole grid of 250×250 pixels by shifing (vertically and horizontally) 1 pixel at a time. In total we end up with $N = 62,001$ overlapping image patches. We then apply 100 iterations of the S5C algorithm with $H = 256$ to the input data. For smaller values of H (e.g., 64) we observe saturation in the performance and for larger values such as $H = 400$, we do not notice any significant boost in the results. For the experiments, the preselection parameter H' and the number of samples were kept at 40 and 100 respectively. The values were chosen through a gradual increment to optimize performance. Although in general we see a trend of improved performance for an increasing number of samples, we here make a cut off at the chosen value to maintain an adequate computational budget. We obtain best results at the task by assuming $\vec{\pi}$ to be isotropic so that $\pi_h = \frac{1}{NH} \sum_n \sum_h \langle b_h \rangle_n$.

We compare the results of the experiments against various competitive approaches in Table 4.1. The performance is measured in peak signal-to-noise ration (PSNR), which is computed between the original noise-free image and the denoised output from an algorithm. We find that S5C outperforms all the other methods for $\sigma = 25$. Figure 4.4 shows the noisy image and the denoised output of the S5C algorithm for $\sigma = 25$. For noise levels 15 and 50, the PSNR of S5C is competitive with other approaches, where for $\sigma = 50$, the expectation truncation based Gaussian sparse coding (GSC-ET) from Chapter 3 holds the benchmark.

	PSNR (dB)						
Noise	Noisy img	MTMKL$^{exp.}$	K-SVD$^{mis.}$	*K-SVD^{match}	Beta pr.	GSC-ET	S5C
σ=15	24.59	**34.29**	30.67	34.22	34.19	33.78	33.50
σ=25	20.22	31.88	31.52	32.08	31.89	32.01	**32.08**
σ=50	14.59	28.08	19.60	27.07	27.85	**28.48**	28.35

Table 4.1: Comparison of the S5C 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), Beta process (Zhou et al., 2009) and GSC-ET (Chapter 3). 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).

Large-scale Run on Natural Image Patches

To demonstrate the scalability of our method, we apply our sparse coding model to natural image patches following the seminal work of Olshausen and Field (1997), who established sparse coding as a standard model for the response properties of V1 simple cells. In our experiment we use 16×16 image patches cut-out from the Van Hateren natural image database (van Hateren and van der Schaaf, 1998). Following Olshausen and Field (1996), we apply pseudo-whitening to preprocess the patches. We perform 50 EM iterations of the S5C algorithm on $N = 10^6$ image patches to extract $H = 10000$ latent components. We

set $H' = 20$ and we draw 50 samples per patch. As described in the previous section, we assume the sparseness vector $\vec{\pi}$ to be isotropic.

In Figure 4.5 we show 400 of the 10000 latent components $\vec{W}^{(h)}$ learned by the S5C algorithm. Most of the components resemble Gabor-Wavelet and Difference of Gaussians like functions. The components are ordered (left to right, top to bottom) with respect to their posterior probabilities. Other than that we observe the learned sparsity π_h to be 6.2×10^{-4}, which implies that on average a little over 6 components get activated to encode a patch. The sparsity value falls in agreement with that inferred on the same dataset (although for different H and N) by the nonlinear spike-and-slab sparse coding model we study in Chapter 5. Other parameter values such as observation noise, etc. are however not directly comparable due to the input normalization used in the other study.

With $H = 10000$, we learn (to the best of our knowledge) the highest number of latent components using a spike-and-slab sparse coding approach. Previously Goodfellow et al. (2013) have reported to learn 8000 from 2×10^6 image patches of $D = 32 \times 32 \times 3$ pixels. In comparison we use half as many patches of 16×16 pixels. To our knowledge, the only other spike-and-slab methods that use image patches of up to 16×16 pixels are (Garrigues and Olshausen, 2007) and the nonlinear spike-and-slab sparse coding we study in Chapter 5.

Figure 4.6 (adapted from Goodfellow et al., 2013) compares different spike-and-slab based and other scalable sparse latent variable models against scales at which they have been demonstrated to work. We see that while being the only exact method in the plot (and also otherwise to the best of our knowledge) for a spike-and-slab sparse coding model, the Gaussian sparse coding (GSC) that we propose in Chapter 2 lies furthermost towards the lower left corner; however, the cost of the method scales linearly with N and therefore in conjunction with the parallel implementation (see Appendix 3.I; Chapter 3), it can be straightforwardly applied to large datasets. Two other spike-and-slab methods that we have in the plot are labelled as GSC-ET and SSMCA. We introduce the expectation truncation (Lücke and Eggert, 2010) based Gaussian sparse coding (GSC-ET) in Chapter 3, which is a scaled-up extension of GSC. SSMCA on the other hand is the only nonlinear spike-and-slab sparse coding model in the comparison, which we will introduce in Chapter 5. The method also employs the select and sample framework (Shelton et al., 2011) for approximate learning and inference. Except for Goodfellow et al. (2013), we can observe that both GSC-ET and computationally more demanding SSMCA already lie ahead of all the the other spike-and-slab approaches. In fact, in supplementary sections in Chapter3 (Appendix 3.H and 3.I), the GSC-ET algorithm has been demonstrated to use even up to $N = 320K$ data points (Figure 3.11) and scale to more than a thousand latent components in a separate experiment (Figure 3.10). Finally we have the S5C algorithm, which we have applied at a very large scale. This puts S5C in Figure 4.6 among the most scalable sparse

latent variable methods. The key factor that makes it possible to apply S5C on such a large scale is the latent preselection, which reduces an otherwise computationally infeasible inference task to an amenable scale, making it possible for the sampler to effectively approximate the posterior within small subspaces of latent dimensions that are considered to be relevant with respect to an observation.

Discussion

In this chapter we have introduced an approximate inference method for linear spike-and-slab sparse coding. We develop the method following the select and sample framework of Shelton et al. (2011), who combine sampling with latent preselection as a model for neural inference – applying it to sparse coding with binary latents. We apply the framework to spike-and-slab sparse coding, where by assuming the slabs to be Gaussian distributed we derive an exact Gibbs sampler for a-posteriori simulation of (preselected) latent variables. Through experiments involving ground-truth values we show that by reducing its domain to a small number of latent dimensions, preselection can enable the sampler to quickly discover dense volumes of posterior probability, while being very effective at avoiding local optima and unreasonably long burn-in periods. The sampler on the other hand exhibits a suboptimal convergence performance without any preselection. While preselection crucially relies on the selection function used for scoring the latent components, our choice of the selection function is adapted from the one used in Chapter 3, where it is reliably tested and employed for a similar model.[1] Therefore for our approach, the performance of the selection function can be corroborated through the empirical results we obtain e.g., in Sections 4.4.1 and 4.4.2.

Our approach inherits three approximation parameters, the number of latents to be preselected H', the number of samples to be drawn and the number of burn-in samples. Although we propose no grounded means for picking the best values of these parameters, in practice we have observed that as long as the input data can be considered to be generated by sparse latent activities, keeping H' under 50 (or considerably low with respect to H) turns out to be sufficient for achieving good results. Moreover, computational constraints, performance objectives or prior knowledge about the data all have an influence on more concretely determining the value of the parameter. With respect to the sampler, while maintaining a computational budget for a chosen value of H', we always found it beneficial to draw an increasing number of samples. We simply kept the burn-in proportion to 1/2 the number of drawn samples, which can be further optimized for a specific task and performance metric.

In the experiments on sythetic bars data we have observed a compelling benefit of com-

[1] In Chapter 3, the observation noise and standard deviation of the latent slabs are assumed to be full-rank matrices, but otherwise the model is the same.

bining sampling with latent preselection. The select and sample inference approach allowed the S5C algorithm to converge to ground-truth data log-likelihood and recover true latent components in relatively fewer EM iterations, while also being computationally more efficient per E-step. The purely sampling based inference on the other hand showed poor log-likelihood convergence and component recovery performance, albeit consuming significantly more computational resources. We have further seen very competitive performance of the S5C algorithm on the task of denoising using a real benchmark, including S5C achieving the state-of-the-art performance for noise level $\sigma = 25$. Lastly through the run on natural image patches we validate that our inference approach can be applied on very large scales, which puts it on par with some of the most scalable works on sparse latent variable models (i.e., Coates and Ng, 2011; Courville et al., 2011; Goodfellow et al., 2013).

To our knowledge, our proposed method is the only sampling based inference approach for spike-and-slab sparse coding that has been demonstrated to scale up to very large dimensions. As a future work it would be interesting to further expand the application domain of the S5C algorithm to high-level tasks: for instance employing the method as either a simple feed-forward (see Coates and Ng, 2011, for instance) or a more thorough, probabilistic feature encoder in a classification hierarchy (e.g., Coates and Ng, 2011; Courville et al., 2011; Goodfellow et al., 2013).

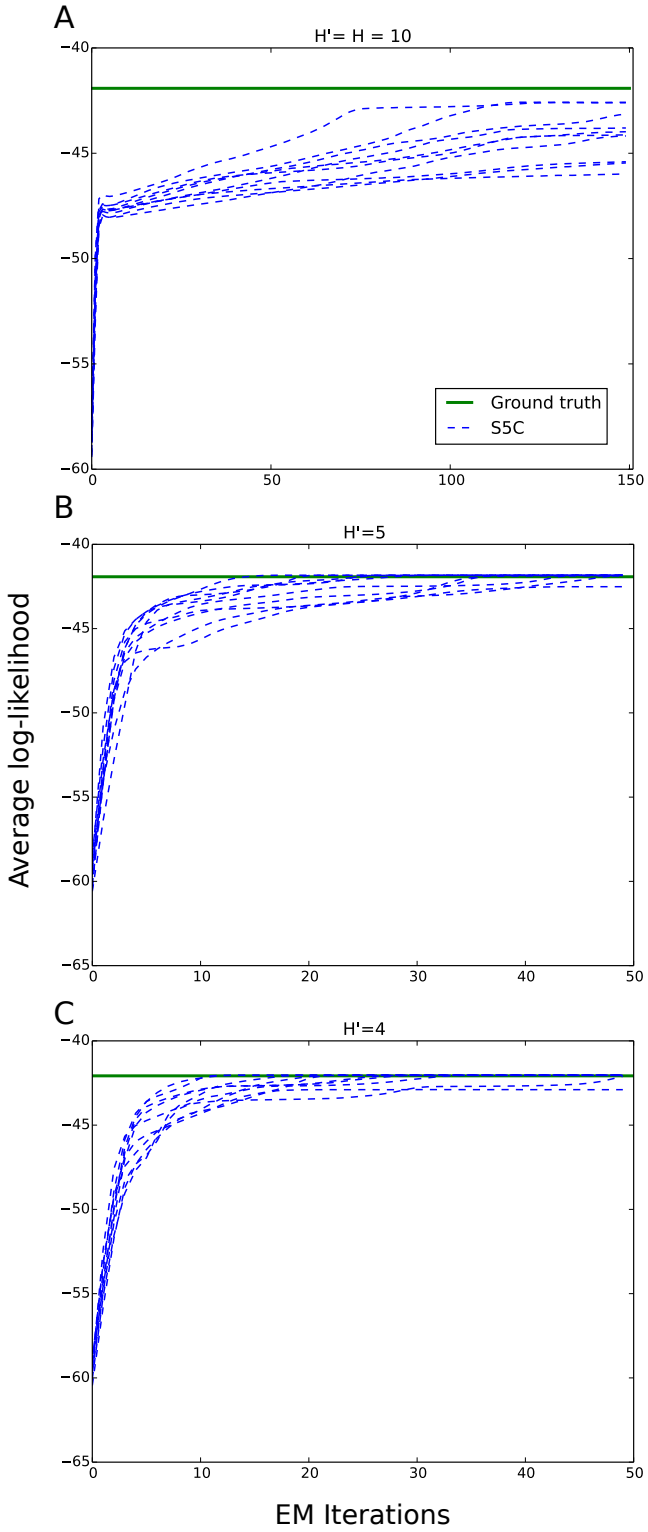

Figure 4.3: Convergence of the average log-likelihood (dashed blue lines) for each of the 10 independent trials given three different settings of the preselection parameter, i.e.: **A** $H' = H = 10$, **B** $H' = 5$ and **C** $H' = 4$. The solid green lines mark the ground-truth values of the log-likelihood, which is computed provided the values of generative parameters.

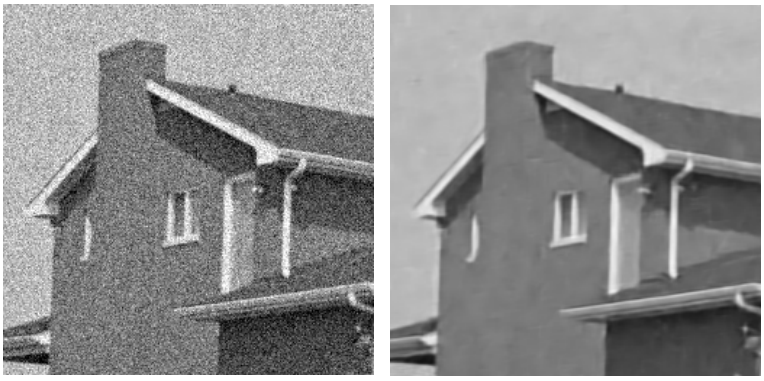

Figure 4.4: Left: Noisy "house" image with $\sigma = 25$. Right: Denoised output of the S5C algorithm.

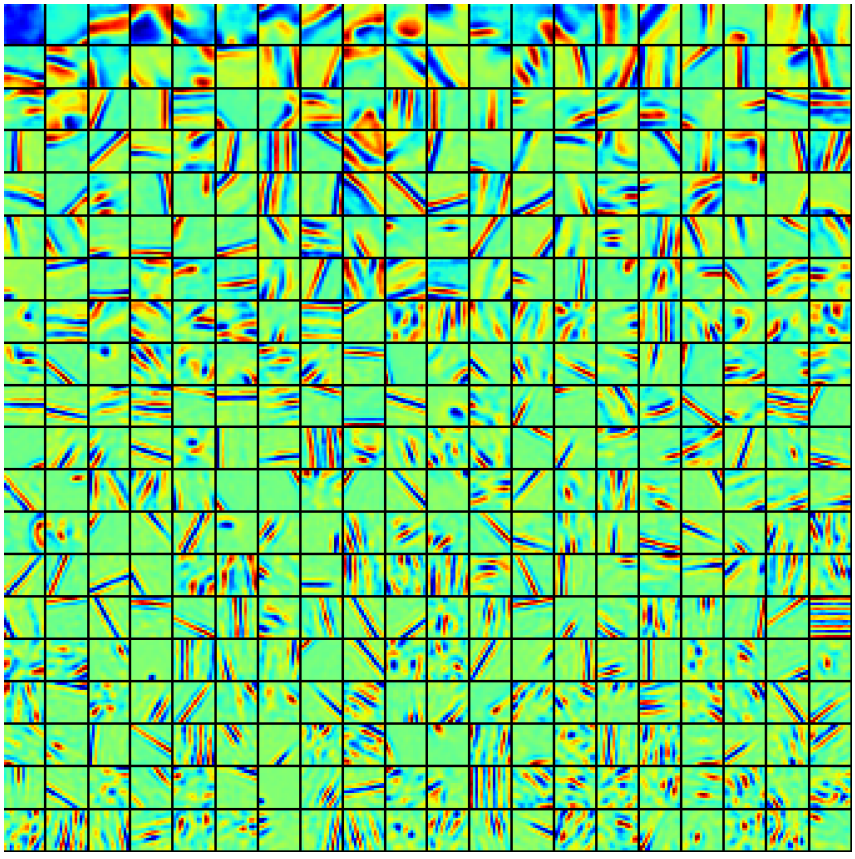

Figure 4.5: Latent components $\vec{W}^{(h)}$ learned by the S5C algorithm. The components are sorted from left to right and top to bottom with respect to their posterior probabilities. Out of $H = 10000$ learned components, we pick every 4^{th} component for display, showing here a total of 400 components.

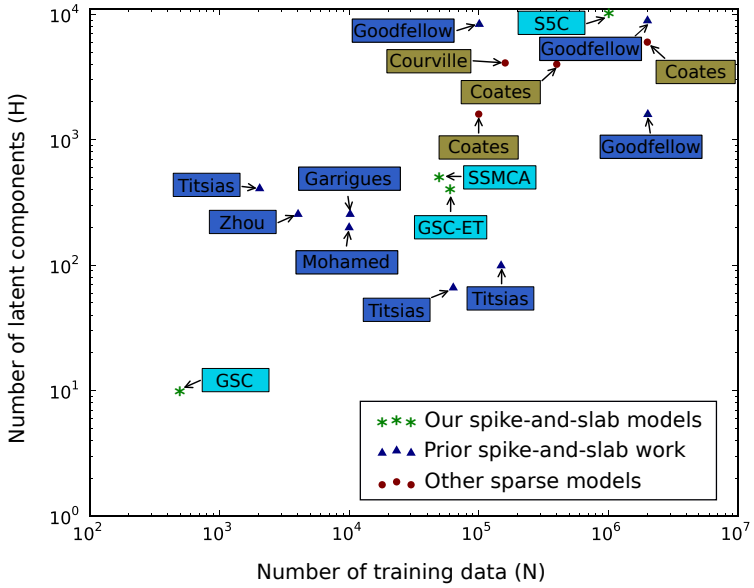

Figure 4.6: Comparison of different methods with respect to their demonstrated scale of application. On x-axis we have the number of training data used by a method to learn number of latent components shown against the y-axis. The methods compared include: the spike-and-slab approaches of Garrigues and Olshausen (2007); Zhou et al. (2009); Titsias and Lazaro-Gredilla (2011); Goodfellow et al. (2013), the work on sparse latent variable models by Coates and Ng (2011); Courville et al. (2011), and our spike-and-slab models: GSC (Chapter 2), GSC-ET (Chapter 3), SSMCA (Chapter 5) and S5C. The figure is adapted from Goodfellow et al. (2013).

Chapter 5

Spike-and-Slab Maximal Causes Analysis

In this chapter we turn our attention to nonlinear sparse coding. Here we recruit the Gaussian spike-and-slab prior for a sparse coding model that assumes a point-wise maximum combination rule for the latents to generate an observation. The maximal causes analysis (MCA) model was originally introduced with a Bernoulli prior by Lücke and Sahani (2008). In comparison to linear sparse coding (i.e., Olshausen and Field, 1996), the model has since been shown to be a more accurate alternative for approximating low-level occlusions in images (Lücke and Sahani, 2008; Puertas et al., 2010; Bornschein et al., 2013).

The spike-and-slab prior allows MCA to encode both the absence or presence and the intensity of a latent component, with which it contributes to the observation generation process. While being more flexible, the spike-and-slab MCA (SSMCA) model does not offer a closed-form solution for posterior computation. We therefore develop a Gibbs sampler for approximate posterior inference in the model. Following Shelton et al. (2011), we further combine sampling with latent preselection (Lücke and Eggert, 2010) to devise a computationally efficient inference scheme.

We perform data-driven encoding experiments on both synthetic and real image data. We find latent components inferred by the SSMCA algorithm to be less localized in nature, which allows for the encoding results to be naturally sparse and well-aligned e.g., with the known ground-truth for synthetic data. In contrast, linear sparse coding (the implementation of Mairal et al., 2009b) finds encoding schemes that can better minimize the mean reconstruction error (MSE) of training data; however, the obtained solutions tend to be fairly non-sparse with inferred components appearing to be less descriptive of the data. We further observe that on a restricted budget for latent activity, the MSE of the linear approach deteriorates significantly, beating the average MSE of SSMCA when the sparsity

level of the linear approach is tuned to be on par with the inferred sparseness of SSMCA. Lastly in congruence with *in vivo* neural recordings, we also qualitatively assess receptive field estimates of SSMCA inferred components from natural image data.

Nonlinear spike-and-slab sparse coding for interpretable image encoding

Jacquelyn A. Shelton[1,*], Abdul-Saboor Sheikh[1], Jörg Bornschein[2] Philip Sterne[3], Jörg Lücke[1,4]

[1]Department of Software Engineering and Theoretical Computer Science
Technical University Berlin
Berlin, Germany
[*]shelton@tu-berlin.de

[2]Department of Computer Science and Operations Research
University of Montreal
Montreal, Quebec, Canada

[3]Frankfurt Institute for Advanced Studies
Goethe-University Frankfurt
Frankfurt, Germany

[4]School of Medicine and Health Sciences and Cluster of Excellence Hearing4all
University of Oldenburg
Oldenburg, Germany

Published in PLoS ONE 10(5): e0124088, 2015.

Abstract: Sparse coding is a popular approach to model natural images but has faced two main challenges: modelling low-level image components (such as edge-like structures and their occlusions) and modelling varying pixel intensities. Traditionally, images are modelled as a sparse linear superposition of dictionary elements, where the probabilistic view of this problem is that the coefficients follow a Laplace or Cauchy prior distribution. We propose a novel model that instead uses a *spike-and-slab prior* and *nonlinear combination of components*. With the prior, our model can easily represent exact zeros for e.g. the absence of an image component, such as an edge, and a distribution over non-zero pixel intensities. With the nonlineary (the nonlinear max combination rule), the idea is to target occlusions; dictionary elements correspond to image components that can occlude each other. There are major consequences of the model assumptions made by both (non)linear approaches, thus the main goal of this paper is to isolate and highlight differences between them. Parameter optimization is analytically and computationally intractable in our model, thus as a main contribution we design an exact Gibbs sampler for efficient inference which we can apply to higher dimensional data using latent variable preselection. Results on natural and artificial occlusion-rich data with controlled forms of sparse structure show that our model can extract a sparse set of edge-like components that closely match the generating process, which we refer to as *interpretable* components. Furthermore, the sparseness of the solution closely follows the ground-truth number of components/edges in the images. The linear model did not learn such edge-like components with any

level of sparsity. This suggests that our model can adaptively well-approximate and characterize the meaningful generation process.

Introduction

Many natural signals, such as visual data, exist in a high-dimensional space. Understanding the structure of visual data is a challenging task that is often approached by forming parametric models of the data following some principles of optimality, in order to learn something about the data's content and composition. As many signals have a low intrinsic dimensionality, in this paper we focus on the domain of *sparse coding models* to address the task of image modelling. The basic idea behind the sparsity principle is to represent a signal – such as an image – as a combination of few basis functions or features. With roots in signal processing, it is often thought that a model assuming or enforcing sparsity can recover the intrinsic signal dimensions and therefore better represent the relevant information content in the signal (e.g., Mallat, 2008; Kutyniok, 2012). Furthermore, one would expect that if the algorithm learns meaningful hidden structure of the signal, then this approach would be successful at many data-driven tasks. When an algorithm can extract and represent the relevant information content from a signal that not only follows the generating process of that data but can also be easily interpreted in the context of the task at hand, we refer to this as *interpretable* data encoding.

Following early physiological recording studies (Hubel and Wiesel, 1959) on simple cells in the visual cortex, sparse coding became popular as a model of the visual data encoding process in the mammalian primary visual cortex (Olshausen and Field, 1996) and has now become not only the standard model to describe coding in simple cells, but also a very popular feature learning algorithm (e.g., Goodfellow et al., 2013; Lee et al., 2007). Formally, sparse coding (which will be referred to as 'SC') assumes that each image (also called an 'observation', or observed variables) $\vec{y} = (y_1, \ldots, y_D)^T$ is associated with a sparse vector of latent variables $\vec{s} = (s_1, \ldots, s_H)^T$ (also called latent 'causes' or coefficients of the data), where D and H denote the dimensionality of the observed image and the latent variable space, respectively. In the setting of visual data, the sparse latent vector $\vec{s}$ describes the set of the possible causes of an observed image and is associated with a set of image components, or *dictionary elements*, $W \in \mathbb{R}^{D\times H}$ (low-level image components, e.g. edge-like structures) where the absence of such an image component is associated with $s_h = 0$. In this way, sparsity means that most of the coefficients s_h in $\vec{s}$ are zero or close to zero.

The *standard linear sparse coding problem* is formulated as follows:

$$loss(\vec{y}^{(n)}, W) \;:=\; \min_{\vec{s}} \; \frac{1}{2}||\vec{y}^{(n)} - W\vec{s}||_2^2 \;+\; a||\vec{s}||_1, \tag{5.1}$$

with the objective to minimize the loss between the image $\vec{y}^{(n)}$ and its *linear* reconstruction/estimation $W\vec{s}$ (or equivalently $\sum_h s_h \vec{W}^{(h)}$ where W is the $D \times H$ matrix of $\vec{W}^{(h)}$ dictionary elements/components), with a penalty on the l_1-norm of the vector $\vec{s}$. The penalty is controlled by a regularization parameter a, which dictates how sparse the coefficients $\vec{s}$ in the reconstruction of $\vec{y}$ will be. Objective (5.1) and associated optimization algorithms are also often referred to as *basis pursuit* (Chen et al., 1998) or the *Lasso* (Tibshirani, 1996).

Probabilistically, linear SC can be formulated as a *generative model*:

$$p(\vec{y}\,|\,\Theta) = \int_{\vec{s}} p(\vec{y}\,|\,\vec{s},\Theta)\,p(\vec{s}\,|\,\Theta)\,d\vec{s}, \tag{5.2}$$

where the latent causes are characterized by $p(\vec{s}\,|\,\Theta)$ with a sparse prior distribution. The observation/image described by $p(\vec{y}\,|\,\vec{s},\,\Theta)$ is typically a Gaussian distribution with a mean $\vec{\mu} = \sum_h s_h\,\vec{W}^{(h)}$, i.e. centered at the linear superposition of components $\vec{W}^{(h)} \in \mathbb{R}^D$. If the Laplace distribution is used as prior distribution, it can be shown that the minimization of objective (5.1) with respect to the dictionary elements corresponds to expectation maximization (EM) learning using the maximum a-posteriori (MAP) approximation for the posterior (e.g., Murphy, 2012). For dictionary learning, the formulation of objective (5.1) is often the method of choice, and the focus is on efficient optimization of the dictionary. With these approaches, no prior parameters can be learned directly and the sparsity penalty, therefore, has to be set by hand or it has to be determined by cross-validation in another optimization loop. Furthermore, MAP estimates of the posterior can lead to a relatively coarse approximation, which has motivated improved probabilistic approaches for the standard model (Opper and Winther, 2005; Seeger, 2008).

The focus of this work is to investigate a new sparse coding model that forms a more realistical image model than the standard linear model with Laplace prior. After motivating and defining the model, we will systematically evaluate the differences to standard sparse coding. The problem setting we focus on is illustrated with the toy example in Figure 5.1. One can see that visual components (such as edges) are either present or absent (i.e. coefficient $s_h = 0$) in an image. This however points to the first challenge that standard models for sparse coding face: standard models using a Laplace or Cauchy prior distribution, which do not intrinsically represent exact zeros, can only either yield coefficients with exact zeros as an artifact of the optimization that artificially enforcing the coefficents to be zero (see e.g., Seeger, 2008; Lee et al., 2007, for examples). These distributions are referred to as "weakly sparse", as they have no coefficients actually at zero, but many very close to zero (Mohamed et al., 2012). Other models, with use of a binary prior distribution, can represent exact zeros (to model e.g. the absence of a visual component with $s_h = 0$) without need for optimization techniques to induce them. These models cannot however model the range of intensities that the image components may manifest (e.g. when the component is present, it is represented by $s_h = 1$). An alternative and recently very popular prior is the spike-and-slab distribution (Titsias and Lazaro-Gredilla, 2011; Mohamed et al., 2012; Goodfellow et al., 2012; Sheikh et al., 2014, e.g.,), which is a distribution consisting of a discrete binary part and a continuous Gaussian part (see the first column in Figure 5.2 for an illustration of the spike-and-slab and Laplace priors). This prior can model not only the absence/presence of a component (via the binary 'spike') but also the visual intensity of that component (via the 'slab'). Second, the standard model assumes that visual components linearly superimpose to form an image, although objects do not actually elicit summed intensity values when they happen to occlude each other. In this setting, when evaluating the pixel intensities of two overlapping components, the standard linear model would sum the two pixels, which poorly estimates the intensity, whereas the max infers that the pixel with the maximal intensity is occluding the other, offering a better estimate,

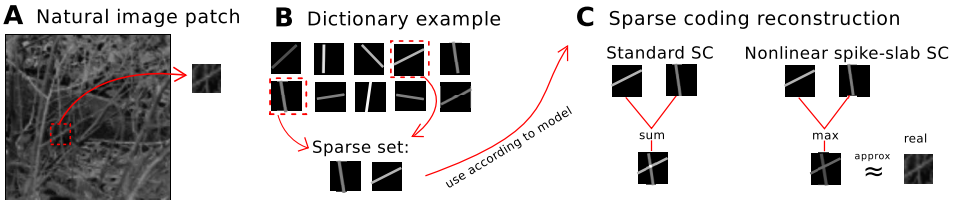

Figure 5.1: **Toy example illustrating the problem setting: approximating occlusions in images.** Given an image patch with occlusions (**A**), assume both the linear and nonlinear sparse coding models were given the true generating dictionary elements (**B**) and the task is for each model to use a sparse set of these to generate a reconstruction of the patch (**C**). **A** Example natural image with one patch to be reconstructed by the models. **B** 10 ground-truth dictionary elements, assumed to be known and with only 2 of 10 having generated the image patch. **C** Image reconstruction using the sparse dictionary set of the 2 models: the standard linear sparse coding model and the nonlinear spike-and-slab SC model. The linear sum leads to inaccurate pixel estimates when components overlap, whereas the nonlinear max aims to approximate this type of data more realistically in this scenario. Furthermore, the spike-and-slab prior (shown here for the the nonlinear model) allows the model to adapt the intensity of each image component to match what it observed in the data.

illustrated in Figure 5.1**C**. Despite these two modelling caveats, the most work on SC models focuses on efficient inference of the optimal parameters for the linear model (e.g., Seeger, 2008; Lee et al., 2007) and not in assessing the model assumptions themselves. The standard model form offers mathematical convenience for inference, namely allowing the use of convex approaches (i.e. the posteriors over latent variables have only one mode, allowing for efficiency/accuracy of maximum a posteriori (MAP) estimations). Consequently, the standard model has continued to use a Laplace prior with a linear superposition, because changing the prior or changing the superposition assumption induces complex and multimodel posteriors and correspondingly poses a challenge for MAP estimates due to many locally optimal solutions. As a result, each proposed modification of the standard model has so far only been investigated in turn.

This work proposes a novel sparse coding model that combines both of these improvements – a *spike-and-slab distribution* and *nonlinear max combination of components* – in order to form a more realistic model of images. For our main technical contribution, we optimize our model by using a combined approximate inference approach with preselection of latent variables (for truncated approximate EM Lücke and Eggert, 2010) in combination with Gibbs sampling (Shelton et al., 2011). Importantly, as we expect to see the most salient differences between the models when occlusions are present, several sets of experiments focus on natural and artificial occlusion-rich datasets where we consider the task of dictionary learning and image reconstruction.

In our experiments we show that we can efficiently train this nonlinear model and perform inference assuming a reasonably high number of observed and latent variables. First, we show on artificial data that the method efficiently and accurately infers all model parameters, including data noise and sparsity. Next, we compare our nonlinear model to a state-of-the-art linear model on occlusion-rich datasets for the task of dictionary learning and image reconstruction on both artificial data with controlled forms of sparse structure as well as natural data.

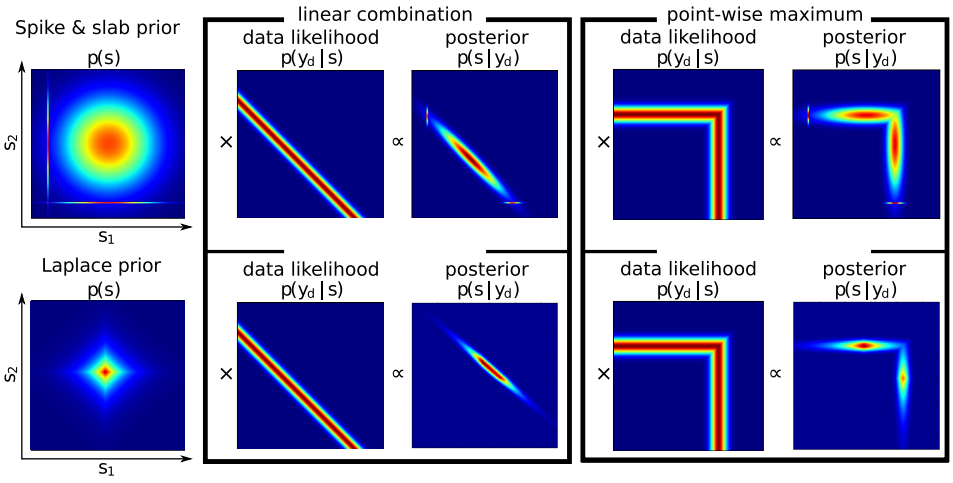

Figure 5.2: **Illustration of choice of prior distribution and multimodality in the latent space.** A H=2-dimensional spike-and-slab and Laplace priors over latent variables and the multimodal posterior distribution induced by these priors for both linear and nonlinear data likelihoods.

With experiments comparing the reconstruction of images by the two models, we show that the nonlinear model extracts/uses a sparse set of interpretable, holistic components that match the generating process, whereas the linear model (at all sparsity levels) uses components which are difficult to interpret and not aligned with the generating process. Finally, with experiments on image patches, we show that our model is consistent with *in vivo* neural recordings and learns image components with which linear models have struggled (Ringach, 2002; Bornschein et al., 2013). With these data we also show that our model is consistent in the sense that the average posterior over the latent variables is approximately equal to the prior.

The paper is organized as follows: first, the proposed model will be presented, second, the details of the inference method will be described, third, all experimental results will be presented, and finally, the results will be discussed.

Model: Nonlinear Spike-and-Slab Sparse Coding

We formulate the data generation process as the probabilistic generative model:

$$p(y_d \,|\, \vec{s}, \Theta) = \mathcal{N}(y_d;\, \max_h\{s_h W_{dh}\},\, \sigma^2). \tag{5.3}$$

Here, in contrast to the standard linear formulation in (5.2), the likelihood contains the nonlinear term $\max_h\{s_h W_{dh}\}$ instead of the linear $\sum_h s_h \vec{W}^{(h)}$ (the $\max_h$ which considers all H latent components and takes the h yielding the maximum value for $s_h W_{dh}$). Also, the latent variable s_h is drawn from a spike-and-slab distribution given by $s_h = b_h z_h$, where b_h is drawn from a Bernoulli distribution and z_h is drawn from a Gaussian distribution ($\mathcal{B}$ and $\mathcal{N}$, respectively), and is parameterized by:

$$p(b_h \,|\, \Theta) = \mathcal{B}(b_h; \pi) = \pi^{b_h} (1-\pi)^{1-b_h} \tag{5.4}$$

$$p(z_h \,|\, \Theta) = \mathcal{N}(z_h;\, \mu_{\mathrm{pr}}, \sigma^2_{\mathrm{pr}}), \tag{5.5}$$

The columns of the matrix $W = (W_{dh})$ are the dictionary elements/generative fields, $(\vec{W}^{(h)})_{h=1}^{H}$, with one $\vec{W}^{(h)}$ associated with each latent variable s_h. We denote the set of all parameters with $\Theta = (\pi, \mu_{\text{pr}}, \sigma_{\text{pr}}, W, \sigma)$.

For inference and in order to optimize the parameters Θ of this model, we are interested in working with the posterior over the latent variables given by

$$p(\vec{s}|\vec{y},\theta) = \frac{p(\vec{y}|\vec{s},\theta)\ p(\vec{s}|\theta)}{\int_{s'} p(\vec{y}|\vec{s'},\theta)\ p(\vec{s'}|\theta)\, d\vec{s'}}. \tag{5.6}$$

Identical to the standard sparse coding formulation in Equations (5.1) and (5.2), our model assumes independent latent variables and Gaussian-distributed observations given the latent variables. In contrast to the standard formulation, the latents are not distributed according to a Laplace prior and the components (i.e. coefficients, dictionary elements, or generative fields) are not combined linearly. Figure 5.1 contains a toy illustration of part of the generative process and model differences between standard linear model and nonlinear model. Figure 5.1**A** shows an example natural image, eliciting naturally occuring occlusions of branches and twigs, from which a patch has been extracted in order to illustrate the effects of each model's (non)linearity assumption. Figure 5.1**B** shows examples of how corresponding generating dictionary elements could look. For the sake of simplicity, this example does not incoorporate the learning process, and assumes each model is simply given these components and instructed which sparse set of components in 5.1**B** generated the image patch in 5.1**A**. Figure 5.1**C** shows how the (non)linear assumptions of the models manifest when the given components from 5.1**B** are combined according to each model in order to reconstruct the patch in 5.1**A**. As can be seen for the sum operation in 5.1**C**, standard linear sparse coding results in strong interference when the dictionary elements overlap, whereas the max can reconstruct the patch using one element or the other when the two overlap, thereby minimizing interference. This effect however leads to correlated multimodal posteriors since each observed pixel y_d must be explained by either one cause or the other, instead of the sum of both. An illustration of the posteriors of these models will be provided in the following section. This example suggests that the max can better model the occluding components (e.g., Lücke and Sahani, 2008; Puertas et al., 2010; Bornschein et al., 2013). Furthermore, for simplification in this example, we implicitly forced all other dictionary elements in 5.1**B** to be unused, i.e. associated with coefficients of $s_h = 0$, which is only possible with a spike-and-slab prior (or other binary prior, which in turn, would not be able to incoorporate the various gray value intensities of the dictionary elements). Additionally, with the spike-and-slab prior (shown here for the nonlinear model in Figure 5.1**C**) allows the model to adapt the intensity of each image component used to match what it observed in the data. Please see Shelton et al. (2012) for a preliminary discussion of this model in a conference submission, in which a thorough analysis of the model was not provided and additionally it contained an error in the computation of expected values, which has been corrected here.

Related Work

While work on improved optimization approaches for the standard sparse coding continues and is important for many applications, the above discussed limitations

of the underlying generative data model have motivated a number of related studies on improved models. In recent years, spike-and-slab priors for linear models have frequently been used. The resulting challenges for parameter optimization have been addressed by applying factorized variational EM (Goodfellow et al., 2013; Titsias and Lazaro-Gredilla, 2011), truncated EM (Sheikh et al., 2014) or sampling (Mohamed et al., 2012). Furthermore, the use of spike-and-slab priors aligns well with the goals of compressed sensing approaches (Donoho, 2006). In a standard formulation, an observed variable is re-expressed as a sum of bases where the corresponding coefficients have hard zeros, and correspondingly the objective function includes an $||\cdot||_0$-norm instead of the $||\cdot||_1$-norm seen in standard sparse coding (see e.g., Kutyniok, 2012, for a review).

Similarly, inference and learning for sparse coding models that replace the linear combination by nonlinear ones have been investigated. Hidden causes models with nonlinearly interacting signal sources include the noisy-or combination rule (Dayan and Zemel, 1995; Saund, 1995; Singliar and Hauskrecht, 2006; Wood et al., 2006; Jernite et al., 2013; Frolov et al., 2014), exclusive causes (Dai et al., 2013) or a maximum superposition Roweis (2003); Lücke and Sahani (2008); Bornschein et al. (2013). Also a combination of linear superposition followed by a sigmoidal nonlinearity (post-linear nonlinearities) have been investigated(nonlinear ICA Valpola et al. (2003), sigmoid belief networks Neal (1992)). By definition, noisy-or models and sigmoid belief networks assume hidden units and observed units to be binary, which generally entails different application domains than used for standard sparse coding. Furthermore, the implicit computational challenges have prevented a scaling to large numbers of hidden dimensions. In contrast, nonlinear ICA and models with maximum superposition can in principle assume continuous observed and hidden variables, and are consequently applicable to the same data domain as standard sparse coding. As for noisy-or models, nonlinear ICA is more challenging to scale to large hidden spaces, however. For the maximum nonlinearity, earlier models (Roweis, 2003) focused on inference instead of unsupervised learning of model parameters. Recent approaches demonstrated scalability of sparse coding with maximum nonlinearity to large hidden and observe dimensions (Maximal causes analysis 'MCA', Lücke and Sahani (2008); Bornschein et al. (2013)) but hidden variables were constrained to be binary in these cases. Binary priors avoid the analytical intractability usually resulting from continuous priors but they prevent a fine-tuned data representation and reconstruction with continuous coefficients. We will return to these approaches in context of the results in the Discussion section.

Methods

In this section we present the optimization of parameters in our model and the novel inference method developed to address the associated intractabilities.

Parameter Estimation

To estimate the model parameters Θ of the generative model in (5.3) we use expectation maximization (EM). We do inference in the E-step with our proposed method combining sampling and latent preselection (Shelton et al., 2011), which we will introduce in the next section. Optimization in the EM framework en-

tails setting the free-energy to zero and solving for the model parameters (M-step equations) (e.g., Neal and Hinton, 1998).

As an example we obtain the following formula for the estimate of image noise:

$$\hat{\sigma}^2 = \frac{1}{NDK} \sum_n \sum_d \sum_k \left(\max_h \left\{ W_{dh} s_{kh}^{(n)} \right\} - y_d^{(n)} \right)^2, \tag{5.7}$$

where we average over all N observed data points, D observed dimensions, and K Gibbs samples. However, this notation is rather unwieldy for a simple underlying idea. As such we will use the following notation:

$$\hat{\sigma}^2 = \left\langle W_{dh} s_h^{(n)} - y_d^{(n)} \right\rangle^*, \tag{5.8}$$

where we maximize for h and average over n and d. That is, we denote the expected values $\langle \, . \, \rangle^*$ to mean the following:

$$\langle f(s) \rangle^* = \sum_n \frac{\int_s p(\vec{s}|\vec{y}^{(n)}, \Theta) \, f(\vec{s}) \, \delta(\text{h is max}) \, d\vec{s}}{\int_s p(\vec{s}|\vec{y}^{(n)}, \Theta) \, \delta(\text{h is max}) \, d\vec{s}}, \tag{5.9}$$

where δ is the indicator function denoting the domain to integrate over, namely where h is the maximum. See the Supporting Information 1 for detailed derivation of update equations. Analogously, to compute the expectations of the Gaussian part of the prior distribution's parameters, the mean $\hat{\mu}_{\text{pr}}$ and the noise $\hat{\sigma}^2_{\text{pr}}$, we denote $\langle \, . \, \rangle^{**}$ to mean the following:

$$\langle f(s) \rangle^{**} = \sum_n \frac{\int_s p(\vec{s}|\vec{y}^{(n)}, \Theta) \, f(\vec{s}) \, \delta(s_h \neq 0) \, d\vec{s}}{\int_s p(\vec{s}|\vec{y}^{(n)}, \Theta) \, \delta(s_h \neq 0) \, d\vec{s}}, \tag{5.10}$$

which is identical to $\langle \, . \, \rangle^*$ in Equation (5.9) except that we are interested in support from *all* of the posterior distribution where $b_h = 1$, regardless of whether s_h is the maximal cause, and δ is modified accordingly.

Using the condensed notation in Equations (5.9) and (5.10) allows us to concisely express the update equations for the remaining model parameters:

$$\hat{W}_{dh} = \frac{\langle s_h y_d \rangle^*}{\langle s_h^2 \rangle^*}, \tag{5.11}$$

$$\hat{\pi} = \langle \delta(\vec{s}) \rangle, \tag{5.12}$$

$$\hat{\mu}_{\text{pr}} = \langle s_h \rangle^{**}, \tag{5.13}$$

$$\hat{\sigma}^2_{\text{pr}} = \langle (s_h - \hat{\mu}_{\text{pr}})^2 \rangle^{**}. \tag{5.14}$$

In this model $\vec{W}^{(h)}$ can be scaled by an arbirtray factor α when the corresponding s_h is scaled by $\frac{1}{\alpha}$. To prevent W from becoming arbitrarily large (which would lead to arbitrarily small values of $\vec{s}$), common practice is to constrain its columns (each latent cause) $(\vec{W}^{(h)})_{h=1}^H$ to have an l_2-norm less than or equal to one. Instead, we constrain all columns $\vec{W}^{(h)}$ to be equal to D (equivalent to normalizing expectation of W_{dh} to one, i.e. all entries are approximately equal to one). This normalization allows the $\hat{\mu}_{\text{pr}}$ to be intuitively more interpretable when comparing results on different datasets where the data dimensions D may vary.

As one can see in the above equations, in order to compute the parameter updates, we need to calculate several expectation values with respect to the posterior

distribution. However, as mentioned in the introduction, the posterior distribution of a model (linear or nonlinear) with a spike-and-slab prior is strongly multimodal. See Figure 5.2 for illustration of the posteriors in the two dimensional case for both (non)linear models with spike-and-slab and Laplace priors. Calculating expectations of this posterior is intractable, thus we must develop a new inference method in order to cope with these computations.

Inference: Exact Gibbs Sampling with Preselection of Latents

As described, parameter optimization is very challenging in this model. Consequently, current inference methods cannot address the task. In order to efficiently handle the intractabilities and the complex posterior (multimodal, high dimensional) illustrated in Figure 5.2, we take a combined approximate inference approach (Shelton et al., 2011). Specifically we design and propose an exact Gibbs sampler for our model in order to draw samples from the unique form of our posterior after we have reduced the set of latent variables to those with the most posterior mass. Reduction via preselection is not strictly necessary, but significantly increases efficiency when considering high dimensional posteriors, particularly in sparse models. As such, we will first describe the sampling step and preselection only later.

Gibbs Sampling

Our main technical contribution for efficient inference in this model is an *exact Gibbs sampler for the multimodal posterior*. Previous work has used Gibbs sampling in combination with spike-and-slab models (Olshausen and Millman, 2000), and for increased efficiency in sparse Bayesian inference Tan et al. (2010).

Our aim is to construct a Markov chain with the target density given by the conditional posterior distribution:

$$p(s_h|\vec{s}_{H\setminus h}, \vec{y}, \theta) \tag{5.15}$$

$$\propto\ p(s_h|\theta)\ \prod_{d=1}^{D} p(y_d|s_h, \vec{s}_{H\setminus h}, \theta). \tag{5.16}$$

We see from Equation (5.16) that the distribution factorizes into $D+1$ factors: a *single factor* for the *prior* and D *factors* for *each likelihood*.

As the difficult part to sample from is the likelihood, $\prod_{d=1}^{D} p(y_d|s_h, \vec{s}_{H\setminus h}, \theta)$, where the nonlinearity of the max plays a role, we begin with its construction and only afterwards will we include the spike-and-slab prior. For the point-wise maximum nonlinear case we are considering, the likelihood of a single D dimension, y_d, is a piecewise function defined as follows:

$$p(y_d|s_h, \vec{s}_{H\setminus h}, \theta) \tag{5.17}$$

$$= \mathcal{N}(y_d;\, \max_{h'}\{W_{dh'}s_{h'}\}, \sigma^2) \tag{5.18}$$

$$= \begin{cases} \underbrace{\mathcal{N}(y_d;\, \max_{h'\setminus h}\{W_{dh'}s_{h'}\}, \sigma^2)}_{\text{constant}} & \text{if } s_h < P_d \\ \mathcal{N}(y_d;\, W_{dh}s_h, \sigma^2) & \text{if } s_h \geq P_d, \end{cases} \tag{5.19}$$

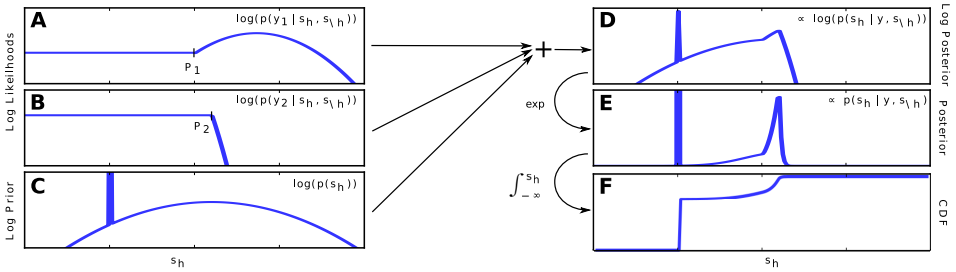

Figure 5.3: **Construction of SSMCA-induced posterior for the Gibbs sampler.** Left column: three contributing factors for the posterior $\propto p(s_h \mid s_{\setminus h}, \vec{y}, \Theta)$ in logspace. **A** and **B**: Log likelihood functions each defined by a transition point P_d and left and right pieces $r_d(s_h)$ and $l_d(s_h)$. **C** Log prior, which consists of an overall gaussian and the Dirac-peak at $s_h = 0$. **D** Log posterior, the sum of functions **A**, **B**, and **C** consists of $D+1$ pieces plus the Dirac-peak at $s_h = 0$. **E** Exponentiation of the **D** log posterior. **F** CDF for s_h from which we do inverse transform sampling.

where the *transition point*, P_d, is defined as the point where $s_h W_{dh}$ becomes the maximal cause:

$$P_d = \frac{\max_{h' \in \{H \setminus h\}} \{W_{dh'} s_{h'}\}}{W_{dh}}. \tag{5.20}$$

We refer to the two pieces of y_d in Equation (5.17)-(5.19) as the *left* and *right* pieces of the function: *left*, $l_d(s_h)$, when the latent cause is smaller than the transition point, $s_h < P_d$, and *right*, $r_d(s_h)$, when the latent is greater than or equal to the transition point, $s_h \geq P_d$. The left piece is constant with respect to s_h because the data is explained by another cause when the value of the latent s_h is smaller than the value of the transition point P_d, and the right piece is a truncated Gaussian when considered a PDF of s_h (see Figure 5.3**A-B**), because s_h is indeed explaining the data. Taking the logarithm of $p(y_d|s_h, \vec{s}_{H\setminus h}, \theta)$ transforms equation (5.17) into a left-piece constant and right-piece quadratic function. Expanding the expression for the logarithm of a given likelihood $p(y_d|s_h, \vec{s}_{H\setminus h}, \theta)$, each left and right piece (the respective sides of each transition point P_d) can be formulated as

$$l_d(s_h) = -\frac{1}{2}\log(2\pi) - \log(\sigma) + \frac{1}{2\sigma^2}(y_d - \max_{h' \setminus h}\{W_{dh'} s'_h\})^2 \tag{5.21}$$

$$r_d(s_h) = -\frac{1}{2}\log(2\pi) - \log(\sigma) + \frac{1}{2\sigma^2}(y_d - W_{dh} s_h)^2, \tag{5.22}$$

or more compactly

$$n_d(s_h) = \begin{cases} l_d(s_h) & \text{if } s_h < P_d \\ r_d(s_h) & \text{if } s_h \geq P_d, \end{cases} \tag{5.23}$$

which from now on will be referred to as an individual function segment of the entire likelihood function.

Now we generalize the likelihood expression in equations(5.17)-(5.19) to consider all observed D dimensions in $\vec{y}$. We take the logarithm of $\prod_{d=1}^{D} p(y_d|s_h, \vec{s}_{H\setminus h}, \theta)$, which results in $D+1$ left-piece constant and right-piece quadratic functions to be summed. The sum of all of these pieces will result in the desired D-dimensional likelihood function, which will be another piecewise function with $D+1$ disjoint

segments. In order to implement the summation of all of these segments efficiently, we need to first sort them by their transition points P_d, from smallest to largest values, which we denote by $\delta = argsort_d(P_d)$. With this notation, the summation of the pieces of the likelihood can be expressed:

$$\sum_{d}^{D} \log p(y_d|s_h, \vec{s}_{H\setminus h}, \theta) \tag{5.24}$$

$$= m(s_h) \tag{5.25}$$

$$= \begin{cases} m_1(s_h) & s < P_{\delta(1)} \\ m_2(s_h) & P_{\delta(1)} \leq s < P_{\delta(2)} \\ m_3(s_h) & P_{\delta(2)} \leq s < P_{\delta(3)} \\ \vdots & \vdots \\ m_{D+1}(s_h) & P_{\delta(D)} \leq s. \end{cases} \tag{5.26}$$

Importantly, we observe from equations (5.21)-(5.22) that each segment $m_d(s_h)$ is a 2nd degree polynomial, which can be represented by computing three coefficients. Thus, we can elegantly compute the operation in Equation (5.24) as the summation of the coefficients for each segment $m_d(s_h)$, and since all pieces $l_d(s_h)$ and $r_d(s_h)$ are polynomials of 2nd degree, the result is still a 2nd degree polynomial. So for all $D+1$ components of the likelihood in (5.17), we can compactly formulate (5.24) with

$$m_d(s_h) = \sum_{j=1}^{d-1} r_{\delta(j)}(s_h) + \sum_{u=d}^{D} l_{\delta(u)}(s_h). \tag{5.27}$$

$$= \sum_{d'=1}^{D} n_{d'}(s_h) \tag{5.28}$$

$$\text{for} \quad 1 \leq d \leq D+1$$

Now that we have computed the difficult part of the posterior, we incoorporate the *spike-and-slab prior* in two steps. The *Gaussian 'slab'* of the prior is taken into account by adding its 2nd degree polynomial to all the pieces $m_d(s_h)$, which also ensures that every piece is a Gaussian. The sparsity, or the 'spike', will be included only after constructing the full piecewise cumulative distribution function (CDF).

To construct the piecewise CDF, we relate each segment in $m_d(s_h)$ to the Gaussian $\propto \exp(m_d(s_h))$ it defines. Next, the *Bernoulli 'spike'* of the prior is accounted for by introducing a step into the CDF that corresponds to $s_h = 0$ (see Figure 5.3**F**), where the height of the step is proportional to the marginal probability $p(s_h = 0|\vec{s}_{\setminus h})$. Once the CDF is constructed, we simulate each s_h from the exact conditional distribution ($s_h \sim p(s_h|\vec{s}_{\setminus h} = \vec{s}_{\setminus h}, \vec{y}, \theta)$) by inverse transform sampling. Figure 5.3 illustrates the entire process.

Preselection

To dramatically improve computational efficiency of inference in our model, we can optionally preselect the most relevant latent variables before doing Gibbs sampling. This can be formulated as a variational approximation to exact inference (Lücke and Eggert, 2010) where the posterior distribution $p(\vec{s}\,|\,\vec{y}^{(n)}, \Theta)$ is approximated

by a truncated distribution $q_n(\vec{s};\Theta)$ which only has support on a subset $\mathcal{K}_n$ of the latent state space:

$$p(\vec{s}\,|\,\vec{y}^{(n)},\Theta) \approx q_n(\vec{s};\Theta) = \frac{p(\vec{s}\,|\,\vec{y}^{(n)},\Theta)}{\int_{\vec{s}'\in\mathcal{K}_n} p(\vec{s}\,'\,|\,\vec{y}^{(n)},\Theta)}\;\delta(\vec{s}\in\mathcal{K}_n) \tag{5.29}$$

where $\delta(\vec{s}\in\mathcal{K}_n)=1$ if $\vec{s}\in\mathcal{K}_n$ and zero otherwise. The subsets $\mathcal{K}_n$ are chosen in a data-driven way using a deterministic *selection function*, they vary per data point $\vec{y}^{(n)}$, and should contain most of the probability mass $p(\vec{s}\,|\,\vec{y})$ while also being significantly smaller than the entire latent space. Using such subsets $\mathcal{K}_n$, eq:. 5.29 results in good approximations to the posteriors. We define $\mathcal{K}_n$ as $\mathcal{K}_n = \{\vec{s}\,|\,\text{for all } h\notin\mathcal{I}:\ s_h=0\}$ where $\mathcal{I}$ contains the indices of the latents estimated to be most relevant for $\vec{y}^{(n)}$. To obtain these latent indices we use the cosine similarity as our selection function:

$$\mathcal{S}_h(\vec{y}^{(n)}) \;=\; \frac{\sum_d W_{dh}\;\vec{y}^{(n)}_d}{\left|\left|\vec{W}^{(h)}\right|\right|_2} \tag{5.30}$$

to select the $H'<H$ highest scoring latents for $\mathcal{I}$. This boils down to selecting the H' dictionary elements that are most similar to each datapoint, hence being most likely to have generated the datapoint. We then sample from this reduced set of latent variables.

Results

The above discribed procedure to optimize the parameters of the nonlinear spike-and-slab model will be referred to as SSMCA. All numerical experiments for SSMCA used a parallel implementation of the EM algorithm for parameter optimization (Bornschein et al., 2010), in which for the E-step we use our developed approximate inference scheme based on Gibbs sampling and latent variable preselection. For all described results, 1/3 of the samples are used for burn-in and 2/3 are used for computing the expectations. We initialized our parameters by setting the σ_{pr} and σ equal to the standard deviation observed in the data, the prior mean μ_{pr} is initialized to the observed data mean. W is initialized at the observed data mean with additive Gaussian noise of the σ observed in the data.

Parameter Recovery on Artificial Ground-Truth Data

The goal of the first set of experiments is to verify that our model and inference method produce an algorithm that can (1) recover ground-truth parameters $\Theta = (\pi, \mu_{\text{pr}}, \sigma_{\text{pr}}, W, \sigma)$ from data that is generated according to the model and (2) that it reliably converges to locally (if not globally) optimal solutions. We generate ground-truth data with $N = 2,000$ consisting of $D = 5\times 5 = 25$ observed and $H = 10$ hidden dimensions according to our model: N images with overlapping 'bars' of varying intensities and with Gaussian observation noise of variance $\sigma_{gt}=2$ (Figure 5.4**A**). On average, each data point contains two bars, $\pi = \frac{2}{H}$.

First, we optimize the model using just Gibbs sampling, which aims to do inference as exactly as possible in this model. Namely, we do sampling without

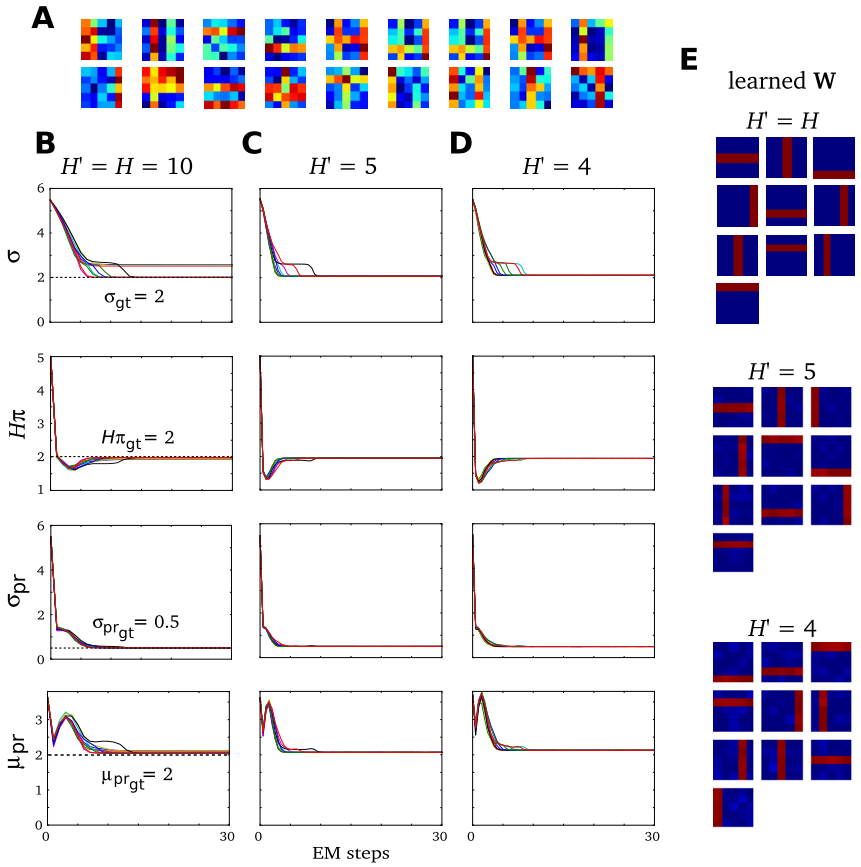

Figure 5.4: **Parameter recovery on synthetic data.** Results of three differently parameterized sets of experiments, each with 10 experimental runs of 30 EM iterations on identical artificial ground-truth data generated according to the SSMCA model: **A** $N = 2,000$, $D = 5 \times 5$. Three shown experimental settings are: **B** $H' = H = 10$, **C** $H' = 5$, and **D** $H' = 4$, although the same results were obtained by the entire range of parameters $H' = [4, 10]$. Importantly, the figure shows accurate recovery of ground-truth parameters which are plotted with dotted lines. **B**, **C** and **D** show in each column the parameter convergence of each of the three experiments, where the rows contain the following: data noise σ, sparsity $H \times \pi$, prior standard dev. σ_{pr}, and the prior mean μ_{pr}. Finally, **E** shows the set of learned generative fields/components $\vec{W}^{(h)}$ corresponding to each experimental set **B** $H' = H = 10$, **C** $H' = 5$, and **D** $H' = 4$.

preselection and draw samples from the entire latent space: we set the preselection parameter $H' = H$ and draw 30 samples from the full H-dimensional posterior. After this, we evaluate our combined approximate inference approach of preselection and Gibbs sampling. Results (Figure 5.4**B**,**E**) show that our algorithm converges quickly and recovers the generating ground-truth parameters.

Next, we investigate a range of numbers of samples drawn and consider the range of preselected latent variables $H' \in (4, 10)$ from the entire H-dimensional posterior space. These experiments yield the same results: our algorithm reliably converges quickly to (at least) locally optimal solutions of all parameters in all runs of the experiments with 30 EM iterations. This suggests that our approximation parameters do not strongly affect the accuracy of our inference results. See Figures (Figure 5.4**C**,**D**,**E**) for some further convergence examples, namely where $H' = 4$ and $H' = 5$.

Occlusions Data: Dictionary Learning and Image Reconstruction

In order to directly evaluate the differences between our nonlinear SSMCA model and the standard linear sparse coding model (which will be referred to as LinSC), we consider dictionary learning and image reconstruction on two datasets consisting of true occlusions. Here the task is to learn the set of components W, i.e. the dictionary elements, that are behind the composition of a given observed dataset $\vec{y}$, and consider reconstruction of individual images/datapoints $\vec{y}^{(n)}$. The goal of these experiments is to understand how the learned components are affected by the models' assumptions and furthermore the effect this has on the quality of the image reconstruction.

For the linear SC comparison we use the sparse online dictionary learning algorithm (Mairal et al., 2009b), which is a state-of-the-art matrix factorization sparse coding approach and is based on the objective function formulated in (5.1). Furthermore, in order to study the effect of the spike-and-slab prior, we apply the SSMCA algorithm with a narrow and fixed prior slab (small variance for the Gaussian of the prior distribution). Such a fixed narrow slab approximates a binary prior. Binary priors have thus far been used with nonlinear approaches (Dayan and Zemel (1995); Lücke and Sahani (2008); Lücke and Eggert (2010); Jernite et al. (2013); Bornschein et al. (2013)) including previous MCA versions Lücke and Sahani (2008); Lücke and Eggert (2010); Bornschein et al. (2013). We will refer to the SSMCA algorithm with fixed narrow slab as SSMCA$^{\text{fix}}$. To make sure that the differences in the results using SSMCA$^{\text{fix}}$ vs. SSMCA can be attributed to the difference between binary-like and spike-and-slab prior, we make sure that SSMCA and SSMCA$^{\text{fix}}$ are identical except for the algorithmic aspects concerned with learning the slab. Note that the data model underlying SSMCA$^{\text{fix}}$ connects to that of standard MCA (Lücke and Sahani, 2008; Lücke and Eggert, 2010) and becomes identical in the limit of an infinitely narrow slab (a delta peak). The algorithms for parameter optimization remain different also in this limit, however (SSMCA$^{\text{fix}}$ remains sampling based, for instance).

Realistic Occlusion Dataset

The first dataset we compare the algorithms on is one with controlled forms of sparse structure, a realistic artificial dataset of true occlusions (data created by actual occlusions and not following any model considered). The data was generated using the Python Image Library (PIL) to draw hundreds of overlapping edges/strokes in a 256×256 pixel image: each stroke had an integer intensity between $(1, 255)$, a width between $(2, 4)$ pixels, and a length, starting, and ending position drawn independently from a uniform distribution. The image was then cut into overlapping $D = 9 \times 9$ patches, each of which contained $k \in (0, 5)$ overlapping strokes, for $N = 61009$. Gaussian observation noise of $\sigma = 25$ and $\mu = 0$ was then independently added to each patch, thus concluding the considered occlusion dataset. Examples are shown in Figure 5.5. Additionally, the dataset also contains the corresponding (automatically obtained) labels for each image, indicating the *ground-truth number of occluding strokes* $k \in (0, 5)$ per image.

Such a dataset represents and isolates challenging aspects of low-level image statistics that are present in all natural images. Particularly, it contains edges of varying intensities and their occlusions. We have selected it because it is complex

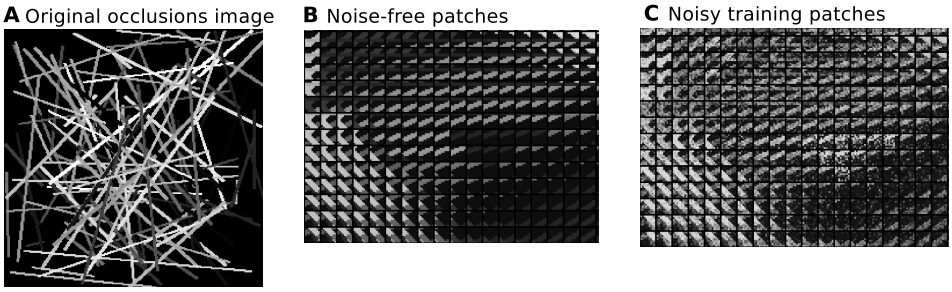

Figure 5.5: **Synthetic occlusion dataset and cut–out original and noisy patches.** Examples taken from the occlusion dataset. **A** shows an original noise-free image of generated occluding strokes of random width, pixel intensity, and starting/ending points. **B** shows a handfull of overlapping image patches cut from the original, noise-free data. **C** shows examples of the noisy training data, with independent $\sigma = 25$ noise added to **B**.

enough to narrow in on the consequences of the different model assumptions, but simple enough that we know what generated/caused the data. In this way, we can interpret the results and evaluate what each approach learns, particularly how they cope with occlusions.

We run the nonlinear SSMCA and the linear SC methods on the occlusions data set. We set the number of dictionary elements to be learned from the dataset to $H = 100$, but we also ran experiments learning larger ($H = 256$) dictionaries, which yielded the same results for both the linear and nonlinear methods. For SSMCA and SSMCA$^{\text{fix}}$, we draw 40 samples per datapoint, per variable (i.e. $40 \times 100 = 40000$ samples per datapoint when sampling 100 variables). The number of preselected latent variables was set to $H' = 10$ with 2 randomly chosen variables each iteration. For LinSC, we used regularization parameters $a = (1, 50, 100)$ in order to evaluate the reconstruction and the components learned across a range of sparse solutions.

The results showcase a number of notable effects. First, we see in Figure 5.6**A** the relationship between sparsity (number of components used for reconstruction) and data complexity (k number of strokes in the data). The complexity of the data reconstruction by SSMCA more closely follows the actual complexity in the data: the SSMCA plot (blue curves) shows a nearly linear relationship of the number of components used for reconstruction versus the number of components (strokes) actually in the data. In other words, although all methods adapt the number of fields used for reconstruction to the complexity of the data, our approach adapts to the extent of using nearly only as many components as are actually in the image (according to ground-truth). Furthermore, Figure 5.6**B** shows the relationship of the reconstruction quality versus the corresponding data complexity, in terms of the k number of strokes in the data. We quantify the quality of reconstruction with the mean squared error (MSE, $\sum_n (\vec{x}_n - \hat{\vec{x}}_n)^2$), which is very sensitive to subtle variances in an image versus its reconstruction. Notably, when the linear method is regularized to yield a solution as sparse as the nonlinear method (LinSC $a = 100$, cyan curves), its reconstruction MSE suffers.

Next, we investigate the actual components each model uses in order to reconstruct a given image patch. Figures 5.7**A-E** contains a comparison of the reconstruction of a handful of image patches by the linear and the nonlinear methods.

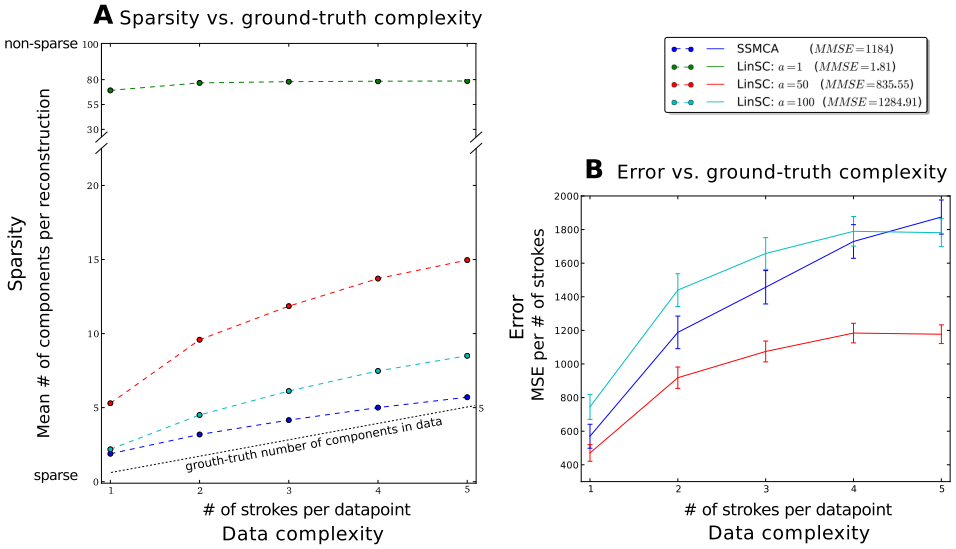

Figure 5.6: **Comparative experiments of linear and nonlinear sparse coding on dictionary learning and image reconstruction.** With $H = 100$ learned dictionary components we evaluate the number learned and used for reconstruction. **A** shows the relationship between sparsity (number of components used for reconstruction) and data complexity (number of strokes in the data). Interestingly, the SSMCA plot (blue curves) shows a nearly linear relationship of the number of components used for reconstruction versus the number of components (strokes) actually in the data, suggesting that reconstruction-complexity of the data by nonlinear model more closely follows the actual complexity in the data. On the contrary, the linear parameterization that yields good reconstruction results $a = 1$ shown in green, does not adapt to the data complexity at all: it consistently uses nearly 80 of the learned 100 components per reconstruction, regardless of the datapoint's actual complexity (note the change in scale of the y-axis around 30 components in order to fit the green curve on the plot). **B** shows the relationship of the mean squared error (MSE) of the reconstructions of all versus the corresponding data complexity (number of strokes in the data). When the reconstruction-complexity (sparsity) is far from the actual complexity of the data (linear methods: red, $a = 50$ and green $a = 1$ cases) the MSE improves. However, when the sparsity is more closely matched to the data, SSMCA and the weakly regularized linear methods result in a poorer MSE. SSMCA nevertheless yields a better MSE in this case, even when it and linSC $a = 100$ have a very similarly sparse solutions/use the same number of components. Note that the error of the least sparse LinSC approach ($a = 1$) is so low (mean MSE= 1.81), it does not even appear on this graph. Error bars shown are scaled to be 10% of the standard deviation for all methods in all stroke-complexity cases. The mean MSE (averaged over the entire dataset) is shown in the legend next to the respective algorithm.

Evaluation of the fields/components learned by each method suggests that the nonlinear max, which aims to model occlusions, is better able to learn generating causes of the occlusion-rich images. Regardless of image complexity – how many causes/strokes are in an image – the components used by the nonlinear method (SSMCA) resemble the true causes of the image: each component contains a single, interpretable stroke. On the other hand, none of the a parameterizations of the linear method yield stroke-like components, even when the solution is regularized to be as sparse as SSMCA. For example, if we just consider sparse solutions, namely compare the methods which use fewest components for reconstruction (SSMCA and LinSC with $a = 100$; blue and cyan curves, respectively), we see that not only is the nonlinear SSMCA solution consistently better in terms of MSE, but also the components learned/used are very different.

Although SSMCA extracts components resembling the generating causes, in some cases the reconstruction MSE suffers because the model does not allow for error correction via adding negative components (which, if it did allow for such corrections, would furthermore lead to a less sparse solution). In contrast, the linear methods are optimized for the best image reconstruction MSE using summation (as can be seen in the method's objective function in equation (5.1)), and consequently are able to learn a set of components which can be added/subtracted for the optimal MSE. This is particularly evident in the linear $a = 1$ case (green plots/highlighting), where sparsity is weakly enforced, and thus a larger set of components can be used to fine-tune a near-perfect reconstruction of the original image. Components learned by a control run with SSMCA$^{\text{fix}}$ with σ_{pr} fixed to 0.25 look similar to the ones of SSMCA and dissimilar to the ones of linear sparse coding (see Figure 5.8 for some examples). The learned sparsity is also similar to the one inferred by SSMCA but we observed an only weak with the complexity of the patches (for $\sigma_{\text{pr}} \leq 0.25$) to no scaling (for $\sigma_{\text{pr}} \geq 0.25$). Also the sparsity values were consistenly higher for SSMCA$^{\text{fix}}$ compared to SSMCA (i.e., fewer components for SSMCA$^{\text{fix}}$). Furthermore, the image reconstruction errors significantly increases for SSMCA$^{\text{fix}}$ compared to SSMCA (e.g., for $\sigma_{\text{pr}} = 0.25$ we get a MMSE of 1833). The significantly decreased reconstruction error is due to a decreased ability to finetune dictionary coefficients to the intensities of the components. For the SSMCA$^{\text{fix}}$ algorithm this also seems to indirectly influence the learned sparsity, maybe due SSMCA$^{\text{fix}}$ attributing components with lower than the learned contrast to background noise. Reconstruction errors and sparsity we observed to increase the more we narrow the fixed slab (the lower σ_{pr}).

Regarding all the results reported here, note that the max is also just an approximation of the true occlusion combination rule. If a dark stroke is occluding a brighter stroke, for instance, the true grey-value of the overlapping region is not reproduced by the max. Still, the SSMCA reconstruction is (given ground-truth strokes as dictionary elements) at least as good as in the linear case, and better except for boundary cases. Therefore, it seems to be easier for the nonlinear model to learn dictionary elements close to the generating components, i.e. interpretable components.

To summarize, SSMCA extracts meaningful, interpretable components – components closely match the generating process, adapts to complexity in the data, as measured by the number of strokes/edge components in an image, and uses correspondingly more or fewer components for the reconstruction. The reconstruction solution SSMCA offers is much sparser than that of LinSC, for any levels of re-

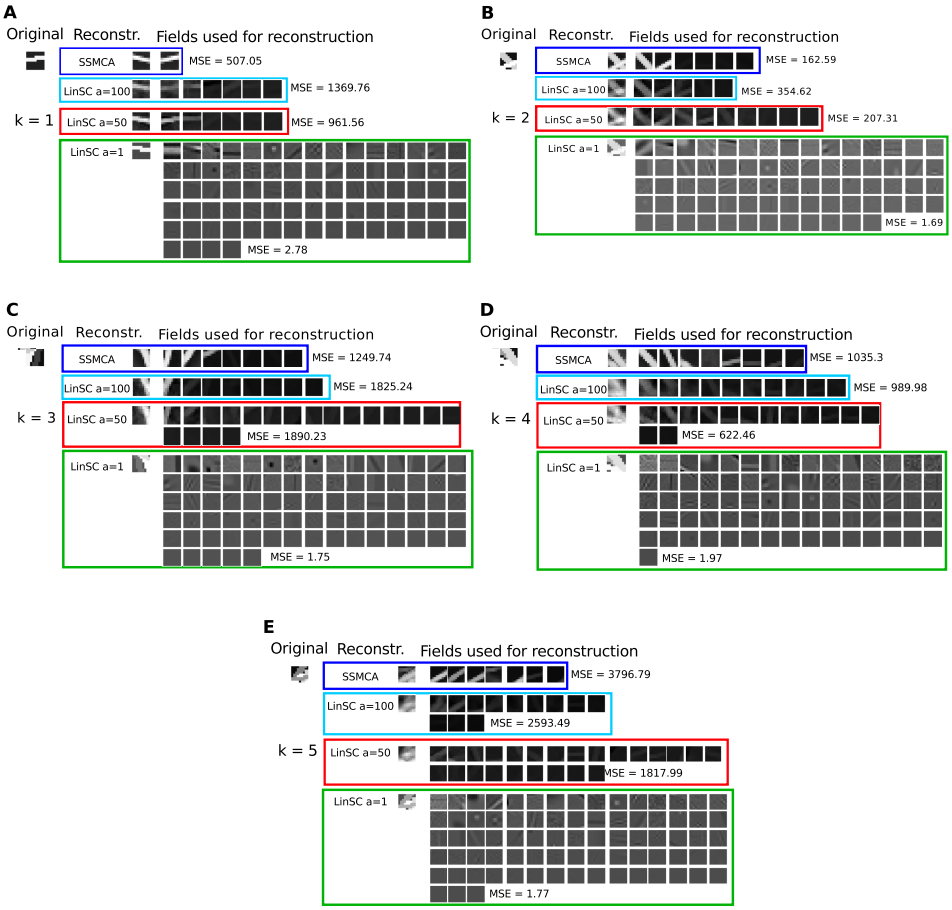

Figure 5.7: **Comparison of linear and nonlinear sparse coding on image reconstruction.** Shown are a handfull of real datapoints of varying complexity in terms of the number of strokes k in each image ($k \in (1, 5)$ strokes per image), the components/fields learned by the various algorithms, the corresponding reconstruction of the given datapoint, and the mean squared error (MSE) of each reconstruction. **A** image with $k = 1$ stroke, **B** $k = 2$ strokes, **C** $k = 3$ strokes, **D** $k = 4$ strokes, and **E** $k = 5$ strokes. Regardless of image complexity – how many causes/strokes are in an image – the components used by the nonlinear method (SSMCA) resemble the true causes of the image: each component contains a single, interpretable stroke. On the other hand, none of the a parameterizations of the linear method yield stroke-like components, even when the solution is regularized to be as sparse as SSMCA ($a = 100$). Note: all images in the $a = 1$ case appear brighter than they actually are, due to visualization with a python toolbox, but are in reality of the identical brightness scale to the original datapoint (and all other shown cases), hence the reconstruction error (MSE) is very low.

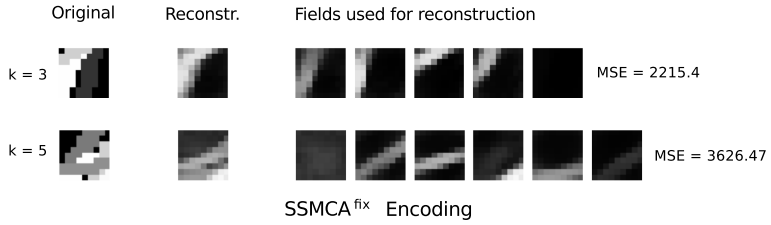

Figure 5.8: **Results of nonlinear sparse coding using a binary prior on image reconstruction.** The nonlinear sparse coding model if applied to artificial strokes using a fixed narrow slab (SSMCA$^{\text{fix}}$). The two figure columns show image reconstruction results for SSMCA$^{\text{fix}}$with $\sigma_{\text{pr}} = 0.25$ for two different ground-truth stroke numbers ($k = 3$ and $k = 5$). SSMCA$^{\text{fix}}$was first trained with fixed σ_{pr} and then applied to the data. Reconstructions were computed as described for Figure 5.7.

construction error (MSE).

As a control, we also ran the same set of experiments, but varying H and H' – learning a larger set of latent components (dictionary set) H and ranging the SSMCA preselection parameter H' values – all of which resulted in the same trends shown in Figures 5.6 and 5.7.

Natural Image Occlusions

We have shown that our approach can model realistic artifical occlusions well. Now we are interested in investigating the performance of the linear and nonlinear approaches on naturally occuring occlusions. We use an image of underbrush in a forest (taken bridge.jpg, which has been used for denoising benchmarking by Mairal et al., 2009b), which is rich with occluding branches and twigs. See Figure 5.9**A** for the original noise-free image, from which we cut a 110×110 pixel occlusion-rich section and scaled it up to 256×256 pixels to use in our dataset, shown in Figure 5.9**B**. To compose the dataset as in the previous experiments, we cut the 256×256 image, with pixel values x_i ranging from $(0, 255)$, into $N = 61009$ overlapping image patches of $D = 9 \times 9$ pixels, then add independent Gaussian noise with $\sigma = 5$. We run the exact same set of experiments as with the original occlusions dataset, with both the nonlinear and linear methods learning a dictionary size of $H = 100$ latents. For SSMCA we again draw 40 samples per datapoint, per variable (i.e. 40×100 samples per datapoint), and set the number of preselected latent variables to $H' = 10$ with 2 randomly chosen per iteration. For linear SC, we again used regularization parameters $a = (1, 50, 100)$.

Because we do not have any ground-truth associated with this dataset as to how many strokes/components are in a given image, we can only compare the average reconstruction error for the entire dataset across methods. Figure 5.9**C** shows the MSE of each method with the associated standard deviation. The results follow the trend outlined in the previous set of experiments (in Figure 5.6**B**), where again if LinSC uses as sparse a reconstruction as SSMCA (in $a = 100$ case), the mean reconstruction error is far poorer than that of SSMCA (MMSE = 269.96 vs. MMSE= 75.71). Furthermore, even when LinSC is less sparse (in $a = 50$ case), the mean reconstruction error is still slightly poorer than SSMCA (MMSE = 83.89 vs. MMSE= 75.71). On the other hand, when the linear model uses a highly non-sparse solution (LinSC $a = 50$, resulting in using 75 of 100

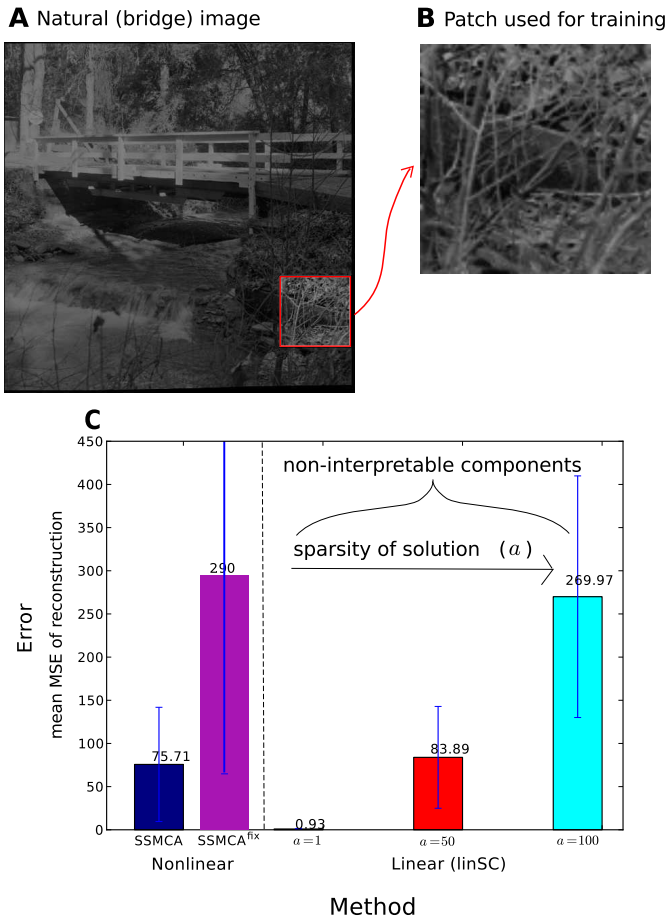

Figure 5.9: **Results of comparative experiments of linear and nonlinear sparse coding methods on component learning/image reconstruction on natural image patches.** **A** shows the original natural image data, bridge.jpg (Mairal et al., 2009b), from which we cut an occlusion-rich underbrush region. **B** shows the original section taken from **A**, scaled up to 256×256 pixels, which was then cut into overlapping patches and given independent Gaussian noise with $\sigma = 5$ to compose the considered dataset. **C** shows the mean squared error (MSE) of the compared nonlinear and linear methods' reconstruction averaged over the entire dataset, with the standard deviation indicated with error bars. The trend is the same as in the artifical occlusions data experiments: the nonlinear method maintains reasonably low MSE, while learning a sparse set of interpretable components, whereas the linear method acheives a very low MSE only when it does not learn a sparse (and never interpretable) solution of components.

components for reconstruction), it can fine-tune its reconstruction to acheive very low error (MMSE = 0.93). However, the components each linear model uses for reconstruction are non-interpretable (e.g. do not resemble edge-like structures) for any of the linear models, regardless of their sparsity or reconstruction error both nonlinear models use components that indeed resemble edge-like structures and are interpretable.

When applying SSMCA$^{\text{fix}}$as a control, the learned dictionary components are similar to the ones by SSMCA, however, the reconstruction error is much worse than that of SSMCA with an MMSE of 290 for $\sigma_{\text{pr}} = 0.25$ (see Figure 5.6 for comparison). When we make the slab still narrower, the reconstruction error further increases (e.g., MMSE= 377 for $\sigma_{\text{pr}} = 0.1$), which is consistent with a reduction of the ability to accurately match the varying stroke intensities using continuous coefficients.

Natural Image Patches and Neural Consistency

Understanding the encoding provided by sparse coding and its capability to extract interpretable data components is important for functional applications but, furthermore, also of high relevance for probabilistic models of the primary visual cortex (V1). Since the seminal study by Olshausen and Field (1997) sparse coding can be considered as a standard model for the response properties of V1 simple cells. Evidence that response properties of V1 simple cells may be better described by a sparse coding model that reflects occlusions has been provided by a recent comparative study (Bornschein et al., 2013). To complete our investigation of the SSMCA model, we will apply it to the same data as used in that study. In contrast to the binary sources assumed by Bornschein et al. (2013), our model allows us to study the statistics of functions under the standard assumption of continuous latents.

We apply our model to $N = 50,000$ image patches of $D = 16 \times 16 = 256$ pixels and learn $H = 500$ hidden dimensions/generative fields, and run 50 EM iterations with 100 samples per datapoint. The patches were extracted from the Van Hateren natural image database (van Hateren and van der Schaaf, 1998) and subsequently preprocessed using pseudo-whitening (Olshausen and Field, 1996). We split the image patches into a positive and negative channel to ensure $y_d \geq 0$: each image patch $\tilde{y}$ of size $\tilde{D} = 16 \times 16$ is converted into a datapoint of size $D = 2\,\tilde{D}$ by assigning $y_d = [\tilde{y}_d]^+$ and $y_{\tilde{D}+d} = [-\tilde{y}_d]^+$, where $[x]^+ = x$ for $x > 0$ and $[x]^+ = 0$ otherwise. This can be motivated by the transfer of visual information by center-on and center-off cells of the mammalian lateral geniculate nucleus (LGN). In a final step, as a form of local contrast normalization, we scaled each image patch so that $0 \leq y_d \leq 10$.

All results are shown in Figure 5.10. In Figure 5.10**A**, we have a handfull of the learned dictionary elements $\vec{W}^{(h)}$ (which are a variety of Gabor-Wavelet and Difference of Gaussians (DoG)-like shapes). To quantitatively interpret the learned fields, we perform reverse correlation on the learned generative fields and fit the resulting estimated receptive fields with Gabor wavelets and DoGs (see Supporting Information 2 for details). Next, we classify the fields as either orientation-sensitive Gabor wavelets or 'globular' fields best matched by DoGs. In Figure 5.10**B** we compare the percentages of 'globular' fields to *in vivo* recordings. These results are consistent with neural recordings: notably, the proportion of DoG-like fields in the

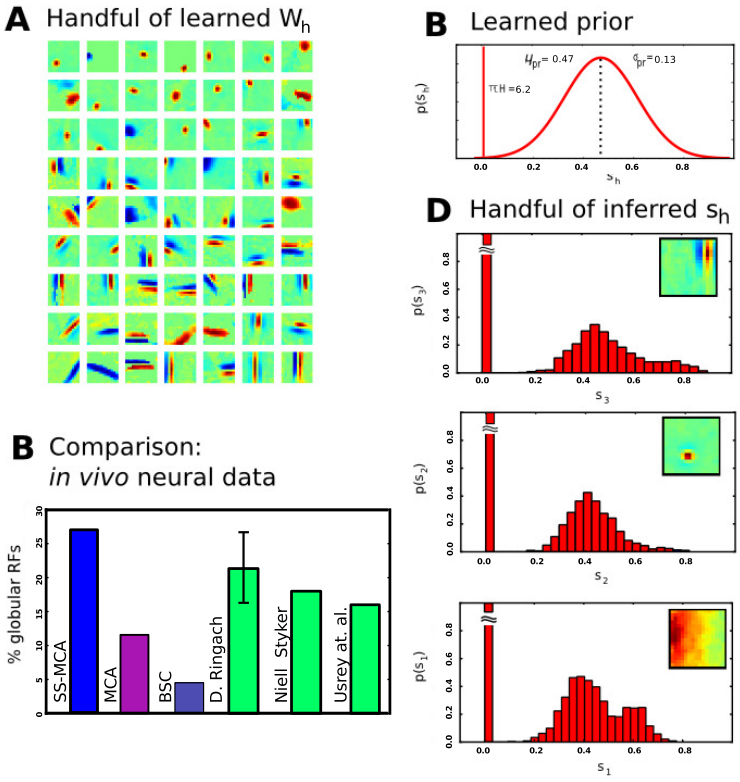

Figure 5.10: **Analysis of dictionary components learned by the SSMCA algorithm on natural image patches. A** Example dictionary elements $\vec{W}^{(h)}$ after learning. **B** Fraction of globular fields estimated from *in vivo* measurements, compared to ours (after fitting with Gabor wavelets and DoG's; globular percentages taken from Bornschein et al. (2013) who analyzed data provided by Ringach (2002) and estimated percentages of globular fields from data in two further papers Usrey et al. (2003); Niell and Stryker (2008). **C** Learned prior. **D** Actual activations of diverse dictionary elements s_h (posterior averaged over data points).

same high range as the proportions found in different species (Ringach, 2002; Usrey et al., 2003; Niell and Stryker, 2008) (See Bornschein et al. (2013) for data and a discussion), which is a result not observed by the established linear SC variants. The learned prior and its parameters are shown in Figure 5.10**C**: learned sparseness was $\pi H = 6.2$ (i.e. on average six active latent variables per image patch), mean $\mu_{pr} = 0.47$, with standard deviation $\sigma_{pr} = 0.13$. The learned data noise was $\sigma = 1.4$. Exhibiting consistency with the learned prior, Figure 5.10**D** shows a handfull of the inferred latent variables (coefficients) s_h. These correspond to the actual activations of the diverse dictionary elements $\vec{W}^{(h)}$, each of which is visualized in the upper right of each subfigure. A part of these results have also appeared in a preliminary application of this model in a conference submission (Shelton et al., 2012).

Since we have shown consistent predictions with neural recordings, we finally test the model for consistency with the natural image patches dataset. Specifically, we are interested in consistency of the prior beliefs with inferred beliefs, as it is a necessary condition of the correct data model that the posterior averaged over the datapoints $\vec{y}^{(n)}$ matches the prior (compare e.g., Berkes et al., 2011):

$$\lim_{N\to\infty} \tfrac{1}{N} \textstyle\sum_n p(\vec{s}\,|\,\vec{y}^{(n)}, \Theta) = p(\vec{s}\,|\,\Theta). \tag{5.31}$$

After the learning on image patches as described above, we observed that averaged posteriors over data points closely resemble the learned prior (see Figure 5.10E for examples). Linear sparse coding has reportedly struggled with this consistency condition (see Olshausen and Millman, 2000, for a discussion).

Discussion

In this work we introduced a sparse coding model that modifies standard sparse coding in two ways: it uses a spike-and-slab distribution instead of a Laplace prior and the nonlinear max superposition instead of the standard linear superposition. With these additions, the proposed model can realistically model low-level image effects. Particularly, the nonlinearity of the max equips the approach to well-approximate occlusions.

As learning and inference in a model with these two modifications is difficult, we also proposed a combined preselection-sampling scheme that constructs the conditional posterior with high accuracy and efficiency. This inference approach allowed us to apply, for the first time, a sparse coding model with continuous latents and strongly nonlinear combination to reasonably high-dimensional observed and hidden space dimensions. The approach is therefore applicable to the typical application domains of standard sparse coding. Furthermore, it offers itself as a novel model for neural responses that encode component intensities. Unlike (linear and nonlinear) models with binary latents (Haft et al., 2004; Henniges et al., 2010; Bornschein et al., 2013), it can capture a more fine-tuned representation of sensory stimuli.

Our main interest in this work was in gaining deeper understanding of the consequences of the component combination assumption (linear or nonlinear) and to highlight these consequences empirically in numerical experiments. First, in experiments on artificial data, we have shown that the model and inference approach can learn ground truth parameters. Furthermore, using experiments on natural image patches, we showed consistency of our model in two ways: its predictions are consistent with (1) *in vivo* neural recordings and with (2) its prior beliefs. Our experiments on dictionary learning and image reconstruction show, as the crucial difference, that the nonlinear method learns and uses interpretable image components when reconstructing a given image patch (unlike the linear method Mairal et al. (2009b)). Namely, we have defined '*interpretable*' to mean that the extracted components *closely match the generating process*. Furthermore, we have shown that our method adapts to complexity in the data and uses correspondingly more or fewer components for the reconstruction. Not only does our method yield meaningful and adaptive solutions, but its solution is always much sparser than that of any of the comparable parameterizations of the linear SC method, for any levels of corresponding reconstruction error (MSE).

Our results consequently show that the max nonlinearity is sufficient to reproduce many properties desired from a hidden causes approach to image patch modeling – especially "interpretable" encoding. Future work could go even further, e.g., by taking into account object depths for a more explicit occlusion modeling. The challenges for inference are significantly increasing in this case as unconstrained object permutations result in a super-exponential scaling of hidden states. While explicit occlusion models can be developed based on similar meth-

ods as used here (see Henniges et al. (2014) and citations therein), a combination with a prior for continuous variables such as the spike-and-slab distribution, poses a considerable and yet to be mastered scientific challenge. Work using the max nonlinearity and related approaches, therefore, focuses on capturing the essential properties of occlusion with more compact models, which allow for larger-scale applications comparable, e.g., to those possible with linear sparse coding approaches. Work by Zoran and Weiss (2012) aims at capturing occlusion nonlinearities using a mixture model approach, and a comparison with linear models shows significant advantages and improved interpretability. Other recent work combines translation invariance with the exclusiveness property of occlusion (Dai et al., 2013). Although that work offers a multiple-causes approach as the one proposed here, they do not provide a model for a continuous distribution of component intensities. Linear approaches are also being continuously further developed. By using a massive increase in the number of hidden units (Olshausen, 2013), it can be observed that, e.g., globular components (compare with the Results section on Natural image patches) can also emerge using linear approaches (also see Bornschein et al., 2013, for a discussion). In such highly overcomplete settings, sparsity can be increased, which tends to increase the interpretability.

Further linear approaches include non-negative matrix factorization (NMF) methods which are usually not formulated probabilistically. Previous work Lücke and Sahani (2008) quantitatively compared different NMF versions to MCA with binary hidden units, which itself is approximated by the SSMCA$^{\text{fix}}$evaluated as control here. Using the bars benchmark test (which can be considered as a simplified version of the data used in the dictionary learning and image reconstruction experiments, see Results), it was shown that MCA performs well in this nonlinear task. Already for the comparably simple bars test, standard NMF was shown to fail (experimental data from Spratling, 2006). Only if constrained appropriately, using hand-tuned constraints on sparsity for weights and latent activity, NMF was reported to learn the correct generating components. Such constrained extensions for NMF objective functions can be combined with any noise metric (e.g., Poisson, Gaussian, Manhatten; compare Hoyer (2004); Guan et al. (2012)), but the sparsity parameter in these approaches is hand-fixed and cannot be learned. In a probabilstic formulation, constraints could be reformulated as priors and indeed be learned, which could potentially make them more similar to the approach used here. Regarding the inherent superposition assumption, all NMF approaches are, by definition, linear and thus in this respect more similar to linear sparse coding than to SSMCA. Consequently, the linearity assumption used by NMF could explain why, e.g. the algorithm requires additional mechanisms in order to learn the correct solution for the bars experiments.

In conclusion, this work marks first steps in uncovering the benefits and drawbacks of the implicit assumptions made within sparse coding models, the understanding of which will help researchers to select the most suitable model for their task. If the primary goal is for image reconstruction, our experiments suggest the linear model to be the better choice, whereas if the goal is to extract a sparse dictionary set approximating the data generation, our approach would be more beneficial.

Acknowledgements: The work was funded by the German Research Foundation (DFG) under grant LU 1196/4-2 and by the Cluster of Excellence EXC 1077/1

"Hearing4all". Initial stages were funded by the German Ministry of Research and Education (BMBF) under grant 01GQ0840 (BFNT Frankfurt) and by the LOEWE Neuronal Coordination Research Focus Frankfurt (NeFF). Furthermore, we acknowledge support by the Frankfurt Center for Scientific Computing (CSC).

Supporting Information 1

M-Step Equation Derivations

The optimal parameters that maximize the data log-likelihood under the generative model can be sought by Expectation Maximization (EM) algorithm (see eg., Neal and Hinton (1998)), which iteratively optimizes a lower bound $\mathcal{F}(\Theta, q)$ of the likelihood w.r.t. the parameters Θ and a distribution q:

$$\mathcal{L}(\Theta) \geq \mathcal{F}(\Theta, q_{\Theta'}) = \sum_{n=1}^{N} \sum_{s} q_n(\vec{s}|\Theta') \log \frac{p(y^{(n)}, \vec{s}|\Theta)}{q_n(\vec{s}|\Theta')} \tag{5.32}$$

$$= \langle \log p(\vec{y}, \vec{s}\,|\,\Theta) \rangle_{q(\vec{s}|\Theta')} + \mathrm{H}[q(\vec{s}|\Theta')]. \tag{5.33}$$

Each iteration consists of an E-step and an M-step. The E-step optimizes the lower bound w.r.t. to the distributions $q_n(s\,|\,\Theta)$ by setting them equal to the posterior distributions $q_n(s\,|\,\Theta) \leftarrow p(s\,|\,y^{(n)}, \Theta)$ while keeping the parameters Θ fixed, denoted by Θ'. The M-step then optimizes $\mathcal{F}(\Theta, q_{\Theta'})$ w.r.t. the parameters Θ keeping the distributions $q_n(s\,|\,\Theta')$ fixed. If we are given many samples of s for the posterior then we wish to find:

$$\Theta^{(t+1)} = \; mathrmargmax_{\Theta} \mathcal{F}(\Theta, q_{\Theta^{(t)}}). \tag{5.34}$$

This is maximised with the maximum likelihood estimate:

$$\Theta^{(t+1)} = \; mathrmargmax_{\Theta} \langle \log p(\vec{y}, \vec{s}\,|\,\Theta) \rangle_{q(\vec{s}|\Theta^{(t)})}. \tag{5.35}$$

To keep the derivation focused, we present a simple derivation of the update equations only for a single element of W. The other parameters are similarly derived and are not covered here. For pedagogical purposes we first derive an update equation *without* a max rule, then we show how this rule should be modified when the max rule is used. Assuming the data $y^{(n)}$ is distributed as follows:

$$y^{(n)} = w s^{(n)} + \varepsilon \tag{5.36}$$

where $\varepsilon \sim \mathcal{N}(\mu = 0; \sigma^2)$. for w. This gives the conditional probability as:

$$p(y^{(n)}\,|\,s^{(n)}, w) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y^{(n)} - w s^{(n)}}{\sigma}\right)^2\right) \tag{5.37}$$

In log space this is a quadratic function:

$$\log p(y^{(n)}\,|\,s^{(n)}, w) = c - \log\sigma - \frac{1}{2}\left(\frac{y^{(n)} - w s^{(n)}}{\sigma}\right)^2 \tag{5.38}$$

and is summed over all datapoints n. The maximum likelihood solution differentiates this sum with respect to w (this function is linear in σ and when differentiated σ can be discarded) to find the maximum:

$$\frac{d}{dw}\left[\sum_{n}\left(y^{(n)} - s^{(n)} w\right)^2\right] = 0. \tag{5.39}$$

From which the maximum is given by:

$$w = \frac{\sum_n s^{(n)} w^{(n)}}{\sum_n s^{(n)2}}. \tag{5.40}$$

However, we care about finding the ML solution for the max rule:

$$y^{(n)} = \max_h \left\{ W_h s_h^{(n)} \right\} + \varepsilon \tag{5.41}$$

If the new estimates of W_h do not change significantly then the simple derivation for w will apply to W_h, but only the data for which W_h is the maximum will be used. The data is going to vary over: the number of images N, the number of samples per image K, and we will estimate W_{hd} per latent dimension h and observed dimension (or pixel) d. This leads to:

$$W_{hd} = \frac{\sum_n^N \sum_k^K \delta(\text{h is max}) s_{hn}^{(k)} y_d^{(n)}}{\sum_n^N \sum_k^K \delta(\text{h is max})\ {s_{hn}^{(k)}}^2} \tag{5.42}$$

which corresponds to the results given in equation (9) of the main paper. $\delta(\text{h is max})$ is used to identify the index for which $W_{hd} s_{hn}^k$ is the maximal cause of the data, if it is not the maximal cause, then $\delta(\cdot)$ returns 0, and the term does not contribute to the sum.

Supporting Information 2

Experiments: Natural Image Patches

Here we show further results from the experiment on $N = 50,000$ preprocessed and channel split natural image patches of 16×16 pixel.

First we relate the generative fields that are learned from image patches to their corresponding receptive fields as measured in biological systems. In order to do so we carried out several reverse correlation experiments, with the aim of identifying the relationship between the generative fields and the reverse-correlation fields. To calculate the reverse correlation for a single latent variable we calculate the average activation for a particular image (since the code is sparse, most of the time activations will be zero). The images are then averaged together, weighted by the average activation.

In Figure 5.11**A** we show the generative fields that are learned with our method. Figure 5.11**B** shows the receptive fields obtained by estimating the first order linear mapping from input to hidden units. The mapping is estimated by combining the preprocessing (a linear mapping with a kernel for pseudo-whitening) with the mapping obtained by reverse correlation using preprocessed patches. As can be observed by comparing Figure 5.11**A** and **B**, the qualitative and quantitative shapes of generative and receptive fields are essentially equal. For the results in the main text in Figure 9 we used the receptive field estimates in Figure 5.11**B**.

For further analysis, we matched the fields with Gabor and DoG functions to determine the numbers of Gabor-like and DoG fields. The corresponding percentage of DoG or "globular" fields was than used for comparison with *in vivo* measurements in Figure 9**C**. The percentages of "globular" fields of the *in vivo* studies were taken from the study by Bornschein et al. (2013) who used original

recordings by Ringach (2002). Please see the preliminary study (Shelton et al., 2012) for more details and analysis of the model's application to response properties in primary visual cortex.

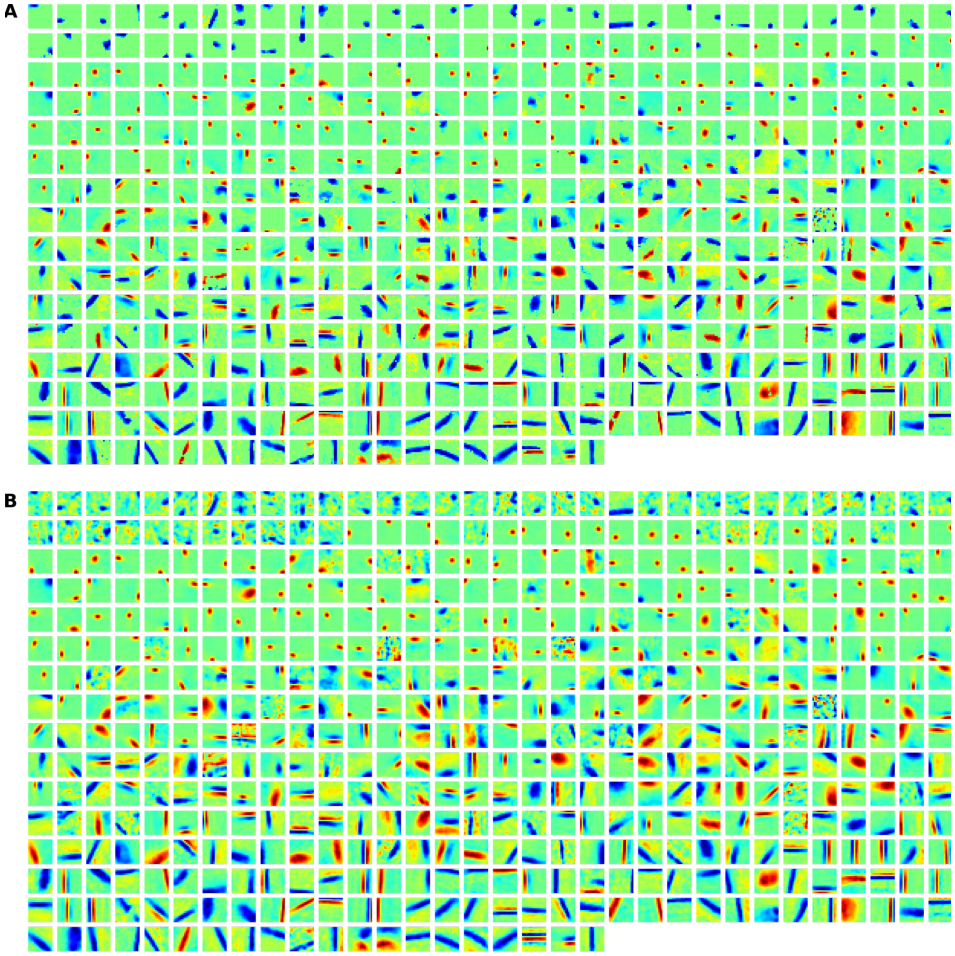

Figure 5.11: **A** Full set of $H = 500$ learned generative fields ($\vec{W}_h$). **B** Fields after reverse correlation with preprocessed input patches.

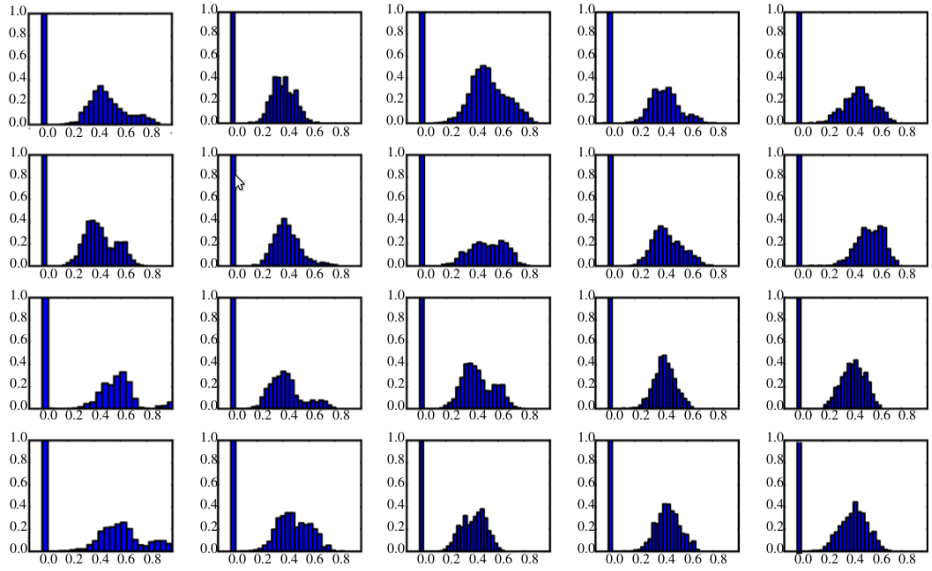

Figure 5.12: Histogram of the latent activations of the 20 most often active dictionary elements. This corresponds to the prior over latent units that we have assumed in our model, thus supporting the consistency of the model.

Chapter 6

Conclusions

Since its advent, sparse coding has produced a vast body of research in the area of data-driven modeling, where the notion of sparse latent representation has spurred the development of a variety of latent variable methods for automated data analysis. With a growing number of applications, research activity in the field of sparse coding remains dense, focusing primarily at the development of more comprehensive data models and efficiently scalable inference techniques. The work presented in this work extends into both of the aforementioned research dimensions. With respect to having more accurate models, here we have worked with a more natural choice for a sparse prior over latent variables, namely the spike-and-slab distribution. While the continuous slab that we assume to be Gaussian distributed models the strengths of active latent components, the binary spike of the distribution allows for an exact encoding of the absence or presence of latent components, hence inducing true sparsity. This is in contrast to more common preferences for sparse priors like Cauchy or Laplace, which cannot exactly induce true sparsity (i.e., zero values) with a high likelihood. We show that for the spike-and-slab prior, the (learnable) sparsity parameter in fact offers a great degree of control over the nature of latent activity: for instance in the linear model (e.g., Chapter 2), the spike-and-slab prior can be flexibly parameterized to generate observed data distributions that can range between being very sparse and long-tailed to fully spread-out in both latent and observed spaces (Figure 2.1). By employing the spike-and-slab prior we indeed demonstrate that we can fit data generated by a standard sparse coding model (using a Cauchy or Laplace prior) with high precision (see Chapters 2 and 3).

Another gain of having the Gaussian spike-and-slab distribution as a prior is the theoretical result presented in Chapter 2. Since both the prior slab and the observation generation process are governed by Gaussian distributions, we apply standard Gaussian identities (Appendix 2.E) to derive a closed-form E-step expression for computing an exact posterior over the latents for a given observation. This is unlike the standard sparse coding, where the posterior cannot be computed in closed-form. To the best of our knowledge, an exact inference method for a spike-and-slab sparse coding model is a novel contribution of Chapter 2.

We find a few other studies on spike-and-slab sparse coding that also take into account the special case of the Gaussian slab; the works however propose different inference procedures for varied formulations of the model and associated optimization problems.

In terms of performance, the Gaussian spike-and-slab model achieves competitive results on standard source separation benchmarks, but since the exact inference requires to visit a number of posterior modes that scales exponentially with respect to the number of latents, the computational cost of the method makes it infeasible for large-scale applications. However, by performing the exact inference we also learn that for data generated by sparse latent causes, dense volumes of posterior probability are actually found in latent subspaces that only span over a few latent components (e.g., Figure 3.2). Therefore in Chapter 3, we apply a latent space truncation technique to approximate the exact inference procedure. Through truncation, we exclude a vast majority of latents from posterior estimation that are found to be the least likely causes of an observation, hence we avoid a large number of insignificant posterior modes which are otherwise associated with the activation combinatorics of the excluded components. The truncated inference computationally decouples itself from the number of latent components, which allows us to apply our sparse coding approach on a large-scale to a number of tasks.

In Chapter 3, our comparison between spike-and-slab and standard sparse coding on source separation benchmarks reveals that the former clearly outperforms the latter; however, standard sparse coding can be computationally more efficiently formulated as a non-probabilistic, regularized optimization problem. In continued source separation experiments, we find our method steadily competitive or better than a recently proposed and popular variational inference approach for spike-and-slab sparse coding. The truncated approach also performs well on image denoising and achieves state-of-the-art results among a number of spike-and-slab and other comparable methods. We furthermore learn that although the variational approach is computationally more efficient due to its linear complexity, when afforded unconstrained computational resources, the method tends to saturate much earlier than the truncated approach, which continues to utilize the resources for much longer to further improve on its results.

In Chapter 4, we continue on to develop a highly scalable inference scheme for linear spike-and-slab sparse coding. We follow the previously introduced select and sample framework by fusing together latent preselection (as applied in Chapter 3) with sampling to devise an approximate inference scheme that only incurs a linear cost with respect to the number of selected latents. For the sampling part, we benefit from the Gaussian slab to derive a computationally efficient exact Gibbs sampler. Through method verification experiments involving ground-truth information we show that not only combining the preselection with sampling is computationally extremely efficient, but it also helps the inference procedure

to become much less prone to local optima (Figure 4.3). Moreover, the image denoising experiments on a real benchmark show the method's competitive performance against a variety of other approaches including other spike-and-slab methods. The experiment on image patches furthermore demonstrates that our select and sample based inference can indeed be applied to scale-up spike-and-slab sparse coding to infer thousands of latent components from very large datasets, essentially making the approach one of the most scalable sparse latent variable methods that can efficiently incorporate sampling for sophisticated posterior inference.

In the last chapter we study a nonlinear spike-and-slab sparse coding method as a better approach for low-level encoding of visual data. We deal with the analytical and computational intractability of performing inference in the model by applying the select and sample procedure. Our experiments on strokes data demonstrate that the nonlinear method is indeed coherent with the ground-truth in terms of inferring the structure of the latent components and in turn encoding the input images with accurate sparsity levels. We on the other hand find standard sparse coding to fall short on both the fronts. Although we find the linear approach better at minimizing the error on data reconstruction, but when constrained to be sparse, the reconstruction performance of the model deteriorates beyond that of the nonlinear approach for comparable sparsity levels. The results therefore indicate that the nonlinear method is better suited for application scenarios with a restricted budget for latent activity e.g., compressive sensing. The final analysis in the light of *in vivo* neural data further affirms the nonlinear model as a statistically more accurate alternative for simulating the functional aspects of simple V1 cells.

We believe that this work presents a valuable contribution to the field of sparse coding and beyond. We have put our efforts in improving the standard approach to sparse coding. By developing novel inference techniques, we have overcome learning and scalability challenges pertinent to our modeling assumptions. Through various experiments between spike-and-slab and (constrained to be sparse) standard sparse coding, we have observed spike-and-slab models to have a clear performance edge. Although the probabilistic nature of spike-and-slab sparse coding models make them computationally more complex to optimize, we demonstrate that the approximate optimization techniques we apply deliver a fair deal of control over the thoroughness versus scalability of the inference procedure. There are also various aspects of our work that encourage further investigation, e.g., to possibly obtain more grounded ways of adjusting the values of free-parameters. Efforts can further be put into finding sophisticated selection procedures that can for instance efficiently consider (high-order) correlations among the components to maximize mutual exclusiveness of the selected latents. The approaches we have developed can also be assessed in the context of high-level tasks such as by integrating them into classification hierarchies either as feed-forward filters

or to utilize their potential for multimodal posterior inference to incorporate uncertainty into the process. The probabilistic models we consider can also be transformed into directed hierarchical models for classification by augmenting the binary part of the latents with a class-conditional structure e.g., by defining a class-driven polynomial distribution to govern the activations of latent components given a class label. Such hierarchical models can still be optimized for an affordable computational cost by means of the inference methods proposed in this work.

Contributions

A major part of this thesis is based on published research work, which resulted from different collaborations between me and my peers. Here I will outline the contributions of each individual to each of the chapters in this thesis.

Chapter 1: I wrote the introductory chapter.

Chapter 2: This chapter has been published as Lücke and Sheikh (2012). Alphabetically listed, Jörg Lücke (JL) and I myself (AS) are the joint first coauthors of the work. Both JL and AS derived in parallel the analytical results presented in the chapter. JL and AS also designed the experiments section. AS implemented the code, ran the experiments and produced all the results and figures with input from JL. Appendix 2.E is not a part of the published work and it is written by AS to delineate his derivations of the analytical results on the (fully parameterized) model presented in the chapter and Chapter 3.

Chapter 3: This chapter appeared as Sheikh et al. (2014). It was mainly written by AS with continuous contributions from JL and JS. The extended analytical results presented in the chapter were derived by AS. All the authors worked on designing the experiments. AS implemented the code, ran the experiments and produced all the results, figures and tables (with help from Georgios Exarchakis to produce the results on publicly available sparse coding algorithms in Table 3.1) with feedback from JL and JS.

Chapter 4: I wrote this chapter. Figures 4.1 and 4.6 in the chapter are adapted from Shelton et al. (2011) and Goodfellow et al. (2013) respectively. I implemented the code, performed the experiments and produced all the results and figures presented in the chapter.

Chapter 5: This chapter is based on Shelton et al. (2015). The paper reproduced therein was written by Jacquelyn Shelton (JS) with contributions from Jörg Lücke (JL) and myself (AS). The experiments were conceived and designed by JS, AS, Jörg Bornschein (JB), Philip Sterne (PS) and JL. Data analysis was done by JS, AS, JB, PS. Code and analysis tools were implemented by PS, JB, JS, AS. The experiments were performed by JS, PS, JB, AS.

Chapter 6: I wrote the conclusions chapter.

Publications

Sheikh, A.-S., and Lücke J.; Select and sample for spike-and-slab sparse coding. *Advances in Neural Information Processing Systems 29.* (2016)

Shelton, J., Sheikh, A.-S., Bornschein, J., Sterne, P., and Lücke J.; Nonlinear spike-and-slab sparse coding for interpretable image encoding. *PLoS ONE.* (2015)

Sheikh, A.-S., Shelton, J., and Lücke J.; A Truncated EM Approach for Spike-and-Slab Sparse Coding. *Journal of Machine Learning Research.* (2014)

Shelton, J., Sterne, P., Bornschein, J., Sheikh, A.-S., and Lücke J.; Why MCA? Nonlinear sparse coding with spike-and-slab prior for neurally plausible image encoding. *Advances in Neural Information Processing Systems 25.* (2012)

Lücke J.,* and Sheikh, A.-S.*; A closed-form EM algorithm for sparse coding and its application to source separation. *International Conference on Latent Variable Analysis and Signal Separation.* (2012) * *joint first authorship*

Shelton, J., Bornschein, J., Sheikh, A.-S., Berkes, P., and Lücke J.; Select and sample — A model of efficient neural inference and learning. *Advances in Neural Information Processing Systems 24.* (2011)

Dai, Z., Shelton, J., Bornschein, J., Sheikh, A.-S., and Lücke J.; Combining approximate inference methods for efficient learning on large computer clusters. *NIPS Workshop on Big Learning.* (2011)

Bibliography

M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *Signal Processing, IEEE Transactions on*, 54(11): 4311–22, nov 2006.

S. Amari, A. Cichocki, and H. H. Yang. A new learning algorithm for blind signal separation. In *Advances in Neural Information Processing Systems*, pages 757–763, 1995.

C. Andrieu, N. Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43, 2003.

H. Attias. Independent factor analysis. *Neural Computation*, 11:803–851, 1999.

Y. Bengio. Learning deep architectures for ai. *Foundations and Trends® in Machine Learning*, 2(1):1–127, January 2009. ISSN 1935-8237.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, August 2013.

P. Berkes, G. Orban, M. Lengyel, and J. Fiser. Spontaneous Cortical Activity Reveals Hallmarks of an Optimal Internal Model of the Environment. *Science*, 331(6013):83–87, January 2011. ISSN 0036-8075.

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 on Big Learning*, 2010.

J. Bornschein, M. Henniges, and J. Lücke. Are v1 simple cells optimized for visual occlusions? A comparative study. *PLoS Computational Biology*, 9(6):e1003062, 2013.

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.

S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20:33–61, 1998.

A. Cichocki, S. Amari, K. Siwek, T. Tanaka, A.H. Phan, and R. Zdunek. ICALAB-MATLAB Toolbox Version 3, 2007.

A. Coates and A. Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 921–928, 2011.

A. Courville, J. Bergstra, and Y. Bengio. Unsupervised models of images by spike-and-slab RBMs. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 1145–1152, 2011.

Z. Dai and J. Lücke. Autonomous cleaning of corrupted scanned documents — a generative modeling approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3338–3345, 2012.

Z Dai, G. Exarchakis, and J. Lücke. What are the invariant occlusive components of image patches? a probabilistic generative approach. In *Advances in Neural Information Processing Systems 26*, pages 243–251. 2013.

R. C. Dalen and M. J. F. Gales. Covariance modelling for noise-robust speech recognition. In *Annual Conference of the International Speech Communication Association*, pages 2000–2003, 2008.

R. C. Dalen and M. J. F. Gales. Extended VTS for noise-robust speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 19(4), 2011.

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.

D. L. Donoho. Compressed sensing. *IEEE Trans. Inform. Theory*, 52:1289–1306, 2006.

M. Elad and M.l Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, 2006.

G. Exarchakis, M. Henniges, J. Eggert, and J. Lücke. Ternary sparse coding. In *Proceedings LVA/ICA*, LNCS. Springer, 2011. in press.

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

A. A. Frolov, D. Husek, and P. Y. Polyakov. Two expectation-maximization algorithms for Boolean factor analysis. *Neurocomputing*, 130:83–97, 2014.

J. V. Gael, Y. W. Teh, and Z. Ghahramani. The infinite factorial hidden Markov model. In *Advances in Neural Information Processing Systems*, pages 1697–1704, 2008.

P. Garrigues and B. A. Olshausen. Learning horizontal connections in a sparse coding model of natural images. In *Advances in Neural Information Processing Systems*, 2007.

I. Goodfellow, A. Courville, and Y. Bengio. Large-scale feature learning with spike-and-slab sparse coding. In *ICML*, 2012.

I. Goodfellow, A. Courville, and Y. Bengio. Scaling up spike-and-slab models for unsupervised feature learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1902–1914, 2013.

N. Guan, D. Tao, Z. Luo, and J. Shawe-Taylor. Mahnmf: Manhattan non-negative matrix factorization. *CoRR*, abs/1207.3438, 2012.

M. Haft, R. Hofman, and V. Tresp. Generative binary codes. *Pattern Anal Appl*, 6:269–284, 2004. ISSN 1433-7541. doi: 10.1007/s10044-003-0194-x.

M. Henniges, G. Puertas, J. Bornschein, J. Eggert, and J. Lücke. Binary Sparse Coding. In *Proceedings LVA/ICA*, LNCS 6365, pages 450–457. Springer, 2010.

M. Henniges, R. E. Turner, M. Sahani, J. Eggert, and J. Lücke. Efficient occlusive components analysis. *Journal of Machine Learning Research*, 15:2689–2722, 2014.

G. E. Hinton, S Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

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.

D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 1959.

A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13:411–430, 2000.

A. Ilin and H. Valpola. On the effect of the form of the posterior approximation in variational learning of ICA models. *Neural Processing Letters*, 22(2):183–204, 2005.

T. Jaakkola. Tutorial on variational approximation methods. In M. Opper and D. Saad, editors, *Advanced mean field methods: theory and practice.* MIT Press, 2000.

Y. Jernite, Y. Halpern, and D. Sontag. Discovering hidden variables in noisy-or networks using quartet tests. In *Advances in Neural Information Processing Systems 26*. MIT Press, 2013.

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.

K. Kavukcuoglu, M. Ranzato, and Y. LeCun. Fast inference in sparse coding algorithms with applications to object recognition. *CoRR*, abs/1010.3467, 2010a.

K. Kavukcuoglu, P. Sermanet, Y. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning convolutional feature hierachies for visual recognition. In *Advances in Neural Information Processing Systems (NIPS 2010)*, volume 23, 2010b.

D. Knowles and Z. Ghahramani. Infinite sparse factor analysis and infinite independent components analysis. In *Proceedings of International Conference on Independent Component Analysis and Signal Separation*, 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, 2011.

G. Kutyniok. Compressed sensing: Theory and applications. *CoRR*, abs/1203.3815, 2012.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–91, 1999.

H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*, pages 801–08, 2007.

H. Li and F. Liu. Image denoising via sparse and redundant representations over learned dictionaries in wavelet domain. In *Proceedings of International Conference on Image and Graphics*, 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*, 11:2855–2900, 2010.

J. Lücke and M. Sahani. Maximal causes for non-linear component extraction. *Journal of Machine Learning Research*, 9:1227–1267, 2008.

J. Lücke and A.-S. Sheikh. Closed-form EM for sparse coding and its application to source separation. In *Proceedings of International Conference on Latent Variable Analysis and Signal Separation*, pages 213–221, 2012.

D. J. C. MacKay. Local minima, symmetry-breaking, and model pruning in variational free energy minimization. Online publication: www.inference.phy.cam.ac.uk/mackay/minima.ps.gz, 2001.

J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *IEEE Transactions on Image Processing*, 17(1):53–69, 2008.

J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of International Conference on Machine Learning*, page 87, 2009a.

J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. *International Conference on Computer Vision*, 25, 2009b.

J Mairal, F Bach, J Ponce, and G Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11, 2010.

S. Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, 3rd edition, 2008.

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. Sparse exponential family latent variable models. NIPS Workshop, 2010.

S. Mohamed, K. Heller, and Z. Ghahramani. Evaluating Bayesian and L1 approaches for sparse unsupervised learning. In *Proceedings of International Conference on Machine Learning*, 2012.

E. Moulines, J.-F. Cardoso, and E. Gassiat. Maximum likelihood for blind separation and deconvolution of noisy signals using mixture models. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 5, pages 3617–3620, 1997.

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

G. J. Mysore, P. Smaragdis, and B. Raj. Non-negative hidden Markov modeling of audio with application to source separation. In *Proceedings of International Conference on Latent Variable Analysis and Signal Separation*, pages 140–148, 2010.

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.

R. M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56(1):71–113, July 1992. ISSN 0004-3702.

Radford M. Neal. Probabilistic inference using markov chain monte carlo methods doc. élec. Technical Report CRG-TR-93-1, University of Toronto, CA, 1993.

C. Niell and M. Stryker. Highly Selective Receptive Fields in Mouse Visual Cortex. *The Journal of Neuroscience*, 28(30):7520–7536, 2008.

B. Olshausen. Highly overcomplete sparse coding. In *Proc. SPIE, 8651, Human Vision and Electronic Imaging XVIII*, 2013.

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

B. Olshausen and D. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

B. Olshausen and K. Millman. Learning sparse codes with a mixture-of-Gaussians prior. In *Advances in Neural Information Processing Systems*, volume 12, pages 841–847, 2000.

M. Opper and O. Winther. Expectation consistent approximate inference, 2005.

J. Paisley and L. Carin. Nonparametric factor analysis with beta process priors. In *Proceedings of International Conference on Machine Learning*, 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.

R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 759–766. ACM, 2007.

M.'A. Ranzato and G. Hinton. Modeling pixel means and covariances using factorized third-order boltzmann machines. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2551–2558, 2010.

M. Rattray, O. Stegle, K. Sharp, and J. Winn. Inference algorithms and learning theory for Bayesian sparse factor analysis. *Journal of Physics: Conference Series*, 197:012002 (10pp), 2009.

F. Ribeiro and M. Opper. Expectation propagation with factorizing distributions: A gaussian approximation and performance results for simple models. *Neural Computation*, 23: 1047–1069, 2011.

Dario L. Ringach. Spatial structure and symmetry of simple-cell receptive fields in macaque primary visual cortex. *Journal of Neurophysiology*, 88:455 – 463, 2002.

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

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

M. Seeger. Bayesian inference and optimal design for the sparse linear model. *Journal of Machine Learning Research*, 9:759–813, 2008.

A.-S. Sheikh and J. Lücke. Select-and-sample for spike-and-slab sparse coding. *Advances in Neural Information Processing Systems*, 29, 2016.

A.-S. Sheikh, J. A. Shelton, and J. Lücke. A truncated EM approach for spike-and-slab sparse coding. *Journal of Machine Learning Research*, 15:2653–2687, 2014.

J. Shelton, J. Bornschein, A.-S. Sheikh, P. Berkes, and J. Lücke. Select and sample — a model of efficient neural inference and learning. In *Advances in Neural Information Processing Systems*, pages 2618–2626, 2011.

J. Shelton, P. Sterne, J. Bornschein, A.-S. Sheikh, and J. Lücke. Why MCA? nonlinear sparse coding with spike-and-slab prior for neurally plausible image encoding. *Advances in Neural Information Processing Systems*, 25, 2012.

J. A. Shelton, A.-S. Sheikh, J. Bornschein, P. Sterne, and J. Lücke. Nonlinear spike-and-slab sparse coding for interpretable image encoding. *PLoS ONE*, 10:e0124088, 05 2015.

T. Singliar and M. Hauskrecht. Noisy-or component analysis and its application to link analysis. *Journal of Machine Learning Research*, pages 2189–2213, 2006.

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.

X. Tan, J. Li, and P. Stoica. Efficient sparse Bayesian learning via Gibbs sampling. In *ICASSP*, pages 3634–3637, 2010.

Y. W. Teh, D. Görür, and Z. Ghahramani. Stick-breaking construction for the Indian buffet process. *Journal of Machine Learning Research*, 2:556–563, 2007.

R Tibshirani. Regression shrinkage and selection via the lasso. *J Royal Statistical Society. Series B*, 58(1):267–288, 1996.

M. Titsias and M. Lazaro-Gredilla. Spike and slab variational inference for multi-task and multiple kernel learning. In *Advances in Neural Information Processing Systems*, pages 2339–2347, 2011.

R. E. Turner and M. Sahani. Two problems with variational expectation maximisation for time-series models. In *Bayesian Time Series Models*. Cambridge University Press, 2011.

W. M. Usrey, M. P. Sceniak, and B. Chapman. Receptive Fields and Response Properties of Neurons in Layer 4 of Ferret Visual Cortex. *Journal of Neurophysiology*, 89:1003–1015, 2003.

H. Valpola, E. Oja, A. Ilin, A. Honkela, and J. Karhunen. Nonlinear blind source separation by variational Bayesian learning. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E86-A(3):532–541, 2003.

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.

M. J. Wainwright and M. I. Jordan. *Graphical models, exponential families, and variational inference*. Foundations and trends in machine learning. Now Publisher, 2008.

J. Wang, J. Yang, K. Yu, F. Lv, T. S. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 3360–3367, 2010.

M. West. Bayesian factor regression models in the "large p, small n" paradigm. *Bayesian Statistics*, pages 723–732, 2003.

F. Wood, T. L. Griffiths, and Z. Ghahramani. A non-parametric bayesian method for inferring hidden causes. In *Uncertainty in Artificial Intelligence*. AUAI Press, 2006.

J. Yang, K. Yu, Y. Gong, and T. S. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1794–1801, 2009.

J. Yang, K. Yu, and T. S. Huang. Efficient highly over-complete sparse coding using a mixture model. In *11th European Conference on Computer Vision (ECCV 2010)*, pages 113–126, 2010.

R. Yoshida and M. West. Bayesian learning in sparse graphical factor models via variational mean-field annealing. *Journal of the American Statistical Association*, 99:1771–1798, 2010.

M. Zhou, H. Chen, J. Paisley, L. Ren, G. Sapiro, and L. Carin. Non-parametric Bayesian dictionary learning for sparse image representations. In *Advances in Neural Information Processing Systems*, pages 2295–2303, 2009.

D. Zoran and Y. Weiss. "natural images, gaussian mixtures and dead leaves". In *Advances in Neural Information Processing Systems*, pages 1745–1753, 2012.