Large-scale Approximate EM-style Learning and Inference in Generative Graphical Models for Sparse Coding

vorgelegt von M.Sc. Informatik
Jacquelyn Ann Shelton
geboren in Flint, Michigan, USA

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 (Chair): Prof. Dr. Manfred Opper (TU Berlin)
Gutachter (Reviewer): Prof. Dr. Matthew B. Blaschko (KU Leuven)
Gutachter (Reviewer): Prof. Dr. Jörg Lücke (University of Oldenburg)
Gutachter (Reviewer): Prof. Dr. Klaus-Robert Müller (TU Berlin)

Tag der wissenschaftlichen Aussprache: 22. Mai 2018

Berlin 2018

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Doktorarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Doktorarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Ort, Datum Unterschrift

Acknowledgements

The work described in this thesis took place at the Frankfurt Institute for Advanced Science and at the Technical University of Berlin in their respective Machine Learning groups. I would like to thank the following people who left significant footprints: Jörg Lücke for supervising my PhD, for always finding creative solutions and ways to reach our goals as well as those of the research group, whether it be moving our group to greener, academically promising pastures or suggesting ways to improve productivity, such as by informing us that beer makes one more efficient for exactly one hour after consumption before the effects reverse. Arthur Gretton for always being there in any ways needed, from best friend to collaborator to academic advisor, despite his constant globe trotting, inabil Equus ferus caballus forever. Klaus-Robert Müller took me under his wing, was unconditionally understanding and supportive of my academic and personal challenges, and with unwaivering patience to boot. Matthew Blaschko has been encouraging since the beginning of my machine learning studies and has remained one of my strongest advocates - from supervising my Masters thesis to reviewing the current work of art - citing inspiration drawn from the Capra aegagrus hircus. Christoph Lampert, for his tough but inspiring support, an amazing internship in his group at IST Austria, and softening it all with ice skating in front of the Vienna Rathaus. Saboor Sheikh, for helping to make our PhD time fun as well as productive, despite continuous strange challenges, and for always offering a supportive ear or two. Jan Gasthaus, you always knew how to make me strive for 150% whether I liked it or not. Heiko Strathmann, for proof-reading and revising my German abstract. Tara Thackeray, when things got rough and the pressure unbearable, you helped me cope and refocus. Finally, thanks to my parents for their unconditional support and for mailing me cake.

Abstract

We propose a nonparametric procedure to achieve fast inference in generative graphical models when the number of latent states is very large. The approach is based on iterative latent variable preselection, where we alternate between learning a 'selection function' to reveal the relevant latent variables, and using this to obtain a compact approximation of the posterior distribution for EM; this can make inference possible where the number of possible latent states is e.g. exponential in the number of latent variables, whereas an exact approach would be computationally infeasible. To increase the efficiency of our approach, we can draw samples from the compact approximate posterior distribution and compute the parameters in the M-step as usual. We refer to the procedure combining these two approximation methods as *Select and Sample*.

In numerical experiments on artificial data and image patches, we compare the performance of the algorithms to the performance of exact EM, latent variable preselection alone, sampling alone, and the combination of the two for the Select and Sample approach. For this Sparse Coding example we show the effectiveness and efficiency: it enables applications easily exceeding a thousand observed and a thousand hidden dimensions.

To apply the Select and Sample approach to a more complex model, we propose a novel, complex Sparse Coding model that targets low-level image structures, such as edges and their occlusions. The model uses a Spike-and-Slab prior distribution and has a nonlinearity in the data likelihood, both of which lead to a highly multimodal posterior distribution and computational/analytical intractabilities. We refer to this model as *SSMCA*. For adequate representation of the complex posterior, we develop an exact Gibbs sampler based on the exact form of the posterior distribution. Results on artificial and natural images show that SSMCA can model the generating process of images with occlusions, including extracting individual edge-like structures that occlude each other, and produce results that are neurally consistent with in vivo neural recordings and with the model's prior beliefs.

We learn the selection function entirely from the observed data and current EM state via Gaussian process regression, calling this method *GP-select*. This is by contrast with earlier approaches, where selection functions were manually-designed for each problem setting. We show that our approach performs as well as these bespoke selection functions on a wide variety of inference problems: in particular, for the challenging case of a hierarchical model for object localization with occlusion, we achieve results that match a customized state-of-the-art selection method, at a far lower computational cost.

Zusammenfassung

Wir beschreiben ein nichtparametrisches Verfahren für schnelle Inferenz in generativen graphischen Modellen mit extrem großen latenten Zustandsräumen. Das Verfahren basiert auf iterativer Selektion der latenten Variablen. Zunächst wird eine 'Selektionsfunktion' zur Aufdeckung von relevanten latenten Zuständen gelernt, welche im nächsten Schritt für eine kompakte Approximierung der a-posteriori Verteilung im EM Algorithmus verwendet wird. Dies ermöglicht Inferenz wenn die Anzahl der latenten Zustände exponentiell mit der Anzahl der latenten Variablen wächst. Für verbesserte Effizienz, können wir Stichproben von der a-posteriori Verteilung benutzen um damit im M-Schritt die Parameter stochastisch zu approximieren. Wir nennen den vorgestellten Algorithmus *'Select and Sample'*.

Wir vergleichen unseren Algorithmus mit den folgenden EM Varianten auf künstlichen Daten und Bildausschnitten: exakter EM, nur Vorselektion der latenten Variablen, nur stochastische Approximation, und die Kombination der beiden Approximationen des 'Select and Sample' Verfahrens. Unser Beispiel belegt die Effizienz unserer Methode, die Inferenz mit tausenden Datenpunkten und latenten Dimensionen ermöglicht.

Um 'Select and Sample' auf ein komplizierteres Modell anzuwenden, entwickeln wir ein neues complexes 'Sparse Coding' Modell, welches auf gründsätzlichen Bildeigenschaften wie z.B. Kanten und deren überlagerung basiert. Unser Modell, genannt *'SSMCA'*, nutzt eine 'Spike-and-Slab' a-priori Verteilung und eine nicht-lineare Likelihoodfunktion, was in einer hochdimensionalen, multi-modalen a-posteriori Verteilung führt. Um die komplexe a-posteriori Verteilung genau vergleichen zu können entwickeln wir eine 'Gibbs Sampling' Methode, welche genaue Form der Verteilung verwendet. Unsere Ergebnisse auf künstlichen und natürlichen Bildern zeigen, dass 'SSMCA' den generativen Prozess der Bildern mit überlagerungen abbilden kann: Sowohl Kantenstrukturen als auch überlagerungen können extrahiert werden und sind neuronal plausibel.

Wir lernen die Selektionsfunktion ausschliesslich von den beobachteten Daten und dem aktuellen EM Zustand mittels Gauss Prozessen. Wir nennen diese Methode *'GP-Select'*. Dies kontrastiert frühere Verfahren, bei denen die Selektionsfunktion per Hand für jedes Problem neu entwickelt werden musste. Wir zeigen, dass unsere Methode Ergebnisse produziert, die genauso gut wie die von Hand entwickelten Selektionsfunktionen sind – auf einer Vielfalt von Inferenzproblemen. Unsere Ergebnisse im herausfordernden Fall von hierarchischen Modellen für Objektlokalisierung mit Uberlagerungen sind en-par mit einer speziell angepassten modernsten Selektionsmethode, bei deutlich reduzierter Rechenszeit.

Contents

Chapter 1

Introduction

We have entered the era of big data - there is a wealth of data available from countless sources, from webpages to YouTube videos to images, and so on. In order to understand, learn, and extract patterns from this huge plethora of data, we need automated methods of data analysis. Machine learning is an interdisciplinary field that offers such methods – it focuses on mathematical foundations and practical applications of systems that can find structure, predict new outcomes, and choose actions wisely. Thus we define machine learning to be a set of methods capable of automatically detecting patterns in data, using the uncovered patterns to predict future data, or to make decisions under uncertainty (even the decision to collect more data).

The probabilistic approach to machine learning uses the tools of probability theory to solve problems involving uncertainty. This approach allows us to quantify and make use of this information instead of regarding it as a nuisance to solving the problem. Uncertainty can come in many forms - each observation/data point provides information, but we are never completely certain about e.g. what can some past data predict about the future, or what the best model is to explain some data, and so on. This is taken into consideration when forming probabilistic models of data. Figure 1.1 shows an illustration of a very simple probabilistic graphical model - the observed data is denoted by y which are conditioned on the (latent) variables denoted by s. In the latent variable model setting, we would only observe y from which the values of latent variables s could be inferred. Following Bayesian reasoning, we denote a prior distribution over the latent variables s with $p(s)$ and the likelihood of the data with $p(y|s)$. Furthermore, this can also be treated as a generative model, as the process that generated the observations $p(y|s)$ can be modelled, necessitating the modelling of the joint distribution of both the latent and observed variables $p(y, s)$.

This body of work focuses on probabilistic modelling and inference in generative graph-

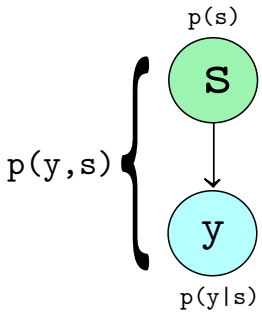


Figure 1.1: *Illustration of a simple graphical model, where the node with s is data modelled by the probability distribution $p(s)$ and the node with y represents the data observed given* s *for the data likelihood $p(y|s)$. In the case of a latent variable model,* s *is not observed. This can depict a generative model if the goal is to model the generation process of the observations $p(y|s)$, for which the joint distribution of both the latent and observed variables $p(y, s)$ needs to be modelled.*

ical models with a large number of hidden variables, each of which may take on one of several state values. Without adequate approximations, inference in these models becomes computationally (and sometimes also analytically) intractable, thus in this thesis we develop methods to efficiently overcome these challenges.

1.1 Problem Setting

When hidden variables are present, learning of parameters in probabilistic graphical models can quickly become intractable as the dimensionality of hidden states increases. Consider, for instance, the floor of a nursery populated with different toys, and images of this floor large enough to contain a number of toys. A nursery easily contains a hundred different toys and any subset of these hundred toys may appear in any image. For one hundred toys there is therefore a combinatorics of 2^{100} different combinations of toys that can make up an image. An inference task may now be to infer, for any given image, the toys it contains. If we approached this task using a probabilistic graphical model, we would define a basic such model using a set of one hundred hidden variables (one for each toy). Given a specific image, inference would then take the form of computing the posterior probability for any combination of toys, and from this, for example, the probability of each toy to be in the image can be computed. If done exactly, this process needs to evaluate all the 2^{100} different toy combinations which easily exceeds currently available computational resources.

While there are also many tasks for which graphical models with few latent variables are sufficient, the requirement for many hidden variables (as in the toy example) is typical for visual, auditory and many other types of data with very rich structure. Graphical models for such data are often a central building block for tasks such as denoising (Elad and Aharon, 2006; Lázaro-gredilla and Titsias, 2011), inpainting (Mairal et al., 2009b,a; Lázaro-gredilla and Titsias, 2011), classification (Raina et al., 2007), or collaborative

filtering (Lázaro-gredilla and Titsias, 2011). Typically, the performance in these tasks improves with the number of latent variables that can be used (and which is usually limited by computational demands).

1.2 Background

We first formulate a probabilistic model of the data. Denote the data set of size N as $Y = \{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(N)}\}$ where a single observed data point (e.g. an image of toys, in the above example) is denoted $\mathbf{y}^{(n)} = (y_1, \ldots, y_D)$, and the set of latent variables is denoted $S = \{\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(N)}\}$ with $\mathbf{s}^{(n)} = (s_1, \ldots, s_H)$, where there are D observed variables (e.g. pixels) and H latent variables (e.g. toys), respectively. We denote the prior distribution over the latent variables as $p(S|\theta)$ and the likelihood of the data as $p(Y|S, \theta)$. Using these expressions the data distribution can then be modelled by a generative data model for the data likelihood:

$$p(Y \mid \Theta) = \sum_S p(Y \mid S, \Theta)\, p(S \mid \Theta) \tag{1.1}$$

with Θ denoting the parameters of the model. Equation (1.1) assumes discrete latent variables, but in the case of continuous variables the sum is replaced by an integral: $p(Y \mid \Theta) = \int_S p(Y \mid S, \Theta)\, p(S \mid \Theta)$. For a hierarchical model, the prior distribution $p(S \mid \Theta)$ may be subdivided hierarchically into different sets of variables.

Considering a single data point, the posterior distribution over latent variables is defined as follows:

$$p(\mathbf{s}|\mathbf{y}^{(n)}, \Theta) = \frac{p(\mathbf{s}|\Theta)\, p(\mathbf{y}^{(n)}|\mathbf{s}, \Theta)}{\sum_{\mathbf{s}'} p(\mathbf{s}'|\Theta)\, p(\mathbf{y}^{(n)}|\mathbf{s}', \Theta)} \tag{1.2}$$

To find the optimal parameters Θ, we must solve the following optimization problem: given data set Y find maximum likelihood parameters Θ^*:

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}}\ p(Y \mid \Theta) \tag{1.3}$$

Solving the optimization problem in Equation (1.3) requires marginalizing over all H latent variables in $\mathbf{s}$, as shown in the denominator of the posterior distribution in Equation (1.2). This step quickly becomes computationally intractable as the number of latent variables increases.

Expectation Maximization (EM) is a standard procedure widely applied to learn the maximum likelihood model parameters given graphical model when hidden variables are present (see e.g. (Dempster et al., 1977; Neal and Hinton, 1998)). EM is an algorithm

that iteratively optimizes a lower bound of the data likelihood, called the *free energy*, by inferring the posterior distribution over hidden variables given the current parameters (the E-step), and then adjusting the parameters to maximize the likelihood of the data averaged over this posterior (the M-step). This process can be formalized as follows:

Maximize objective function $\mathcal{L}(\Theta) = \log p(Y \mid \Theta)$ with respect to Θ by optimizing the free energy $\mathcal{F}$:

$$\mathcal{L}(\Theta) \geq \mathcal{F}(\Theta, q) = \sum_s q(S|\Theta) \log \frac{p(Y, S|\Theta)}{p(S|\Theta)} \tag{1.4}$$

$$= \langle \log p(Y, S) \rangle_{q(S|\Theta)} + \mathrm{H}[q(S)] \tag{1.5}$$

where q is a distribution over the hidden variables used to obtain the lower bound on the log likelihood $\mathcal{L}(\Theta)$ and $H[q]$ is the entropy of $q(S)$. For simplification we now temporarily drop the index to the nth data point.

Using EM, iteratively optimize $\mathcal{F}(\Theta, q)$:

E-Step: compute posterior distribution q, parameters fixed

$$\underset{q(\mathbf{s}|\Theta)}{\operatorname{argmax}} \mathcal{F}(\Theta, q) \rightarrow q(\mathbf{s}|\Theta) := p(\mathbf{s}|\mathbf{y}, \Theta) \tag{1.6}$$

M-Step: estimate model parameters, q fixed

$$\underset{\Theta}{\operatorname{argmax}} \mathcal{F}(\Theta, q) \rightarrow \Theta := \underset{\Theta}{\operatorname{argmax}} \langle \log p(\mathbf{y}, \mathbf{s}) \rangle_{q(\mathbf{s}|\Theta)} \tag{1.7}$$

M-step is only concerned with the expected log likelihood as the entropy of $q(\mathbf{s})$ does not depend directly on Θ. The free energy can be rewritten as $\mathcal{F}(\Theta, q) = \mathcal{L}(\Theta) - \mathbf{KL}[\, q(\mathbf{s}) \,||\, p(\mathbf{s}|\mathbf{y}, \Theta)\,]$, where the second term is the Kullback-Leibler divergence. In other words, for fixed Θ, $\mathcal{F}(\Theta, q)$ is bounded above by the log likelihood, which is only fulfilled when $\mathbf{KL}[\, q(\mathbf{s}) \,||\, p(\mathbf{s}|\mathbf{y}, \Theta)\,] = 0$, or rather, when $q(\mathbf{s}) = p(\mathbf{s}|\mathbf{y}, \Theta)$. Thus, the E-step simply sets $q(\mathbf{s})$ to be equal to the current computation of the posterior distribution $p(\mathbf{s}|\mathbf{y}, \Theta)$.

The M-step updates typically depend only on a small number of expectations with respect to the posterior distribution as given by

$$\langle g(\mathbf{s}) \rangle_{p(\mathbf{s} \,|\, \mathbf{y}^{\,(n)}, \Theta)} = \sum_{\mathbf{s}} p(\mathbf{s} \,|\, \mathbf{y}^{\,(n)}, \Theta)\, g(\mathbf{s})\,, \tag{1.8}$$

where $g(\mathbf{s})$ is usually an elementary function of the hidden variables, namely, the sufficient statistics under $p(\mathbf{s} \,|\, \mathbf{y}^{\,(n)}, \Theta)$. The sum is replaced by an integral for continuous latents. Calculating the expectations (1.8) is the computationally demanding part of EM optimization for any non-trivial generative model as the posterior distribution computed in the E-step (1.6) can be very complex and high-dimensional. The exact computation

of the expectations is therefore often intractable and many well-known algorithms (e.g. Olshausen and Field, 1996; Lee et al., 2007) have to rely on approximations. Especially when the number of latent states to consider is large (e.g. exponential in the number of latent variables), computing the posterior distribution becomes intractable, rendering approximations unavoidable.

A wide variety of approximate inference methods exist, all with their own advantages and disadvantages, and the area continues to be a hugely active field of research. We describe some of the more popular approaches. The Laplace approximation method makes a Gaussian approximation to the posterior distribution (Laplace, 1774) (see e.g. (Murphy, 2012) Section 8.4.1 for an illustration). This method is simple, efficient and can be a reasonable approximation when sample sizes are large, since posteriors often become more "Gaussian-like", for reasons analogous to the central limit theorem. However, it is not gauranteed to converge. The posterior can also be approximated with stochastic methods such as Markov chain Monte Carlo methods (MCMC) (see (Neal, 1993) for a review). MCMC methods draw samples from the posterior distribution and can be used to e.g. compute the expected sufficient statistics under the posterior in Equaton (1.8). These methods are guaranteed to converge to the true posterior distribution in the limit, even when the posterior is high-dimensional and/or multimodal. However, many samples are required to ensure accuracy, thus their ability to represent complex posterior distributions is limited by available computational resources. Additionally, there is often high variance in the estimated integrals/summations and it is sometimes difficult to assess convergence. Another way to approximate the posterior distribution is by using variational approximations (see (Wainwright and Jordan, 2003) for a review) where the objective, based directly on the form of the free-energy function (1.4), is to approximate $q(\mathbf{s})$ such that the KL-divergence between it and the true posterior, $\mathbf{KL}[\, q(\mathbf{s}) \,||\, p(\mathbf{s}|\mathbf{y}, \Theta) \,]$, is minimized. These methods are efficient and easy to compute but produce biased estimates. They often use a factorial prior over the latent variables which results in a factored posterior distribution, however this is a generally unrealistic assumption and neglects correlations between latents (explaining away). Instead, in this work we focus on an approximate inference approach that can represent complex posterior distributions without ignoring correlations between latent variables (i.e. can avoid the effects of explaining away), has the ability to represent multiple mode in the posteriors, and is efficiently computable.

Expectation truncation (ET) is an approximate EM algorithm for accelerating inference and learning in graphical models with many latent variables (Lücke and Eggert, 2010). Its basic idea is to restrict the inference performed during the E-step to an "interesting" subset of states of the latent variables, chosen per data point according to a *selection function*. This subspace reduction can lead to a significant decrease in computational demand with very little loss of accuracy (compared with the full model). This approach is

inherently suited for problems with sparse latent variables as it narrows in on the areas of the posterior where the variables in the 'restricted set' have the greatest probability mass. To provide an intuition: For the toy example introduced earlier, we could for instance first analyze the colors contained in a given image. If the image did not contain the color "red", we could already assume red toys or partly red toys to be absent. Only in a second step would we then consider the combinatorics of the remaining toys. More features and more refined features would allow for a reduction to still smaller sets of toys until the combinatorics of these selected toys becomes computationally tractable. The selection function of ET mathematically models the process of selecting the relevant hidden variables (the relevant toys); while truncated posterior distributions then models their remaining combinatorics (see further below). Using ET, the posterior distribution in Equation (1.2) can then be approximated by a posterior distribution with support truncated to the preselected latent variables:

$$\begin{aligned} & p(\mathbf{s}|\mathbf{y}^{(n)}, \Theta) \\ & \quad \approx q_n(\mathbf{s}; \Theta) = \frac{p(\mathbf{s}, \mathbf{y}^{(n)}|\,\Theta)\, \mathbb{I}(\mathbf{s} \in \ \mathcal{K}_n)}{\sum\limits_{\mathbf{s}'\in\mathcal{K}_n} p(\mathbf{s}', \mathbf{y}^{(n)}|\,\Theta)}, \end{aligned} \tag{1.9}$$

where $\mathcal{K}_n$ contains the latent states of the relevant variables for data point $\mathbf{y}^{(n)}$, and $\mathbb{I}(\mathbf{s} \in \mathcal{K}_n) = 1$ if $\mathbf{s} \in \mathcal{K}_n$ and 0 otherwise. In other words, Equation (1.9) is proportional to Equation (1.2) if $s \in \mathcal{K}_n$ (and zero otherwise). The set $\mathcal{K}_n$ contains only states for which $s_h = 0$ for all h that are not selected, i.e. all states where $s_h = 1$ for non-selected h are assigned zero probability. This means that there are fewer terms in the denominator of the truncated posterior in Equation (1.9) compared with that of full posterior in Equation (1.2), which affects the overall scaling of the terms. The truncated posterior (1.9) still remains proportional to the full posterior (1.2) for the selected terms $s \in K_n$, however. The sum over $\mathcal{K}_n$ is much more efficient than the sum for the full posterior, since it need only be computed over the reduced set of latent variable states deemed relevant: the state configurations of the irrelevant variables are fixed to be zero. The variable selection parameter is selected based on the compute resources available: i.e. as large as resources allow in order to be closer to true EM, although empirically it has been shown that much smaller values suffice with very little loss of accuracy (see e.g. Sheikh et al., 2014, Appendix B on complexity-accuracy trade-offs).

The approximation of the full posterior distribution (1.9) can be used to compute efficiently the expectations needed in the M-step (1.7):

$$\langle g(\mathbf{s})\rangle_{p(\mathbf{s}\,|\,\mathbf{y}^{\,(n)},\Theta)} \approx \langle g(\mathbf{s})\rangle_{q^{(n)}(\mathbf{s};\Theta)} = \frac{\sum_{\mathbf{s}\in\mathcal{K}_n} p(\mathbf{s}, \mathbf{y}^{\,(n)}\,|\,\Theta)\, g(\mathbf{s})}{\sum_{\mathbf{s}'\in\mathcal{K}_n} p(\mathbf{s}', \mathbf{y}^{\,(n)}\,|\,\Theta)}\,. \tag{1.10}$$

Equation (1.10) represents a reduction in required computational resources as it involves only summations (or integrations) over the smaller state space $\mathcal{K}_n$. The bottleneck here is that the set $\mathcal{K}_n$ needs to be selected prior to the computation of expectations, and the final computational acceleration relies on such selections being efficiently computable and well-suited to the model under consideration.

A *selection function* $\mathcal{S}_h(\mathbf{y}, \Theta)$ is used to identify a subset of salient variables from H, denoted by H' where $H' \ll H$. This is in turn used to define a subset, denoted $\mathcal{K}_n$, of the possible state configurations of the space per data point $\mathbf{y}$. State configurations not in this space (of variables deemed to be non-relevant) are fixed to 0 (assigned zero probability mass). $\mathcal{K}_n$ can be formally defined as follows:

$$\mathcal{K}_n = \{\mathbf{s} \,|\, \text{for all } h \notin \mathcal{I} : \; s_h = 0\}\,, \tag{1.11}$$

where $\mathcal{I}$ contains the H' indices h with the highest values of a selection function $\mathcal{S}_h(\mathbf{y}, \Theta)$. Appropriate selection functions $\mathcal{S}_h(\mathbf{y}, \Theta)$, for e.g. Sparse Coding models, can be found by deriving efficiently computable upper-bounds for probabilities $p(s_h = 1 \,|\, \mathbf{y}^{(n)}, \Theta)$ (Puertas et al., 2010) or via deterministic relations $\mathbf{s} = f(\mathbf{y}, \Theta)$ in the limit of no data noise (Henniges et al., 2010). Usually however, the selection functions are derived by hand for each individual model for which ET is using them, requiring expertise in the problem domain. We can expect the approximation to be accurate only if restricting the combinatorics (e.g. combinations of a restricted number of toys) does not miss large parts of posterior mass. The larger $\mathcal{K}_n$ the less precise the selection has to be, but with the obvious trade-off of necessary compute resources. For $\mathcal{K}_n$ equal to the entire state space, no selection is required and the approximations (1.9) and (1.10) fall back to the case of exact inference.

1.3 Roadmap

In Chapter 2, we introduce an approach that combines latent variable preselection (via ET) with MCMC sampling methods for the acceleration of inference and learning with EM. The idea is to combine the strengths of each approach in order to represent complex posterior distributions while also reducing the necessary computations to do so. Incorporating a sampling method with ET allows for the representation of multiple modes and arbitrary correlations within the posterior regions that have already been preselected to the reduced state space with highest probability mass. Generally a huge amount of samples is necessary to adequately represent complex distributions – they may be very correlated (because of explaining away effects), multimodal (multiple possible interpretations), and very high-dimensional. As our method only draws samples from the more

compact posterior distribution truncated to the 'interesting' variables instead of the full complex distribution, this main drawback of sampling methods is minimized. We refer to our combined approach as *Select and Sample*.

Select and Sample can be interpreted as neurally plausible. Inference and learning in neural circuits can be regarded as the task of inferring the true hidden causes of a stimulus. An example is inferring the objects in a visual scene based on the image projected on the retina. Latent variable preselection connects sampling to very influential models of neural processing that emphasize feed-forward processing ((Rosenblatt, 1958; Riesenhuber and Poggio, 1999) and many more), and is consistent with the popular view of neural processing and learning as an interplay between feed-forward and recurrent stages of processing (Yuille and Kersten, 2006; Hinton et al., 1995; Körner et al., 1999; Lee and Mumford, 2003a). Our combined approach encompasses these views of processing in neural circuits naturally by interpreting feed-forward selection and sampling as approximations to exact inference in a probabilistic framework for perception.

We demonstrate the effectiveness and efficiency of the Select and Sample approach on a vanilla Sparse Coding model optimized with EM. In numerical experiments on artificial data and image patches, we compare the performance of Sparse Coding with posterior distributions in the E-step computed using the following: the full posterior for exact EM (tractable with simpler data), latent variable preselection alone, a Gibbs sampling method (details in Chapter 2), and the combination of the two (Select and Sample). Our approach performs well in applications exceeding a thousand observed and a thousand hidden variables.

In Chapter 3, we apply the Select and Sample approach introduced in Chapter 2 to a novel and complex Sparse Coding model. 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 modeled 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 nonlinearity (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 an important goal of this Chapter is to isolate and highlight differences between them. Furthermore, parameter optimization is analytically and computationally intractable in our model. Thus another core contribution of this work is the development of an exact Gibbs sampler in order to adequately

sample the complex posterior distribution resulting from the nonlinearity in the data likelihood. This approach allows for efficient inference, but by first preselecting the most "interesting" latent variables (as in Select and Sample), we can speed up the inference process as well as handle higher dimensional data.

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. Finally, experiments on natural image patches, show neural consistency of our model in two ways: its predictions are consistent with (1) *in vivo* neural recordings and with (2) the model's prior beliefs.

In Chapter 4, we generalize the preselection optimization method (introduced in Chapter 2 and applied in Chapter 3) in a model-independent nonparametric black-box way for further acceleration of EM. The definition of appropriate selection functions for basic graphical models (such as the nursery floor example discussed above) is already nontrivial. For models incorporating more detailed data properties, the definition of selections functions becomes still more demanding. For visual data, e.g. models that also capture mutual object occlusions (Henniges et al., 2014) and/or the object position (Dai and Lücke, 2014), the design of suitable selection functions is extremely challenging: it requires both expert knowledge on the problem domain and considerable computational resources to implement (indeed, the design of such functions for particular problems has been a major contribution in previous work on the topic).

We propose a generalization of the ET approach, where we completely avoid the challenge of problem-specific selection function design. Instead, we learn selection functions adaptively and nonparametrically from the data, while learning the model parameters simultaneously using EM. We emphasize that the selection function is used only to "guide" the underlying base inference algorithm to regions of high posterior probability, but is not itself used as an approximation to the posterior distribution. As such, the learned function does not have to be a completely accurate indication of latent variable predictivity as long as the *relative importance* of the latent states likely to contribute posterior probability mass is preserved. We use Gaussian process regression (Rasmussen and Williams, 2005) to learn the selection function – by regressing the expected values of the latent variables onto the observed data – though other regression techniques could also be applied. The main advantage of GPs is that they do not need to be re-trained when only the output changes, as long as the inputs remain the same. This makes adaptive learning of a changing target function (given fixed inputs) computationally trivial (this will become clear in

Section 4.4). We term this part of our approach *GP-select*. Our nonparametric generalization of ET may be applied as a black-box meta algorithm for accelerating inference in generative graphical models, with no expert knowledge required.

Our approach is the first to make ET a general purpose algorithm for discrete latent variables,whereas previously, ET had to be modified by hand for each latent variable model addressed. For instance, we will show that preselection is crucial for efficient inference in complex models. Although ET has already been successful in some models, this work shows that more complex models will crucially depend on an improved selection step and focuses on automating this step.

For empirical evaluation, we have applied GP-select in a number of experimental settings. First, we considered the case of Sparse Coding models (binary Sparse Coding, spike-and-slab, nonlinear spike-and-slab), where the relationship between the observed and latent variables is known to be complex and nonlinear.[1] We show that GP-select can produce results with equal performance to the respective manually-derived selection functions. Interestingly, we find it can be essential to use nonlinear GP regression in the spike-and-slab case, and that simple linear regression is not sufficiently flexible in modelling the posterior shape. Second, we illustrate GP-select on a simple Gaussian mixture model, where we can provide intuition and explicitly visualize the form of the learned regression function. We find that even for a simple model, it can be be essential to learn a nonlinear mapping. Finally, we present results for a recent hierarchical model for translation invariant occlusive components analysis (Dai and Lücke, 2014). The performance of our inference algorithm matches that of the complex hand-engineered selection function of the previous work, while being straightforward to implement and having a far lower computational cost.

In Chapter 5 we summarize and review the overall conclusions and contributions of this body of work and discuss potential future directions.

1.4 Summary of Main Chapters

The Roadmap provided a comprehensive outline of the thesis in which the three chapter containing the main content were introduced. A condensed summary of these chapters is as follows:

[1] Note that even when linear relations exist between the latents and outputs, a nonlinear regression may still be necessary in finding relevant variables, as a result of explaining away.

- **Chapter 2 (Efficient Inference with 'Select and Sample'):** In this chapter we introduce an inference method, 'Select and Sample', that combines latent variable preselection with Markov Chain Monte Carlo (MCMC) sampling methods for the acceleration of EM. We demonstrate the effectiveness of our approach by considering a simple Sparse Coding model parameterized with a binary prior distribution over the latent variables and a Gaussian distribution over the observed variables.
- **Chapter 3 (Nonlinear Spike-and-Slab Sparse Coding):** In this chapter we introduce a novel Sparse Coding model with a Spike-and-Slab prior distribution and a nonlinearity in the data likelihood. This leads to a highly multimodal posterior distribution and computational/analytical intractabilities. We apply the Select and Sample approach of Chapter 2 to this model, which due to the complex posterior, requires the development of new MCMC method.
- **Chapter 4 (Generalizing Subspace Preselection with GP-select):** In this chapter we introduce a nonparametric generalization of Select and Sample. In the previous chapters, the 'selection' step of the method used a manually-designed function to learn a mapping from the observed data to the current EM state for each problem setting. Instead here we use Gaussian process regression to learn such a function automatically and with increased efficiency. We demonstrate the effectiveness and efficiency of this approach on a wide variety of inference problems, including a hierarchical model for object localization with occlusion.

1.5 Scientific Contributions

The work in this thesis makes several contributions. The peer-reviewed articles corresponding to these contributions are cited at the beginning of the relevant Chapter. The following are the main contributions:

- **An approach for efficient approximate inference in generative graphical models (Select and Sample)**: We propose the fundamental approximate inference approach for accelerated inference and learning, which was built upon and applied in the subsequent Chapters. This approach combines latent variable preselection with Markov Chain Monte Carlo (MCMC) sampling methods for the acceleration of inference and learning with EM. The preselection of the relevant latent variables is done with a *selection function*. This allows us to capture the strengths of each approach in representing complex posterior distributions and simultaneously reduce computational costs of inference with little to no loss of accuracy.
- **A nonlinear probabilistic model of occlusions in images (SSMCA)**: We intro-

duce a complex and novel Sparse Coding model designed to model low-level image components (such as edge-like structures and their occlusions). The model used a complex prior distribution (Spike-and-slab) – to model the presence/absence of e.g. an edge as well as its pixel intensity – and has a nonlinearity in the data likelihood (the nonlinear max combination rule) to target occlusions, i.e. dictionary elements correspond to image components that can occlude each other. We apply Select and Sample to this complex Sparse Coding model.

- **An exact MCMC method for inference in the proposed nonlinear Sparse Coding model**: In order to do inference in the SSMCA model, we develop an exact Gibbs sampler constructed exactly after the nonlinearity in the data likelihood of the SSMCA model. The nonlinearity in the data likelihood leads to a highly multimodal complex posterior distribution, thus in order to adequately sample this distribution we develop an exact Gibbs sampler based on the exact form of the posterior distribution. This allows for the successful application of Select and Sample to a complex Sparse Coding model. Furthermore, our results show that SSMCA can model the generating process of images with occlusions, including extracting individual edge-like structures that occlude each other, and makes predictions that are neurally consistent.
- **The generalization of the variable preselection process (GP-select)**: We propose a model-independent nonparametric black-box way to define a suitable selection function efficiently. Namely, we learned the selection function entirely from the observed data and current EM state using Gaussian process regression. Empirical experiments show equivalent performance between our inference algorithm (using GP-select to preselect variable) and algorithms of previous work (using a complex hand-engineered selection function for preselection). At the same time, GP-select is straightforward to implement and has a far lower computational cost.

1.6 Publications

Many aspects of this thesis have been published previously in peer-reviewed venues:

- **Shelton, J.**, Gasthaus, J., Dai, Z., Lücke, J., Gretton, A., 2017. GP-select: Accelerating EM using adaptive subspace preselection. Neural Computation 29 (8), 2177–2202.
- **Shelton, J.**, Sheikh, A.-S., Bornschein, J., Sterne, P., Lücke, J., 2015. Nonlinear Spike-and-Slab Sparse Coding for Interpretable Image Encoding. PLOS ONE 10 (5), 1–25.
- **Shelton, J.**, Gasthaus, J., Dai, Z., Lücke, J., Gretton, A., 2014. GP-select: Accelerating EM using adaptive subspace preselection. Women in Machine Learning Workshop (WiML 2014) in conjunction with NIPS, Montreal, Quebec.
- Lücke, J., **Shelton, J.**, Bornschein, J., Sterne, P., Berkes, P., Sheikh, A.-S., 2013. Combining Feed-Forward Processing and Sampling for Neurally Plausible Encoding Models. Computational and Systems Neuroscience 12.
- **Shelton, J.**, Sheikh, A.-S., Sterne, P., Bornschein, J., Lücke, J., 2013. Nonlinear Spike-and-Slab Sparse Coding for Interpretable Image Encoding. NIPS Workshop on High-dimensional Statistical Inference in the Brain.
- **Shelton, J.**, Sterne, P., Bornschein, J., Sheikh, A.-S., Lücke, J., 2012. Why MCA? Nonlinear Spike-and-Slab Sparse Coding with Spike-and-Slab Prior for Neurally Plausible Image Encoding. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (Eds.), Advances in Neural Information Processing Systems 25. pp. 2285–2293.
- **Shelton, J.**, Sterne, P., Bornschein, J., Sheikh, A.-S., Lücke, J., 2012. Why MCA? Nonlinear Spike-and-Slab Sparse Coding with Spike-and-Slab Prior for Neurally Plausible Image Encoding. Women in Machine Learning Workshop (WiML 2012) in conjunction with NIPS, Lake Tahoe, Nevada.
- **Shelton, J.**, Bornschein, J., Sheikh, A.-S., Berkes, P., Lücke, J., 2011. Select and Sample - A Model of Efficient Neural Inference and Learning. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. (Eds.), Advances in Neural Information Processing Systems 24. pp. 2618–2626.
- Dai, Z., **Shelton, J.**, Bornschein, J., Sheikh, A.-S., Lücke, J., 2011. Combining Approximate Inference Methods for Efficient Learning on Large Computer Clusters. NIPS workshop on Big Learning: Algorithms, Systems, and Tools for Learning at Scale.

- **Shelton, J.**, Bornschein, J., Sheikh, A.-S., Berkes, P., Lücke, J., 2011. Select and Sample - A Model of Efficient Neural Inference and Learning. Women in Machine Learning Workshop (WiML 2011) in conjunction with NIPS, Malaga, Spain.

In addition, here are peer-reviewed works I was involved with not appearing in the thesis:

- Sheikh, A-S., **Shelton, J.**, Lücke, J., 2014. A Truncated EM Approach for Spike-and-Slab Sparse Coding. Journal of Machine Learning Research 15, 2653–2687.
- **Shelton, J.**, Lampert, C., 2013. Approximate Inference with δ-insensitive Marginal Loss. Women in Machine Learning Workshop (WiML 2013) in conjunction with NIPS, Lake Tahoe, Nevada.
- Bornschein, J., **Shelton, J.**, Sheikh, A.-S., and Lücke, J., 2011. The Maximal Causes of Binary Data. Bernstein Conference on Computational Neuroscience (BCCN).

Chapter 2

Select and Sample - A Model of Efficient Neural Inference and Learning

In this Chapter we introduce a novel inference scheme combining latent variable preselection and sampling for the acceleration of EM.

The work presented in this Chapter can be found in the following publications: Shelton et al. (2011a,b).

2.1 Introduction

An increasing number of experimental studies indicate that perception encodes a posterior probability distribution over possible causes of sensory stimuli, which is used to act close to optimally in the environment. One outstanding difficulty with this hypothesis is that the exact posterior will in general be too complex to be represented directly, and thus neurons will have to represent an approximation of this distribution. Two influential proposals for how neural populations could achieve an efficient posterior representation are 1) that neural activity represents samples of the underlying distribution, or 2) that they represent a parametric representation of a variational approximation of the posterior. In this work, we propose a method for accelerated inference in EM that can account for both of these suggested neural approaches and their respective advantages. Namely, our approach combines the strengths of both sampling methods and variational approximations: it can represent multiple modes and arbitrary correlations and reduce the represented space to regions of high probability mass, respectively. Neurally, the combined method can be interpreted as a feed-forward preselection of the relevant state space, followed by a neural dynamics implementing Markov Chain Monte Carlo (MCMC) to approximate the pos-

terior over the relevant states. We demonstrate the effectiveness and efficiency of this approach on a sparse coding model. In numerical experiments on artificial data and image patches, we compare the performance of the algorithms to the performance of exact EM, variational state space selection alone, MCMC alone, and the combined Select and Sample approach. For sparse coding we find that it enables applications easily exceeding a thousand observed and a thousand hidden dimensions.

2.1.1 Background

According to the statistical approach to perception which is becoming increasingly popular, our brain would represent not only the most likely interpretation of a stimulus, but also the uncertainty associated with it. Ideally, the brain would represent the full posterior distribution over all possible interpretations of the stimulus, which is statistically optimal for inference and learning (Dayan and Abbott, 2001; Rao et al., 2002; Fiser et al., 2010). An increasing number of psychophysical and electrophysiological studies (Ernst and Banks, 2002; Weiss et al., 2002; Kording and Wolpert, 2004; Beck et al., 2008; Trommershäuser et al., 2008; Berkes et al., 2011a) support this hypothesis.

One approach researchers have taken to explore processing in neural circuits is to assume that neuronal activity represent the parameters of a variational approximation of the real posterior (Ma et al., 2006; Turner et al., 2011). While this approach allows the instantaneous representation of an approximate version of the full posterior, the number of neurons still explodes with the number of variables. For example, approximating the posterior with a Gaussian distribution requires N^2 parameters to represent the covariance matrix over N variables. Another approach is to identify neurons as variables, and interpret neural activity as samples from their posterior (Lee and Mumford, 2003a; Hoyer, 2003; Fiser et al., 2010). This interpretation is consistent with a range of experimental observations, including neural variability (which would result from the uncertainty in the posterior) and spontaneous activity (corresponding to samples from the prior in the absence of a stimulus (Fiser et al., 2010; Berkes et al., 2011a). The advantage of a sampling-based representation is that the number of neurons scales linearly with the number of variables, while retaining the ability to represent arbitrarily complex posteriors given enough samples. However, an important consideration with this approach is the amount of time needed to collect a sufficient number of samples in order to form a complete enough representation of the posterior. modelling studies have shown that a small number of samples are sufficient to perform well on low-dimensional tasks (intuitively, this is because taking a low-dimensional marginal of the posterior accumulates samples over all dimensions) (Vul et al., 2009; Berkes et al., 2011b). However, most sensory data is inherently very high-dimensional. For instance, to faithfully represent visual scenes

containing potentially many objects and object parts, one requires a high-dimensional latent space to represent the high number of potential causes. For high dimensionalities, sampling approaches usually suffer from long burn-in times and require in general a high number of samples to represent non-trivial posteriors. With variable preselection, Select and Sample efficiently minimizes the drawbacks of sampling methods and simultaneously takes advantage of their ability to represent complex distributions.

2.2 A Select and Sample Approach to Approximate Inference

To illustrate Select and Sample, we will consider the task of inferring the true latent variables in a visual stimulus, e.g. the objects in a visual scene based on the image projected on the retina. We will refer to the sensory stimulus (the image) as a *data point*, $\mathbf{y} = (y_1, \ldots, y_D)$, and we will refer to the hidden causes (the objects) as $\mathbf{s} = (s_1, \ldots, s_H)$ with s_h denoting the *hidden variable* or *hidden unit* h. The data distribution can then be modelled by a generative data model: $p(\mathbf{y} \,|\, \Theta) = \sum_{\mathbf{s}} p(\mathbf{y} \,|\, \mathbf{s}, \Theta)\, p(\mathbf{s} \,|\, \Theta)$ with Θ denoting the parameters of the model. If we assume that the data distribution can be optimally modelled by the generative distribution for optimal parameter Θ^*, then the posterior probability $p(\mathbf{s} \,|\, \mathbf{y}, \Theta^*)$ represents optimal inference given a data point $\mathbf{y}$. The parameters Θ^* given a set of N data points $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ are given by the maximum likelihood parameters $\Theta^* = \mathrm{argmax}_{\Theta}\{p(\mathbf{y} \,|\, \Theta)\}$.

We use EM to to find the maximum likelihood solution of the parameters. As described in Chapter 1.2, EM is an iterative algorithm to compute the maximum likelihood estimate of the model parameters of a given graphical model (see e.g. (Dempster et al., 1977; Neal and Hinton, 1998)). EM algorithm iterates between the E-step (1.6) and M-step (1.7) where the posterior distribution is optimized, followed by the the optimization model parameters, respectively. The parameter updates in the M-step usually depend just on a small number of expectations of the posterior, $\langle g(\mathbf{s}) \rangle_{p(\mathbf{s} \,|\, \mathbf{y}^{(n)}, \Theta)} = \sum_{\mathbf{s}} p(\mathbf{s} \,|\, \mathbf{y}^{(n)}, \Theta)\, g(\mathbf{s})$. Although $g(\mathbf{s})$ is usually an elementary function of the latent variables $\mathbf{s}$, the expectations are the computationally demanding part of EM optimization and must be approximated with some method. The EM iterations can be associated to neural processing by relating the E-step to the assumption that neural activity represents the posterior over hidden variables, and relating the M-step equations to long-term changes in synaptic weights due to synaptic plasticity. Here we will combine two approximations of the expectations in EM and show that these approximations are analogous to prominent models of neural processing.

2.2.1 Selection

Feed-forward processing has frequently been discussed as an important component of neural processing (Rosenblatt, 1958; LeCun et al., 1989; Riesenhuber and Poggio, 1999, 2002). One perspective on this early componenet of neural activity is as a preselection of candidate units (variables, hypotheses, etc.) for a given sensory stimulus ((Körner et al., 1999; Lee and Mumford, 2003b; Yuille and Kersten, 2006) and many more), with the goal of reducing the computational demand of an otherwise too complex computation. In the context of probabilistic approaches, we can emulate neural feed-forward processing with preselection using Expectation Truncation (ET; described in 1.2). After preselection of relevant variables, the posterior distribution can be approximated by a truncated distribution over a reduced set of latent states:

$$\begin{aligned} & p(\mathbf{s}^{(n)}|\mathbf{y}^{(n)},\Theta) \\ & \quad \approx q_n(\mathbf{s}^{(n)};\Theta) = \frac{p(\mathbf{s}^{(n)},\mathbf{y}^{(n)}|\,\Theta)\,\mathbb{I}(\mathbf{s}^{(n)} \in \ \mathcal{K}_n)}{\sum\limits_{\mathbf{s}'^{(n)}\in\mathcal{K}_n} p(\mathbf{s}'^{(n)},\mathbf{y}^{(n)}|\,\Theta)}, \end{aligned} \tag{2.1}$$

where $\mathcal{K}_n$ contains the latent states of the relevant variables for data point $\mathbf{y}^{(n)}$, and $\mathbb{I}(\mathbf{s} \in \mathcal{K}_n) = 1$ if $\mathbf{s} \in \mathcal{K}_n$ and 0 otherwise. Since for many applications the posterior mass is concentrated in small volumes of the state space, the approximation quality can stay high even for relatively small sets $\mathcal{K}_n$. Indeed, this property is observed to hold for many types of data in the auditory, visual or general pattern recognition domains. The expectations in the M-step can be computed using the truncated posterior distribution:

$$\langle g(\mathbf{s})\rangle_{p(\mathbf{s}\,|\,\mathbf{y}^{\,(n)},\Theta)} \approx \ \langle g(\mathbf{s})\rangle_{q^{(n)}(\mathbf{s};\Theta)} = \frac{\sum_{\mathbf{s}\in\mathcal{K}_n} p(\mathbf{s},\mathbf{y}^{\,(n)}\,|\,\Theta)\,g(\mathbf{s})}{\sum_{\mathbf{s}'\in\mathcal{K}_n} p(\mathbf{s}',\mathbf{y}^{\,(n)}\,|\,\Theta)}\,. \tag{2.2}$$

This approximation allows the expectations to be computed over the smaller state space $\mathcal{K}_n$, leading to a reduction in computational demands. As discussed in the Introduction Chapter 1.2, the set $\mathcal{K}_n$ needs to be carefully and efficiently selected prior to the computation of expectations in the E-step for significant acceleration of EM. A selection function $\mathcal{S}_h(\mathbf{y},\Theta)$ preselects latent variables s_h that are most likely to have contributed to a data point $\mathbf{y}^{\,(n)}$. The reduced set of preselected latent variable states $\mathcal{K}_n$ is defined as follows: $\mathcal{K}_n = \{\mathbf{s}\,|\,\text{for all } h \notin \mathcal{I}:\ s_h = 0\}$, where $\mathcal{I}$ contains the $H' \ll H$ indices with the highest values of a selection function $\mathcal{S}_h(\mathbf{y},\Theta)$ (compare Figure 2.1).

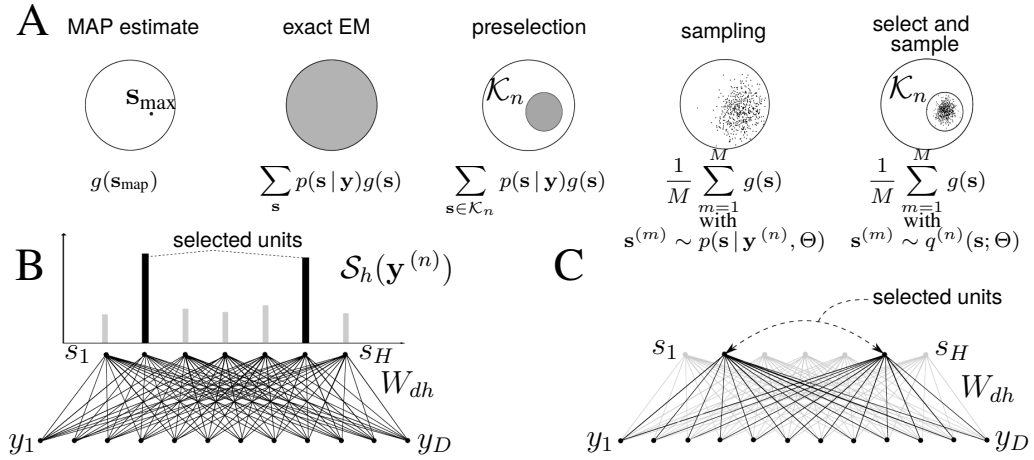


Figure 2.1: **A** *Simplified illustration of the posterior mass and the respective regions used by each approximation to compute the expectations* $\langle g(\mathbf{s})\rangle$. **B** *Graphical model showing full connections* W_{dh} *between the data point* $\mathbf{y}^{(n)}$ *and latent variables/units* $\mathbf{s} = (s_1, \ldots, s_H)$ *and how* H' *variables are selected from* H *to form a given state set* K_n *and* W_{dh} *is affected accordingly.* **C** *Illustrates how sampling draws samples from this reduced set for the Select and Sample approach (with e.g. the right-most posterior mass in* **A***).*

2.2.2 Sampling

An alternative way to approximate the expectations in Equation 2.2 is by sampling from the posterior distribution, and using the samples to compute the average:

$$\langle g(\mathbf{s})\rangle_{p(\mathbf{s}\,|\,\mathbf{y}^{(n)},\Theta)} \approx \tfrac{1}{M}\sum_{m=1}^{M} g(\mathbf{s}^{(m)}) \text{ with } \mathbf{s}^{(m)} \sim p(\mathbf{s}\,|\,\mathbf{y}^{(n)}, \Theta), \tag{2.3}$$

where M denotes the total number of samples drawn for a data point $\mathbf{y}^{(n)}$. The challenging aspect of this approach is to efficiently draw samples from the posterior. In large dimensional sample space, this is mostly done by Markov Chain Monte Carlo (MCMC). This class of methods draws samples from the posterior distribution such that each subsequent sample is drawn relative to the current state, and the resulting sequence of samples form a Markov chain. A new sample is accepted with a probability of $\max(1, \frac{p(\mathbf{s}^{\text{new}})}{p(\mathbf{s}^{\text{current}})})$. In the limit of a large number of samples, MCMC methods are theoretically able to represent any probability distribution. However, the number of samples required in large dimensional space can be very large (Figure 2.1**A**, sampling). See (Neal, 1993) for an extensive review of MCMC methods for probabilistic inference.

2.2.3 Select and Sample

While preselection and sampling seem to be very different in nature, their formulations as approximations to expectations (2.2) allow for a straight-forward combination of both approaches: Given a data point, $\mathbf{y}^{(n)}$, we first approximate the expectation (2.2) using the variational distribution $q^{(n)}(\mathbf{s};\Theta)$ as defined by preselection (2.2). Second, we approximate the expectations with respect to $q^{(n)}(\mathbf{s};\Theta)$ using sampling. The combined approach is thus given by:

$$\langle g(\mathbf{s})\rangle_{p(\mathbf{s}\,|\,\mathbf{y}^{(n)},\Theta)} \;\approx\; \langle g(\mathbf{s})\rangle_{q^{(n)}(\mathbf{s};\Theta)} \;\approx\; \tfrac{1}{M}\sum_{m=1}^{M} g(\mathbf{s}^{(m)}) \quad \text{with} \quad \mathbf{s}^{(m)} \sim q^{(n)}(\mathbf{s};\Theta) \quad (2.4)$$

where $\mathbf{s}^{(1)}$ to $\mathbf{s}^{(M)}$ denote samples from the truncated distribution $q^{(n)}$. Instead of drawing from a distribution over the entire state space, approximation (2.4) requires only samples from a potentially very small subspace $\mathcal{K}_n$ (Figure 2.1). In the subspace $\mathcal{K}_n$, most of the original probability mass is concentrated in a smaller volume, and thus we expect MCMC algorithms to perform more efficiently, as they need to explore a smaller volume, shortening burn-in times, and reducing the number of samples needed to trace the distribution. Compared to selection alone, the Select and Sample approach will represent an increase in efficiency as soon as the number of samples required for a good approximation is less then the number of states in $\mathcal{K}_n$. In the following, we will systematically investigate the computational efficiency of the Select and Sample approach in comparison with selection and sampling alone using concrete examples of generative models with real-world scales.

2.3 Sparse Coding: An Example Application

To explore the efficiency and performance properties of the Select and Sample approach as well as to study its biological plausibility, we apply it to a sparse coding model of images. The choice of a sparse coding model has numerous advantages. First, it is a non-trivial model that has been extremely well-studied in machine learning research, and for which efficient algorithms exist (e.g. (Lee et al., 2007; Mairal et al., 2010)). Second, it has become a standard (albeit somewhat simplistic) model of the organization of receptive fields in primary visual cortex (Olshausen and Field, 1996; van Hateren and van der Schaaf, 1998; Ringach, 2002). Here we consider a discrete variant of this model known as Binary Sparse Coding (BSC; (Henniges et al., 2010; Lücke and Eggert, 2010), also compare (Haft et al., 2004)), which has binary hidden variables but otherwise the same features as standard sparse coding versions. The generative model for BSC is expressed by

$$p(\mathbf{s}|\pi) = \prod_{h=1}^{H} \pi^{s_h} \big(1-\pi\big)^{1-s_h}\,, \qquad p(\mathbf{y}|\mathbf{s},W,\sigma) = \mathcal{N}(\mathbf{y}; W\mathbf{s}, \sigma^2 I)\,, \tag{2.5}$$

where $W \in \mathbb{R}^{D\times H}$ denotes the basis vectors between $\mathbf{s}$ and $\mathbf{y}$, and π parameterizes the sparsity of the Bernoulli prior $p(\mathbf{s}|\pi)$ on $\mathbf{s}$, which we denote $\mathcal{B}(\mathbf{s};\pi)$.

The M-step updates of the BSC learning algorithm (see e.g. (Lücke and Eggert, 2010)) are given by:

$$W^{\text{new}} = \big(\sum_{n=1}^{N} \mathbf{y}^{(n)} \left\langle \mathbf{s} \right\rangle_{q^{(n)}}^{T}\big)\,\big(\sum_{n=1}^{N} \left\langle \mathbf{s}\,\mathbf{s}^{T}\right\rangle_{q^{(n)}}\big)^{-1}\,, \tag{2.6}$$

$$(\sigma^2)^{\text{new}} = \tfrac{1}{ND}\sum_{n}\Big\langle \big|\big|\mathbf{y}^{(n)} - W\,\mathbf{s}\big|\big|^2\Big\rangle_{q^{(n)}}, \qquad \pi^{\text{new}} = \tfrac{1}{N}\sum_{n} |\left\langle \mathbf{s}\right\rangle_{q^{(n)}}|, \tag{2.7}$$

$$\text{where } |\mathbf{x}| = \tfrac{1}{H}\sum_{h} x_h.$$

The only expectations needed for the M-step are thus $\left\langle \mathbf{s}\right\rangle_{q^{(n)}}$ and $\left\langle \mathbf{s}\mathbf{s}^{T}\right\rangle_{q^{(n)}}$.

We will compare inference and learning using different methods:

BSC$^{\text{exact}}$. An EM algorithm without approximations is obtained if we use the exact posterior for the expectations: $q^{(n)} = p(\mathbf{s}\,|\,\mathbf{y}^{\,(n)}, \Theta)$. We will refer to this exact algorithm as $\mathrm{BSC}^{\,\mathrm{exact}}$. While directly computable, the expectations for $\mathrm{BSC}^{\,\mathrm{exact}}$ require sums over the entire state space, i.e. over 2^H terms. For large numbers of latent dimensions $\mathrm{BSC}^{\,\mathrm{exact}}$ is thus intractable.

BSC$^{\text{select}}$. An algorithm that more efficiently scales with the number of hidden dimensions is obtained by applying preselection. For the BSC model we use $q^{(n)}$ as given in (2.2) and $\mathcal{K}_n = \{\mathbf{s}\,|\,(\text{for all } h \notin \mathcal{I}:\ s_h = 0) \text{ or } \sum_h s_h = 1\}$. Note that in addition to states as in (1.11) we include all states with one non-zero unit (all singletons). Including them avoids EM iterations in the initial phases of learning that leave some basis functions unmodified (see (Lücke and Eggert, 2010)). As selection function $\mathcal{S}_h(\mathbf{y}^{\,(n)})$ to define $\mathcal{K}_n$ we use:

$$\mathcal{S}_h(\mathbf{y}^{\,(n)}) = \frac{\mathbf{W}_h^T\,\mathbf{y}^{\,(n)}}{||\mathbf{W}_h||_2} \quad \text{with} \quad ||\mathbf{W}_h||_2 = \sqrt{\textstyle\sum_{d=1}^{D}(W_{dh})^2}\,. \tag{2.8}$$

A large value of $\mathcal{S}_h(\mathbf{y}^{\,(n)})$ strongly indicates that $\mathbf{y}^{\,(n)}$ contains the basis function $\mathbf{W}_h$ as a component (see Figure 2.1C). Note that (2.8) can be related to a deterministic ICA-like selection of a hidden state $\mathbf{s}^{(n)}$ in the limit case of no noise (compare (Lücke and Eggert, 2010)). Further restrictions of the state space are possible but require modified

M-step equations (see (Lücke and Eggert, 2010; Henniges et al., 2010)), which will not be considered here. Selection functions for more complex models will be thoroughly addressed in Chapter 4.

BSC$^{\text{sample}}$. An alternative non-deterministic (stochastic) approach can be derived using Gibbs sampling. Gibbs sampling is an MCMC algorithm which systematically explores the sample space, considering each dimension individually, conditioning the acceptance of a new sample based on values of the remaining samples. In other words, the transition probability from the current sample to a new candidate sample is given by $p(s_h^{\text{new}}|\mathbf{s}_{\setminus h}^{\text{current}})$. In our case of a binary sample space, this equates to selecting one random axis $h \in \{1, \ldots, H\}$ and toggling its bit value (thereby changing the binary state in that dimension), leaving the remaining axes unchanged. Specifically, the posterior probability computed for each candidate sample is expressed by:

$$p(s_h = 1 \,|\, \mathbf{s}_{\setminus h}, \mathbf{y}) = \frac{p(s_h = 1, \mathbf{s}_{\setminus h}, \mathbf{y})^\beta}{p(s_h = 0, \mathbf{s}_{\setminus h}, \mathbf{y})^\beta + p(s_h = 1, \mathbf{s}_{\setminus h}, \mathbf{y})^\beta}, \tag{2.9}$$

where β is used to smooth out the posterior distribution. To ensure an appropriate mixing behavior over a wide range of σ (note that σ is a model parameter that changes with learning), we define $\beta = \frac{T}{\sigma^2}$, where T is an annealing temperature parameter that is set manually and is selected such that sufficient mixing of the MCMC chains is achieved. A chain is referred to as well mixed when the sampler has adequately explored the state space and the samples can be considered to have been drawn from the desired target distribution (i.e. the proposal distribution converges to the stationary distribution), and annealing is often used to speed up mixing times (see e.g. (Neal, 1993)). The samples drawn by applying the described procedure can then be used to approximate the expectations of the sufficient statistics in the parameter Equations (2.6) and (2.7) using Equation (2.3).

BSC$^{\text{s+s}}$. The EM learning algorithm given by combining selection and sampling is obtained by applying (2.4). First note that inserting the BSC generative model into (2.1) results in:

$$q^{(n)}(\mathbf{s};\, \Theta) \;=\; \frac{\mathcal{N}(\mathbf{y}; W\mathbf{s}, \sigma^2 I)\, \mathcal{B}_{\mathcal{K}_n}(\mathbf{s}; \pi)\, \mathbb{I}(\mathbf{s} \in \mathcal{K}_n)}{\sum_{\mathbf{s}' \in \mathcal{K}_n} \mathcal{N}(\mathbf{y}; W\mathbf{s}', \sigma^2 I)\, \mathcal{B}_{\mathcal{K}_n}(\mathbf{s}'; \pi)} \tag{2.10}$$

where $\mathcal{B}_{\mathcal{K}_n}(\mathbf{s}; \pi) = \prod_{h \in \mathcal{I}} \pi^{s_h} (1 - \pi)^{1 - s_h}$. The remainder of the Bernoulli distribution cancels out. If we define $\tilde{\mathbf{s}}$ to be the binary vector containing all selected variables of $\mathbf{s}$, and if $\tilde{W} \in \mathbb{R}^{D \times H'}$ contains all basis functions of the selected units, we observe that the distribution is equal to the posterior with respect to a BSC generative model with H'

instead of H hidden variables:

$$p(\tilde{\mathbf{s}}\,|\,\mathbf{y},\Theta) \;=\; \frac{\mathcal{N}(\mathbf{y};\tilde{W}\tilde{\mathbf{s}},\sigma^2 I_{H'})\,\mathcal{B}(\tilde{\mathbf{s}};\pi)}{\sum_{\tilde{\mathbf{s}}'}\mathcal{N}(\mathbf{y};\tilde{W}\tilde{\mathbf{s}}',\sigma^2 I_{H'})\,\mathcal{B}(\tilde{\mathbf{s}}';\pi)} \;=\; p(\tilde{\mathbf{s}}\,|\,\mathbf{y},\Theta)$$

Instead of drawing samples from $q^{(n)}(\mathbf{s};\,\Theta)$ we can thus draw samples from the exact posterior with respect to the BSC generative model with H' dimensions. The sampling procedure for $\mathrm{BSC}^{\mathrm{sample}}$can thus be applied simply by ignoring the non-selected dimensions and their associated parameters. For different data points different latent dimensions will be selected such that averaging over data points can update all model parameters. For selection we again use selection functions (2.8) and again define $\mathcal{K}_n$ similar to (1.11):

$$\mathcal{K}_n = \left\{\mathbf{s}\,|\,\text{for all } h\notin\mathcal{I}:\;\; s_h = 0\right\}, \tag{2.11}$$

where $\mathcal{I}$ contains the H'-2 indices h with the highest values of a selection function $\mathcal{S}_h(\mathbf{y},\Theta)$ and two randomly selected dimensions (drawn from a uniform distribution over all non-selected dimensions). The two randomly selected dimensions fulfill the same purpose as the inclusion of singleton states for $\mathrm{BSC}^{\mathrm{select}}$. This prevents the possible propagation of errors from $q_{(n)}$ continuously assigning small probabilities to a variable s_h in early EM iterations. Preselection and Gibbs sampling on the selected dimensions define an approximation to the required expectations (2.2) and result in an EM algorithm referred to as $\mathrm{BSC}^{\mathrm{s+s}}$.

Complexity. Collecting the number of operations necessary to compute the expectation values for all four BSC cases, we arrive at

$$\mathcal{O}\big(NS(\underbrace{D}_{p(\mathbf{s},\mathbf{y})}+\underbrace{1}_{\langle\mathbf{s}\rangle}+\underbrace{H}_{\langle\mathbf{s}\mathbf{s}^T\rangle})\big) \tag{2.12}$$

where S denotes the number of hidden states that contribute to the calculation of the expectations. For the approaches with preselection ($\mathrm{BSC}^{\mathrm{select}}$, $\mathrm{BSC}^{\mathrm{s+s}}$), all the calculations of the expectations can be performed on the reduced latent space; therefore the H is replaced by H'. For $\mathrm{BSC}^{\mathrm{exact}}$this number scales exponentially in H: $S^{\mathrm{exact}} = 2^H$, and in in the $\mathrm{BSC}^{\mathrm{select}}$case, it scales exponentially in the number of preselected hidden variables: $S^{select} = 2^{H'}$. However, for the sampling based approaches ($\mathrm{BSC}^{\mathrm{sample}}$and $\mathrm{BSC}^{\mathrm{s+s}}$), the number S directly corresponds to the number of samples to be evaluated and is obtained empirically. As we will show later, $S^{\mathrm{s+s}} = 200\times H'$ is a reasonable choice for the interval of H' that we investigate in this work ($1 \leq H' \leq 40$).

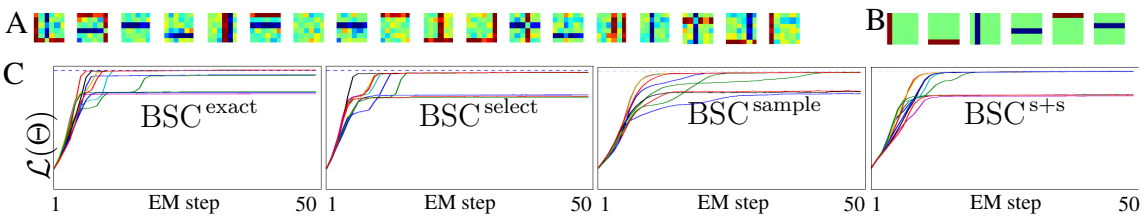


Figure 2.2: *Experiments using artificial bars data generated with $H = 12$ bars (hidden variables) and $D = 6 \times 6$ pixels (observed variables). Dotted line indicates the ground-truth log likelihood value.* **A** *Random selection of the $N = 2,000$ training data points $\mathbf{y}^{(n)}$* **B** *Learned basis functions W_{dh} after an successful training run* **C** *Development of the log likelihood over a period of 50 EM steps for all 4 investigated algorithms where different color plots are different runs of the same algorithm.*

2.4 Experiments

We compare the Select and Sample approach with selection and sampling applied individually on different data sets. In all experiments evaluating the two algorithms that use sampling, we draw 20 independent chains that are initialized at random states in order to increase the mixing of the samples. Also, of the samples drawn per chain, $\frac{1}{3}$ were used to as burn-in samples, and $\frac{2}{3}$ were retained samples.

2.4.1 Artificial Data

Before we apply the Select and Sample approach to large scale learning on image patches, we investigate its convergence properties on artificial data sets where ground-truth is available. As the following experiments were run on a small scale problem, all four algorithms ($\text{BSC}^{\text{exact}}$, $\text{BSC}^{\text{select}}$, $\text{BSC}^{\text{sample}}$and $\text{BSC}^{\text{s+s}}$) can be applied and compared. Furthermore, for all the experiments we computed the exact likelihood for each EM step.

Data for these experiments consisted of images generated with $H = 12$ ground-truth basis functions $\mathbf{W}_h^{\text{gt}}$ in the form of horizontal and vertical bars on a $D = 6 \times 6 = 36$ pixel grid. Each bar was randomly assigned to be either positive ($W_{dh}^{\text{gt}} \in \{0.0, 10.0\}$) or negative ($W_{dh'}^{gt} \in \{-10.0, 0.0\}$). $N = 2,000$ data points $\mathbf{y}^{(n)}$ were generated by linear combining these basis functions (see e.g. (Hoyer, 2002)). Using a sparseness value of $\pi_{\text{gt}} = \frac{2}{H}$ resulted in, on average, two active bars per data point. According to the model, we added Gaussian noise ($\sigma_{\text{gt}} = 2.0$) to the data (Figure 2.2 **A**). We applied all algorithms to the same data set and monitored the exact likelihood while these algorithms converged over a period of 50 EM steps (Figure 2.2 **C**). Although the calculation of the exact likelihood requires $\mathcal{O}(N2^H(D+H))$ operations, this is still feasible for such a small scale problem. When running models using preselection ($\text{BSC}^{\text{select}}$and $\text{BSC}^{\text{s+s}}$), we set

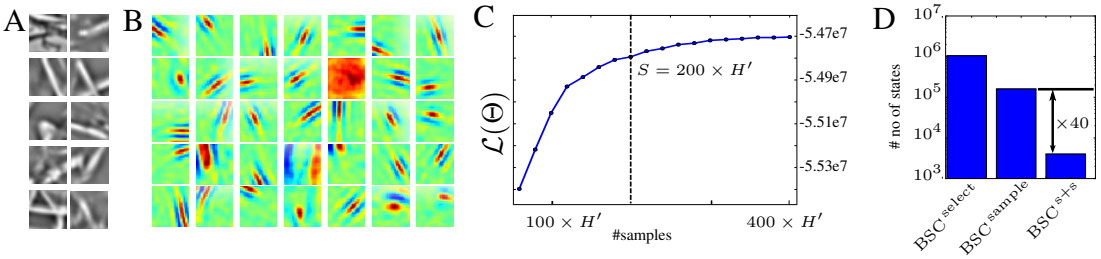


Figure 2.3: *Experiments on $D = 26 \times 26$ image patches:* **A** *A set of 10 randomly chosen image patches after DoG preprocessing.* **B** *A selection of the learned basis functions for $H = 800, H' = 20$ and number of samples set to $200 \times H'$. See the Supplementary Material 2.6 for the full set of basis functions.* **C** *The final approximate log likelihood then running the* $\mathrm{BSC}^{\mathrm{s+s}}$ *for various number of samples drawn per data point.* **D** *Assuming $H = 800$ and $H' = 20$, the number of hidden states to be evaluated are shown.*

H' to 6, effectively halving the number of hidden variables participating in the calculation of the expectations. For $\mathrm{BSC}^{\mathrm{sample}}$, we drew $S^{\mathrm{sample}} = 200 \times H = 2400$ samples from the posterior $p(\mathbf{s} \mid \mathbf{y}^{(n)})$ of each data point. For $\mathrm{BSC}^{\mathrm{s+s}}$, we drew $S^{\mathrm{s+s}} = 200 * H' = 1,200$ samples from the posterior of each data point. To ensure an appropriate mixing behavior, we furthermore set the annealing temperature to $T = 50$. In all these experiments we initialized the basis functions to the data mean plus Gaussian noise, the prior probability to $\pi_{\mathrm{init}} = \frac{1}{H}$ and the data noise to the variance of the data. All algorithms recover the correct set of bases functions in $> 50\%$ of the trials, and the sparseness prior π and the data noise σ with high accuracy. Comparing the computational costs of algorithms shows the benefits of preselection already for this small scale problem: While $\mathrm{BSC}^{\mathrm{exact}}$ calculates the expectations using the full set of $2^H = 4096$ hidden states, $\mathrm{BSC}^{\mathrm{select}}$ only considers $2^{H'} + (H - H') = 70$ states. The pure sampling based approaches performs $2,400$ evaluations while $\mathrm{BSC}^{\mathrm{s+s}}$ requires $1,200$ evaluations.

2.4.2 Natural Image Patches

To demonstrate the applicability of the Select and Sample approach to larger scale problems, we tested our approach natural image patches. We extracted $N = 40,000$ patches of size $D = 26 \times 26 = 676$ pixels from the van Hateren image database (van Hateren and van der Schaaf, 1998) [1], and preprocessed them using a Difference of Gaussians (DoG) filter, which approximates the sensitivity of center-on and center-off neurons found in the early stages of the mammalian visual processing. Filter parameters where chosen as in (Lücke, 2009; Puertas et al., 2010). The annealing temperature was set to $T = 20$. In all

[1] We restricted the set of images to 900 images without man-made structures (see Figure 2.3A). The brightest 2% of the pixels were clamped to the maximal value of the remaining 98% (influences of light-reflections were reduced this way)

following simulations, we used initialization procedure described earlier and ran 100 EM iterations to ensure proper convergence.

We first ran a series of experiments in order to investigate the effect of the number of samples used on the approximated log likelihood for values of H' ranging between 12 and 36, and setting $H = 800$. We observe with $\mathrm{BSC}^{\mathrm{s+s}}$that 200 samples per hidden dimension (total samples $= 200 \times H'$) are sufficient in that the final value of the likelihood after 100 EM steps begins to saturate. Particularly, increasing the number of samples does not increase the likelihood by more than 1% In Figure 2.3**C** we show the curve for $H' = 20$, but we obseverd the same trend for all other values of H'. Furthermore, we tested this number of samples ($200 \times H$) in the pure sampling case ($\mathrm{BSC}^{\mathrm{sample}}$) in order to monitor the likelihood behavior. We observed two consistent trends: 1) the algorithm did not converge to a high-likelihood solution, and 2) even when initialized at solution with high likelihood, the likelihood always decreases. This example demonstrates the gains of using Select and Sample above pure sampling: while $\mathrm{BSC}^{\mathrm{s+s}}$only needs $200 \times 20 = 4,000$ samples to robustly reach a high-likelihood solutions, by following the same regime with $\mathrm{BSC}^{\mathrm{sample}}$, not only did the algorithm poorly converge on a high-likelihood solution, but it used $200 \times 800 = 160,000$ samples to do so (Figure 2.3**D**).

2.4.3 Large Scale Experiment on Natural Image Patches

Comparison of the above results shows that the most efficient algorithm is obtained by a combination of preselection and sampling, our Select and Sample approach ($\mathrm{BSC}^{\mathrm{s+s}}$), with no or only minimal effect on the performance accuracy of the algorithm – as depicted in Figure 2.2 and 2.3. This efficiency allows for applications to much larger scale problems than would be possible by individual approximation approaches (Figure 2.3 and 2.5). To demonstrate the efficiency of the combined approach, we applied $\mathrm{BSC}^{\mathrm{s+s}}$to the same image data set, but with a very high number of observed and hidden dimensions. We extracted from the database $N = 500,000$ patches of size $D = 40 \times 40 = 1,600$ pixels. $\mathrm{BSC}^{\mathrm{s+s}}$was applied with the number of hidden units set to $H = 1,600$ and with $H' = 34$. Using the same conditions as in the previous experiments (notably $S = 200 * H' = 6,800$ samples and 100 EM iterations) we again obtain a set of Gabor-like basis functions, examples of which are shown in Figure 2.4**A**. Figure 2.4**B** shows the scaling behavior of models $\mathrm{BSC}^{\mathrm{select}}$and $\mathrm{BSC}^{\mathrm{s+s}}$in terms of the number of states the algorithm needs to evaluate, where the former scales exponentially and the latter scales linearly. We can see a great difference in necessary computational resources of these models at the considered preselection parameterization, $H' = 34$. To our knowledge, the presented results represent the largest application of sparse coding with a reasonably complete representation of the posterior.

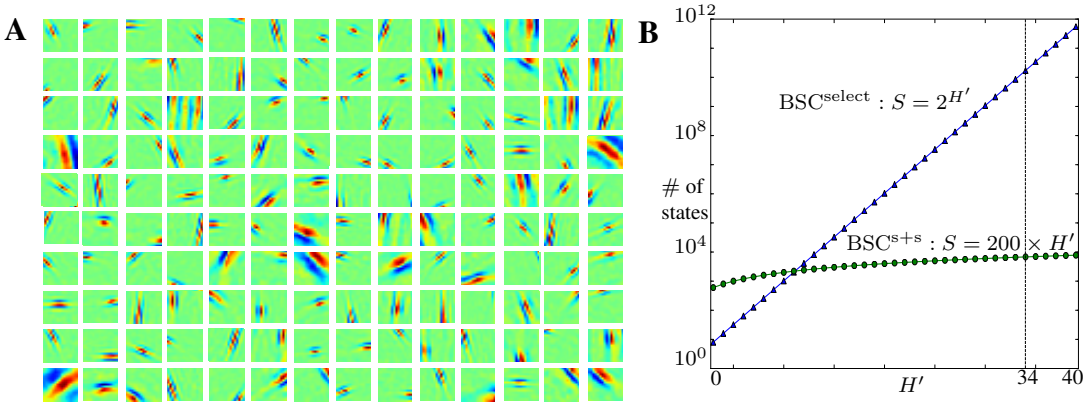


Figure 2.4: *Results of the large-scale experiment of* $\mathrm{BSC}^{\mathrm{s+s}}$ *on* $N = 500,000$ *image patches with* $D = 40 \times 40 = 1,600$ *pixels,* $H = 1,600$ *hidden dimensions, and* $H' = 34$. **A** *Random selection of the learned basis functions* W_{dh}*, exhibiting Gabor-like features.* **B** *Corresponding complexity in terms of the most costly operations for both of the preselection models consididered, namely, the number of states the algorithm needs to evaluate. Here* $\mathrm{BSC}^{\mathrm{select}}$ *scales exponentially, whereas* $\mathrm{BSC}^{\mathrm{s+s}}$ *scales linearly. Note the difference in complexity at the preselection parameterization,* $H' = 34$.

2.5 Discussion

In this Chapter, we introduced a novel efficient method for inference and unsupervised learning in probabilistic models that could plausibly be implemented in neural circuits which would allow the brain to use large scale models of the sensory input to make sense of its environment. This could be plausibly realized using two mechanisms that had been independently suggested in the context of a the statistical framework for perception: feed-forward input preselection (Lücke and Eggert, 2010), and sampling (Lee and Mumford, 2003a; Hoyer, 2003; Fiser et al., 2010). We showed that the two seemingly contrasting approaches can be combined based on their interpretation as approximate inference methods, resulting in a considerable increase in computational efficiency.

We investigated the applicability and efficiency of Select and Sample analytically and numerically using sparse coding model of natural images—a standard model for neural response properties in V1 (Olshausen and Field, 1996; van Hateren and van der Schaaf, 1998). Comparisons to exact inference, preselection alone, and sampling alone showed dramatically improved scaling behavior with the number of observed and hidden dimensions. To the best of our knowledge, the only other sparse coding implementation applied to a comparable problem size ($D = 20 \times 20$, $H = 2,000$) assumed a Laplace prior, which results in a simple uni-modal posterior, and used a MAP approximation, which further reduces the posterior to a single point, and is known to produces biased parameters (Lee et al., 2007). Our method is able to achieve convergence while still representing a sig-

nificant part of the posterior, which is crucial for learning parameters like data noise and sparsity, and to correctly act when faced with uncertain input (Rao et al., 2002; Trommershäuser et al., 2008; Fiser et al., 2010).

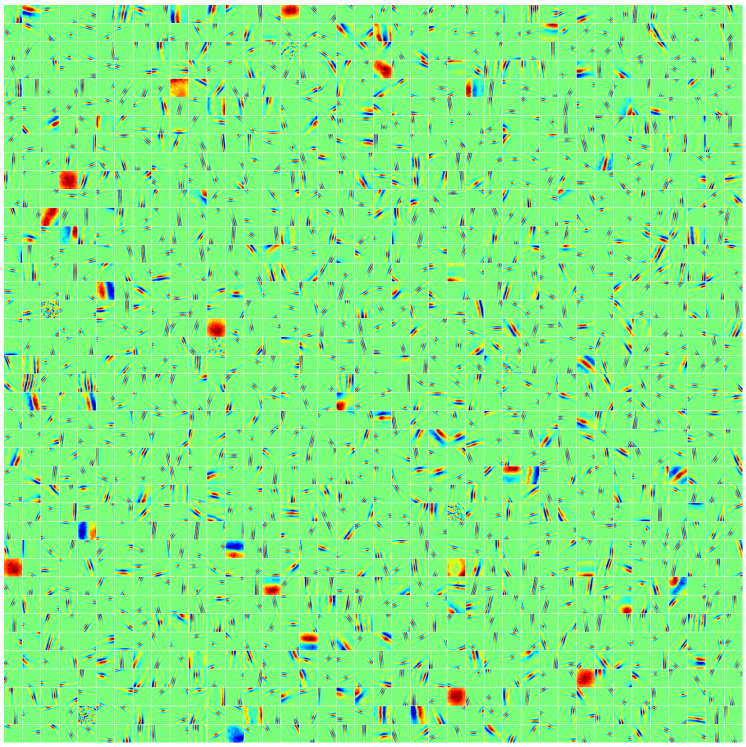
Concretely, we used a sparse coding model with binary latent variables, mainly because it offered a systematic comparison of Select and Sample with exact EM for low-dimensional problems, but we will show an application to a sparse coding model with continuous variables in the next Chapter. In the model, the selection step results in a simple, local and neurally plausible integration of input data, given by (2.8). Sampling was implemented using the Gibbs sampling method, which is also neurally plausible – neurons can individually sample their next state based on the current state of the other neurons as transmitted through recurrent connections (Berkes et al., 2011b).

The selection-based EM algorithms frequently encountered the problem of propagating errors due to $q_{(n)}$ continuously assigning small probabilities to a variable s_h in early EM iterations. This lead to the algorithm not being able to converge (i.e. verified on ground-truth data). This would occur because the optimization of $q_{(n)}$ in early iterations of EM starts from randomly initialized $\mathbf{s}_h$. Unfortunately this meant that selection-based EM would "get stuck", assigning not only arbitrarily small probabilities to a variable s_h, but also higher probability values to s_h that were selected due to an initially high probability value. Those s_h would therefore always be deemed relevant, regardless of their true relevancy. We solved this problem by randomly selecting a few extra hidden indices $H' + \frac{H'}{10}$ to give the algorithm an opportunity to evaluate possibly unused variables which might be relevant for $\mathbf{y}^{(n)}$.

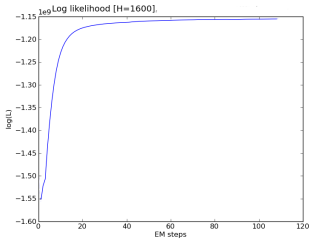
We expect the Select and Sample strategy to be widely applicable to machine learning models whenever the posterior probability mass is concentrated in a small subspace of the entire latent space. Furthermore, we expect that more sophisticated preselection mechanisms and sampling schemes to lead to further reduction in computational costs. Both of these ideas will be explored in the subsequent Chapters.

2.6 Supplementary Material

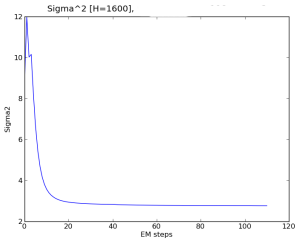
Large Scale Sparse Coding Application using Select and Sample



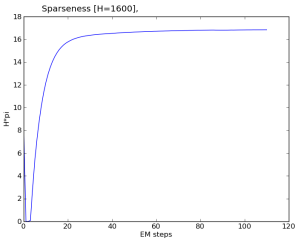
(a) Set of W bases functions



(b) Log likelihood



(c) Data noise σ^2



(d) Sparsity $\pi H'$

Figure 2.5: *Large scale results for Select and Sample with Binary Sparse Coding (*$\mathrm{BSC}^{\mathrm{s+s}}$*) ran for* 100 *EM iterations: applied to* $N = 500,000$ *image patches of size* $D = 40 \times 40 = 1.600$ *with* $H = 1600$ *hidden variables and* $H' = 36$ *relevant preselected variables. All other parameters were set as described in 2.4. The figure shows the (a) the learned basis functions* $\mathbf{W}_h$*, (b) approximated log likelihood, (c) data noise (c) and sparsity.*

Chapter 3

Nonlinear Spike-and-Slab Sparse Coding for Intepretable Image Encoding

In this Chapter, we use the Select and Sample approach from Chapter 2, but apply it to a more complicated Sparse Coding model. We introduce a novel model, *Nonlinear Spike-and-slab sparse coding*, for which, due to its intractabilities, we modify Select and Sample and derive a new optimization method using *exact Gibbs sampling with latent variable preselection*.

The work presented hierin can be found in the following publications: Shelton et al. (2012b,a, 2013, 2015).

3.1 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 chapter 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; Eldar and 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 on simple cells in the visual cortex (Hubel and Wiesel, 1959), 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) $\mathbf{y} = (y_1, \ldots, y_D)^T$ is associated with a sparse vector of latent variables $\mathbf{s} = (s_1, \ldots, s_H)^T$ (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 $\mathbf{s}$ 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 $\mathbf{s}$ are zero or close to zero.

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

$$loss(\mathbf{y}, W) \quad := \quad \min_{\mathbf{s}} \ \frac{1}{2}||\mathbf{y} - \mathbf{W}\mathbf{s}||_2^2 \ + \ a||\mathbf{s}||_1, \tag{3.1}$$

with the objective to minimize the loss between the image $\mathbf{y}$ and its *linear* reconstruction/estimation $\mathbf{W}\mathbf{s}$ (or equivalently $\sum_h s_h \mathbf{W}_h$ where $\mathbf{W}$ is the $D \times H$ matrix of $\mathbf{W}_h$ dictionary elements/components), with a penalty on the l_1-norm of the vector $\mathbf{s}$. The penalty is controlled by a regularization parameter a, which dictates how sparse the coefficients $\mathbf{s}$ in the reconstruction of $\mathbf{y}$ will be. Objective (3.1) and associated optimization algorithms are often referred to as *basis pursuit* (Chen et al., 1998) and also appear in nonlinear programming literature (Mangasarian, 1969; Han and Mangasarian, 1979). Additionally, objective (3.1) is associated with a related learning problem called the *Lasso* (Tibshirani, 1996), for which $\mathbf{W}$ is fixed and for which an expectation over squared error terms is minimized subject to an l_1 penalty.

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

$$p(\mathbf{y} \mid \Theta) = \int_{\mathbf{s}} p(\mathbf{y} \mid \mathbf{s}, \Theta)\, p(\mathbf{s} \mid \Theta)\, d\mathbf{s}, \tag{3.2}$$

where the latent causes are characterized by $p(\mathbf{s} \mid \Theta)$ with a sparse prior distribution. The observation/image described by $p(\mathbf{y} \mid \mathbf{s}, \Theta)$ is typically a Gaussian distribution with a mean $\mu = \sum_h s_h \mathbf{W}_h$, i.e. centered at the linear superposition of components $\mathbf{W}_h \in \mathbb{R}^D$. If the Laplace distribution is used as prior distribution, it can be shown that the minimization of objective (3.1) with respect to the dictionary elements corresponds to EM learning using the maximum a-posteriori (MAP) approximation for the posterior (e.g. (Murphy, 2012)). For dictionary learning, the formulation of objective (3.1) 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 realistic 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 3.1. 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 coefficients 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 (e.g. (Lázaro-gredilla and Titsias, 2011; Mohamed et al., 2012; Goodfellow et al., 2012; Sheikh et al., 2014)), which is a distribution consisting of a discrete binary part and a continuous Gaussian part (see the first column in Figure 3.2 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

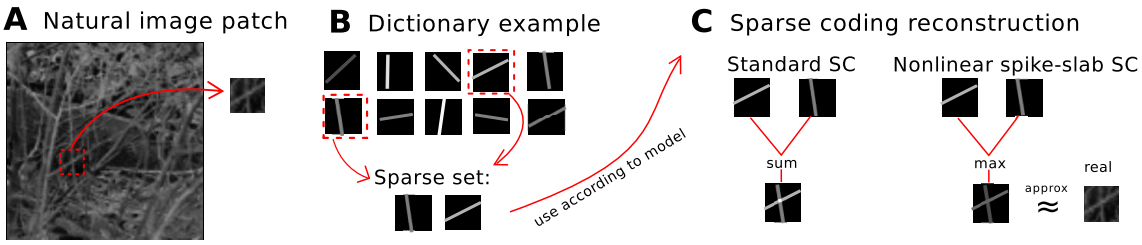


Figure 3.1: Toy example illustrating the problem setting: approximating occlusions in images. *Given an image patch with occlusions (***A***), assume both the linear and nonlinear sparse coding models were given the true generating dictionary elements (***B***) and the task is for each model to use a 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.*

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, illustrated in Figure 3.1**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 linear 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 multimodal 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. An illustration of both the linear and nonlinear models, both the Laplace and spike-and-slab prior distributions, and the resulting posterior distributions from either combination of model and prior is shown in Figure 3.2.

This work proposes a novel sparse coding model that combines both of these improve-

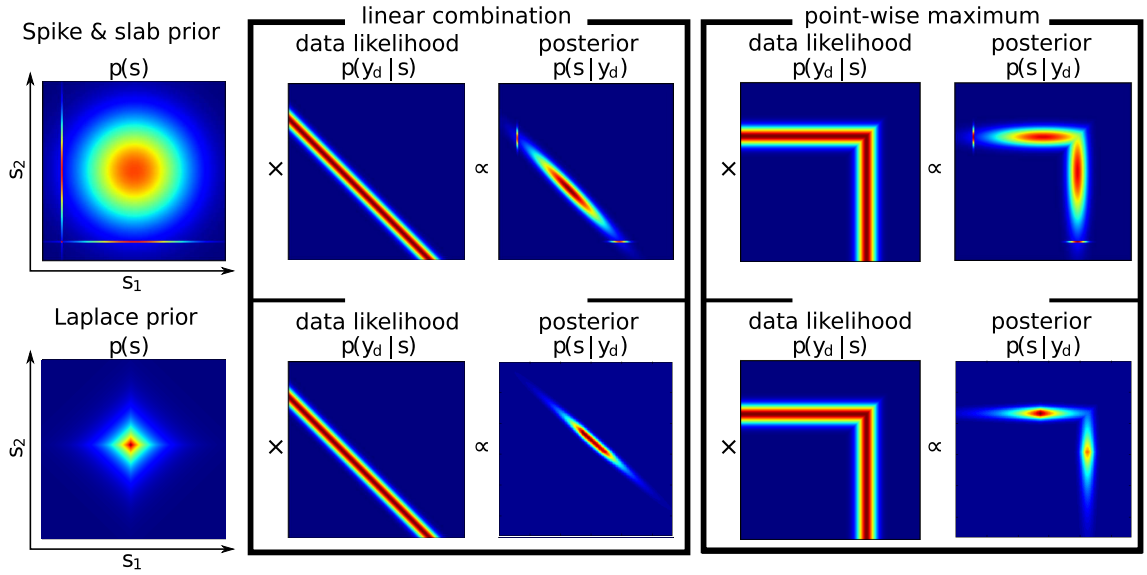


Figure 3.2: Illustration of choice of prior distribution and multimodality in the latent space. *A H=2-dimensional spike-and-slab and Laplace priors over latent variables and the multimodal posterior distribution induced by these priors for both linear and nonlinear data likelihoods.*

ments – 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., 2011b). 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 data sets 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 data sets for the task of dictionary learning and image reconstruction on both artificial data with controlled forms of sparse structure as well as natural data. 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 Chapter 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.

3.2 Model: Nonlinear Spike-and-Slab Sparse Coding

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

$$p(y_d \,|\, \mathbf{s}, \Theta) = \mathcal{N}(y_d;\, \max_h\{s_h W_{dh}\},\, \sigma^2). \tag{3.3}$$

Here, in contrast to the standard linear formulation in (3.2), the likelihood contains the nonlinear term $\max_h\{s_h W_{dh}\}$ instead of the linear $\sum_h s_h W_{dh}$ (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:

$$p(b_h \,|\, \Theta) = \mathcal{B}(b_h; \pi) = \pi^{b_h}\,(1-\pi)^{1-b_h} \tag{3.4}$$

$$p(z_h \,|\, \Theta) = \mathcal{N}(z_h;\, \mu_{\text{pr}}, \sigma^2_{\text{pr}}), \tag{3.5}$$

The columns of the matrix $\mathbf{W} = (W_{dh})$ are the dictionary elements/generative fields, $(\mathbf{W}_h)_{h=1}^H$, with one $\mathbf{W}_h$ 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

$$p(\mathbf{s}|\mathbf{y}, \theta) = \frac{p(\mathbf{y}|\mathbf{s}, \theta)\; p(\mathbf{s}|\theta)}{\int_{\mathbf{s}'} p(\mathbf{y}|\mathbf{s}', \theta)\; p(\mathbf{s}'|\theta)\, d\mathbf{s}'}. \tag{3.6}$$

Identical to the standard sparse coding formulation in Equations (3.1) and (3.2), 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 3.1 contains a toy illustration of part

of the generative process and model differences between the standard linear model and nonlinear model. Figure 3.1**A** shows an example natural image, exhibiting naturally occurring 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 3.1**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 3.1**B** generated the image patch in 3.1**A**. Figure 3.1**C** shows how the (non)linear assumptions of the models manifest when the given components from 3.1**B** are combined according to each model in order to reconstruct the patch in 3.1**A**. As can be seen for the sum operation in 3.1**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 3.1**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 3.1**C**) allows the model to adapt the intensity of each image component used to match what it observed in the data.

3.2.1 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., 2011; Lázaro-gredilla and Titsias, 2011; Goodfellow et al., 2012), 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. (Eldar and 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., 2013a; 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., 1999), 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. 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.

3.3 Inference: Exact Gibbs Sampling with Preselection

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

3.3.1 Parameter Estimation

To estimate the model parameters Θ of the generative model in (3.3) we use Expectation Maximization (EM). We do inference in the E-step with our proposed method combining

sampling and latent preselection (Shelton et al., 2011b), which we will introduce in the next Section. Optimization in the EM framework entails 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:

$$\hat{\sigma}^2 = \frac{1}{NDK} \sum_n \sum_d \sum_k \left(\max_h \left\{ W_{dh} s_{kh}^{(n)} \right\} - y_d^{(n)} \right)^2, \tag{3.7}$$

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:

$$\hat{\sigma}^2 = \left\langle W_{dh} s_h^{(n)} - y_d^{(n)} \right\rangle^*, \tag{3.8}$$

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

$$\langle f(s) \rangle^* = \sum_n \frac{\int_s p(\mathbf{s}|\mathbf{y}^{(n)}, \Theta) \, f(\mathbf{s}) \, \delta(\text{h is max}) \, d\mathbf{s}}{\int_s p(\mathbf{s}|\mathbf{y}^{(n)}, \Theta) \, \delta(\text{h is max}) \, d\mathbf{s}}, \tag{3.9}$$

where δ is the indicator function denoting the domain to integrate over, namely where h is the maximum. See the Supplementary Material 3.6.1 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:

$$\langle f(s) \rangle^{**} = \sum_n \frac{\int_s p(\mathbf{s}|\mathbf{y}^{(n)}, \Theta) \, f(\mathbf{s}) \, \delta(s_h \neq 0) \, d\mathbf{s}}{\int_s p(\mathbf{s}|\mathbf{y}^{(n)}, \Theta) \, \delta(s_h \neq 0) \, d\mathbf{s}}, \tag{3.10}$$

which is identical to $\langle \, . \, \rangle^*$ in Equation (3.9) 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 (3.9) and (3.10) allows us to concisely express the update equations for the remaining model parameters:

$$\hat{W}_{hd} = \frac{\langle s_h y_d \rangle^*}{\langle s_h^2 \rangle^*}, \qquad \hat{\pi} = \langle \delta(\mathbf{s}) \rangle, \tag{3.11}$$

$$\hat{\mu}_{\text{pr}} = \langle s_h \rangle^{**}, \qquad \hat{\sigma}^2_{\text{pr}} = \langle (s_h - \hat{\mu}_{\text{pr}})^2 \rangle^{**} \tag{3.12}$$

In this model $\mathbf{W}_h$ can be scaled by an arbitrary factor α when the corresponding s_h is scaled by $\frac{1}{\alpha}$. To prevent $\mathbf{W}$ from becoming arbitrarily large (which would lead to arbitrarily small values of $\mathbf{s}$), common practice is to constrain its columns (each latent cause) $(\mathbf{W}_h)_{h=1}^H$ to have an l_2-norm less than or equal to one. Instead, we constrain all columns $\mathbf{W}_h$ 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 $\hat{\mu}_{\mathrm{pr}}$ to be intuitively more interpretable when comparing results on different data sets 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 expectations with respect to a complex posterior distribution. However, as mentioned in the Introduction to the Chapter, the posterior distribution of a model (linear or nonlinear) with a spike-and-slab prior is strongly multimodal. See Figure 3.2 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.

3.3.2 Exact Gibbs Sampling with Latent Variable Preselection

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 3.2, we take a combined approximate inference approach proposed by Shelton et al. (2011b). Specifically, we design and propose an exact Gibbs sampler for our model in order to draw samples from the unique form of our posterior. We draw samples from the 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:

$$p(s_h|\mathbf{s}_{H\setminus h}, \mathbf{y}, \theta) \propto p(s_h|\theta) \prod_{d=1}^{D} p(y_d|s_h, \mathbf{s}_{H\setminus h}, \theta). \tag{3.13}$$

We see from Equation (3.13) 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, $\prod_{d=1}^{D} p(y_d|s_h, \mathbf{s}_{H\setminus h}, \theta)$, 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:

$$\begin{aligned} &p(y_d|s_h, \mathbf{s}_{H\setminus h}, \theta) \\ &\quad = \mathcal{N}(y_d;\, \max_{h'}\{W_{dh'}s_{h'}\}, \sigma^2) \\ &\quad = \begin{cases} \underbrace{\mathcal{N}(y_d;\, \max_{h'\setminus h}\{W_{dh'}s_{h'}\}, \sigma^2)}_{\text{constant}} & \text{if } s_h < P_d \\ \mathcal{N}(y_d;\, W_{dh}s_h, \sigma^2) & \text{if } s_h \geq P_d, \end{cases} \end{aligned} \tag{3.14}$$

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

$$P_d = \frac{\max_{h' \in \{H\setminus h\}}\{W_{dh'}s_{h'}\}}{W_{dh}}. \tag{3.15}$$

We refer to the two pieces of y_d in Equation (3.14) 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 3.3**A-B**), because s_h is indeed explaining the data. Taking the logarithm of $p(y_d|s_h, \mathbf{s}_{H\setminus h}, \theta)$ transforms equation (3.14) into a left-piece constant and right-piece quadratic function. Expanding the expression for the logarithm of a given likelihood $p(y_d|s_h, \mathbf{s}_{H\setminus h}, \theta)$, each

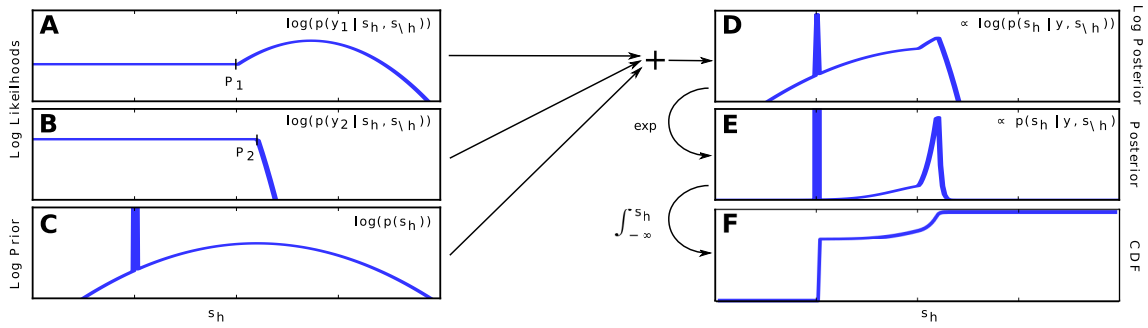


Figure 3.3: Construction of SSMCA-induced posterior for the Gibbs sampler. *Left column: three contributing factors for the posterior* $\propto p(s_h \mid s_{\setminus h}, \mathbf{y}, \Theta)$ *in log space.* **A** *and* **B***: Log likelihood functions each defined by a transition point* P_d *and left 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.*

left and right piece (the respective sides of each transition point P_d) can be formulated as

$$l_d(s_h) = -\frac{1}{2}\log(2\pi) - \log(\sigma) + \frac{1}{2\sigma^2}(y_d - \max_{h' \setminus h}\{W_{dh'} s'_h\})^2 \tag{3.16}$$

$$r_d(s_h) = -\frac{1}{2}\log(2\pi) - \log(\sigma) + \frac{1}{2\sigma^2}(y_d - W_{dh} s_h)^2, \tag{3.17}$$

or more compactly

$$n_d(s_h) = \begin{cases} l_d(s_h) & \text{if } s_h < P_d \\ r_d(s_h) & \text{if } s_h \geq P_d, \end{cases} \tag{3.18}$$

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 (3.14) to consider all observed D dimensions in $\mathbf{y}$. We take the logarithm of $\prod_{d=1}^{D} p(y_d | s_h, \mathbf{s}_{H\setminus h}, \theta)$, 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 $m_d(s_h)$ 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:

$$\sum_{d}^{D} \log p(y_d|s_h, \mathbf{s}_{H\setminus h}, \theta) \tag{3.19}$$

$$= m(s_h) \tag{3.20}$$

$$= \begin{cases} m_1(s_h) & s < P_{\delta(1)} \\ m_2(s_h) & P_{\delta(1)} \leq s < P_{\delta(2)} \\ m_3(s_h) & P_{\delta(2)} \leq s < P_{\delta(3)} \\ \vdots & \vdots \\ m_{D+1}(s_h) & P_{\delta(D)} \leq s. \end{cases} \tag{3.21}$$

Importantly, we observe from Equations (3.16) and (3.17) 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 (3.19) 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 (3.14), we can compactly formulate (3.19) with

$$m_d(s_h) = \sum_{j=1}^{d-1} r_{\delta(j)}(s_h) + \sum_{u=d}^{D} l_{\delta(u)}(s_h). \tag{3.22}$$

$$= \sum_{d'=1}^{D} n_{d'}(s_h) \tag{3.23}$$

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

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 3.3**F**), where the height of the step is proportional to the marginal probability $p(s_h = 0|\mathbf{s}_{\setminus h})$. Once the CDF is constructed, we simulate each s_h from the exact conditional distribution ($s_h \sim p(s_h|\mathbf{s}_{\setminus h} = \mathbf{s}_{\setminus h}, \mathbf{y}, \theta)$) by inverse transform sampling. Figure 3.3 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. We do this by using the same optimization method introduced in Chapter 1.1 and used in the Select and Sample process in Chapter 2. With latent variable preselection, the posterior distribution $p(\mathbf{s} \,|\, \mathbf{y}^{(n)}, \Theta)$ can be approximated by a truncated distribution $q_n(\mathbf{s}; \Theta)$ computed over a reduced latent state space:

$$\begin{aligned} & p(\mathbf{s}|\mathbf{y}^{(n)}, \Theta) \\ & \quad \approx q_n(\mathbf{s}; \Theta) = \frac{p(\mathbf{s}, \mathbf{y}^{(n)}|\, \Theta)\, \mathbb{I}(\mathbf{s} \in \ \mathcal{K}_n)}{\sum\limits_{\mathbf{s}' \in \mathcal{K}_n} p(\mathbf{s}'^{(n)}, \mathbf{y}^{(n)}|\, \Theta)}, \end{aligned} \tag{3.24}$$

where $\mathcal{K}_n$ contains the latent states of the relevant variables for data point $\mathbf{y}^{(n)}$, and $\mathbb{I}(\mathbf{s} \in \mathcal{K}_n) = 1$ if $\mathbf{s} \in \mathcal{K}_n$ and 0 otherwise. To recap, the subsets $\mathcal{K}_n$ are chosen in a data-driven way using a deterministic *selection function*, they vary per data point $\mathbf{y}^{(n)}$, and should contain most of the probability mass $p(\mathbf{s} \,|\, \mathbf{y})$ while also being significantly smaller than the entire latent space. We define $\mathcal{K}_n = \{\mathbf{s} \,|\, \text{for all } h \notin \mathcal{I} :\ s_h = 0\}$ where $\mathcal{I}$ contains the indices of the latents estimated to be most relevant for $\mathbf{y}^{(n)}$. The selection function we use for this model to obtain these latent indices is the cosine similarity:

$$\mathcal{S}_h(\mathbf{y}^{(n)}) \ = \ \frac{\mathbf{W}_h^T\, \mathbf{y}^{(n)}}{||\mathbf{W}_h||_2} \tag{3.25}$$

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

3.4 Experiments

The above described 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), where we compute the E-step approximately using variable preselection and our exact Gibbs sampler. 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. $\mathbf{W}$ is initialized at the observed data mean with

additive Gaussian noise of the σ observed in the data.

3.4.1 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) reliably converge 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 $H = 10$ latent variables according to our model: N images with overlapping 'bars' of varying intensities and with Gaussian observation noise of variance $\sigma_{gt} = 2$ (Figure 3.4**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 variable 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 3.4**B**,**E**) show that our algorithm converges quickly and learns 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 3.4**C**,**D**,**E**) for some further convergence examples, namely where $H' = 4$ and $H' = 5$.

3.4.2 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 data sets consisting of true occlusions. Here the task is to learn the set of components $\mathbf{W}$, i.e. the dictionary elements, that are behind the composition of a given observed data set $\mathbf{y}$, and consider reconstruction of individual images/data points $\mathbf{y}^{(n)}$. The goal of these experiments is to understand

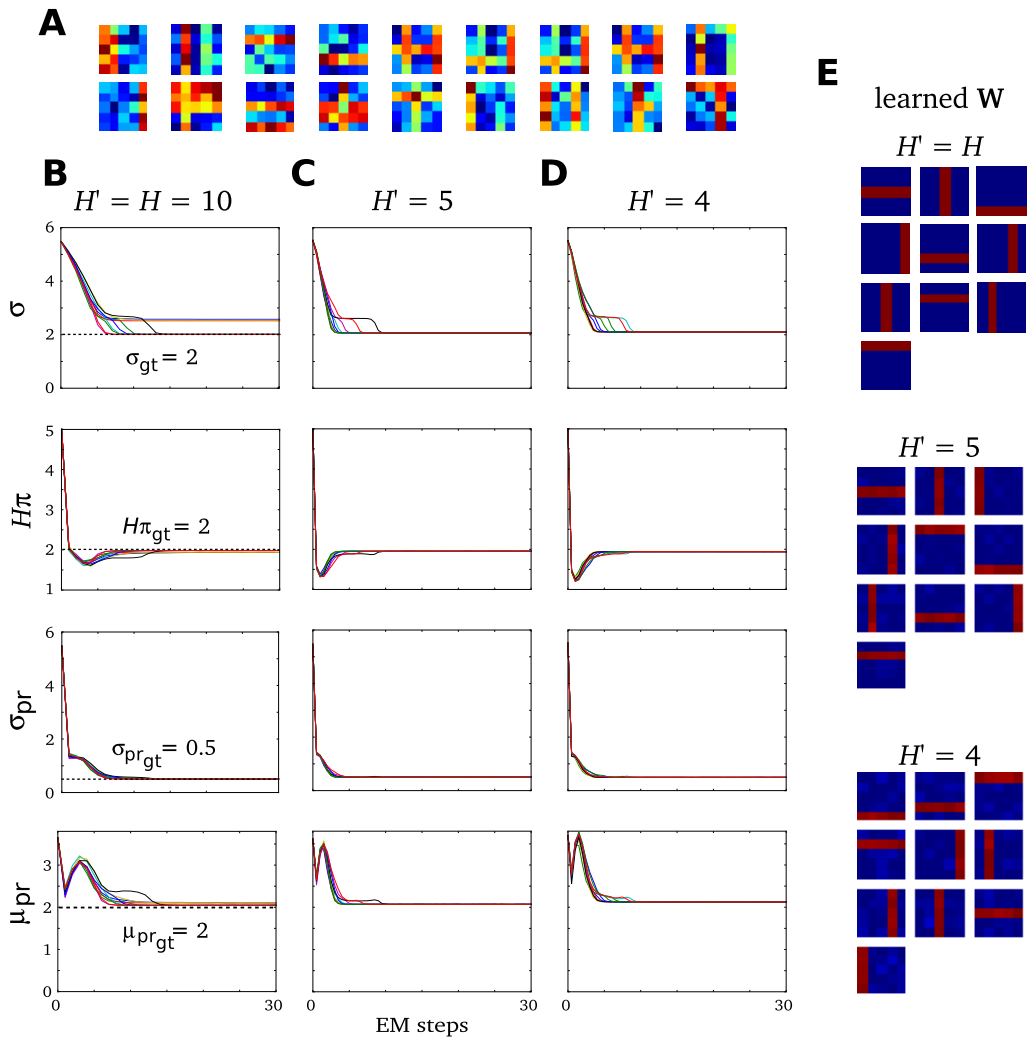


Figure 3.4: Parameter recovery on synthetic data. *Results of three differently parameterized sets of experiments, each with 10 experimental runs of 30 EM iterations on identical artificial ground-truth data generated according to the SSMCA model:* **A** $N = 2,000$, $D = 5 \times 5$. *Three experimental settings, each preselecting a different number of latent variables, are shown:* **B** $H' = H = 10$, **C** $H' = 5$, *and* **D** $H' = 4$, *although the same results were obtained by the entire range of preselection 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* $\mathbf{W}_h$ *corresponding to each experimental set* **B** $H' = H = 10$, **C** $H' = 5$, *and* **D** $H' = 4$.

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 (3.1). 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., 2013b; 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). However, the algorithms for parameter optimization remain different also in this limit (SSMCA$^{\text{fix}}$ remains sampling based, for instance).

Realistic Occlusion data set

The first data set we compare the algorithms on is one with controlled forms of sparse structure, a realistic artificial data set of true occlusions (data created by actual occlusions and not following any model considered here). 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 $N = 61,009$. Gaussian observation noise of $\sigma = 25$ and $\mu = 0$ was then independently added to each patch. Examples are shown in Figure 3.5. Additionally, the data set contains the corresponding (automatically obtained) labels for each image, indicating the *ground-truth number of occluding strokes* $k \in (0, 5)$ per image (for e.g. optional use in performance evaluation).

Such a data set 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 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.

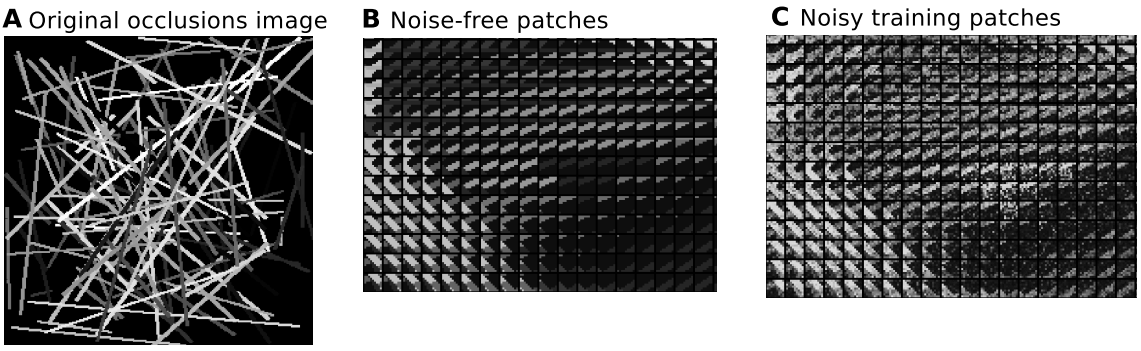


Figure 3.5: Synthetic occlusion data set and cut-out original and noisy patches. *Examples taken from the occlusion data set.* ***A*** *shows an original noise-free image of generated occluding strokes of random width, pixel intensity, and starting/ending points.* ***B*** *shows a handfull of overlapping 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****.*

We set the number of dictionary elements to be learned from the data set 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 data point, per variable (i.e. $40 \times 100 = 40000$ samples per data point 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 the linear objective (3.1) in order to evaluate the reconstruction and the components learned across a range of sparse solutions. For regularization, α enforces the sparsity of a solution, where a small α indicates less penalization of learning a non-sparse solution and thereby the learning of a large set of components (and vice versa for large values of α).

The results showcase a number of notable effects. First, we see in Figure 3.6**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 3.6**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, $\sum_n (\mathbf{x}_n - \hat{\mathbf{x}}_n)^2$, or the mean MSE, MMSE, which is MSE averaged over the respective data set), 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.

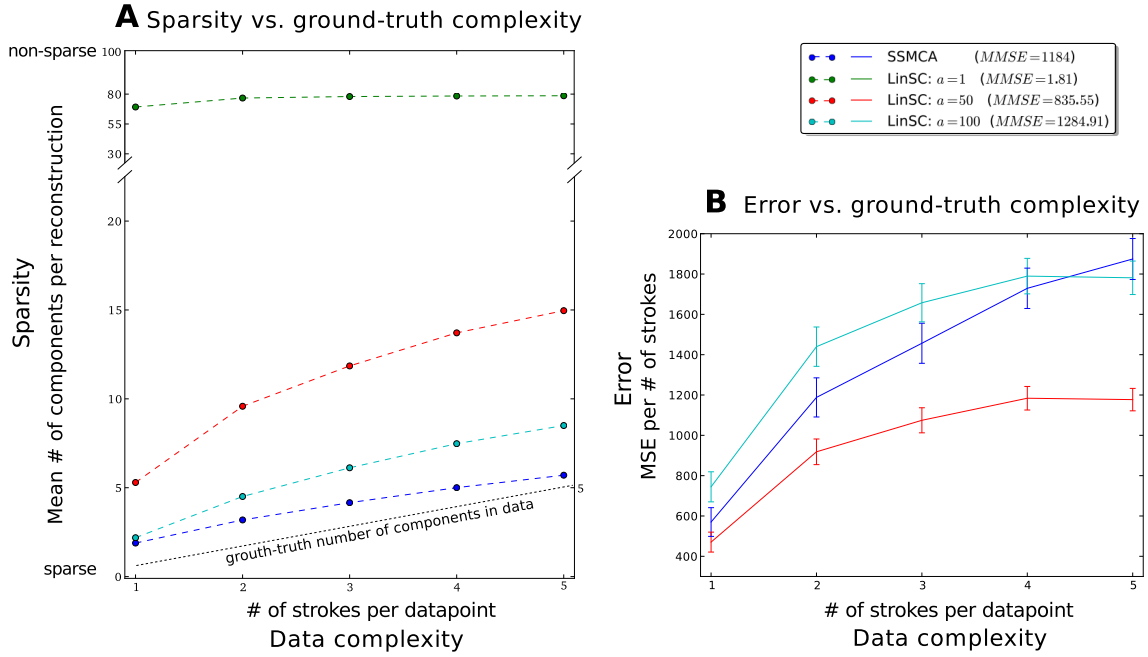


Figure 3.6: Comparative experiments of linear and nonlinear sparse coding on dictionary learning and image reconstruction. *With $H = 100$ learned dictionary components we evaluate the number learned and used for reconstruction.* **A** *shows the relationship between sparsity (number of components used 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 data point'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 (MMSE; averaged over the entire data set) is shown in the legend next to the respective algorithm.*

Next, we investigate the actual components each model uses in order to reconstruct a given image patch. Figures 3.7**A-E** contains a comparison of the reconstruction of a handful of image patches by the linear and the nonlinear methods. 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 (3.1)), 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 those learned by SSMCA while they look quite different than those learned by linear sparse coding (see Figure 3.8 for some examples). The learned sparsity is also similar to the one learned by SSMCA but we observed only weak scaling with the complexity of the patches (for $\sigma_{\text{pr}} \geq 0.25$) to no scaling (for $\sigma_{\text{pr}} \leq 0.25$). Also the sparsity values were consistently higher for SSMCA$^{\text{fix}}$ compared to SSMCA (i.e. fewer components for SSMCA$^{\text{fix}}$). Furthermore, the average 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 significant increase in reconstruction error is due to a decreased ability to fine-tune dictionary coefficients to the intensities of the components – the intensity value range the coefficients can use is lower. For the SSMCA$^{\text{fix}}$ algorithm this also seems to indirectly influence the learned sparsity, maybe due to SSMCA$^{\text{fix}}$ attributing components with a low pixel intensity to background noise. We observe the reconstruction errors and sparsity to increase when we decrease the width of the fixed slab (namely, when decreasing σ_{pr}).

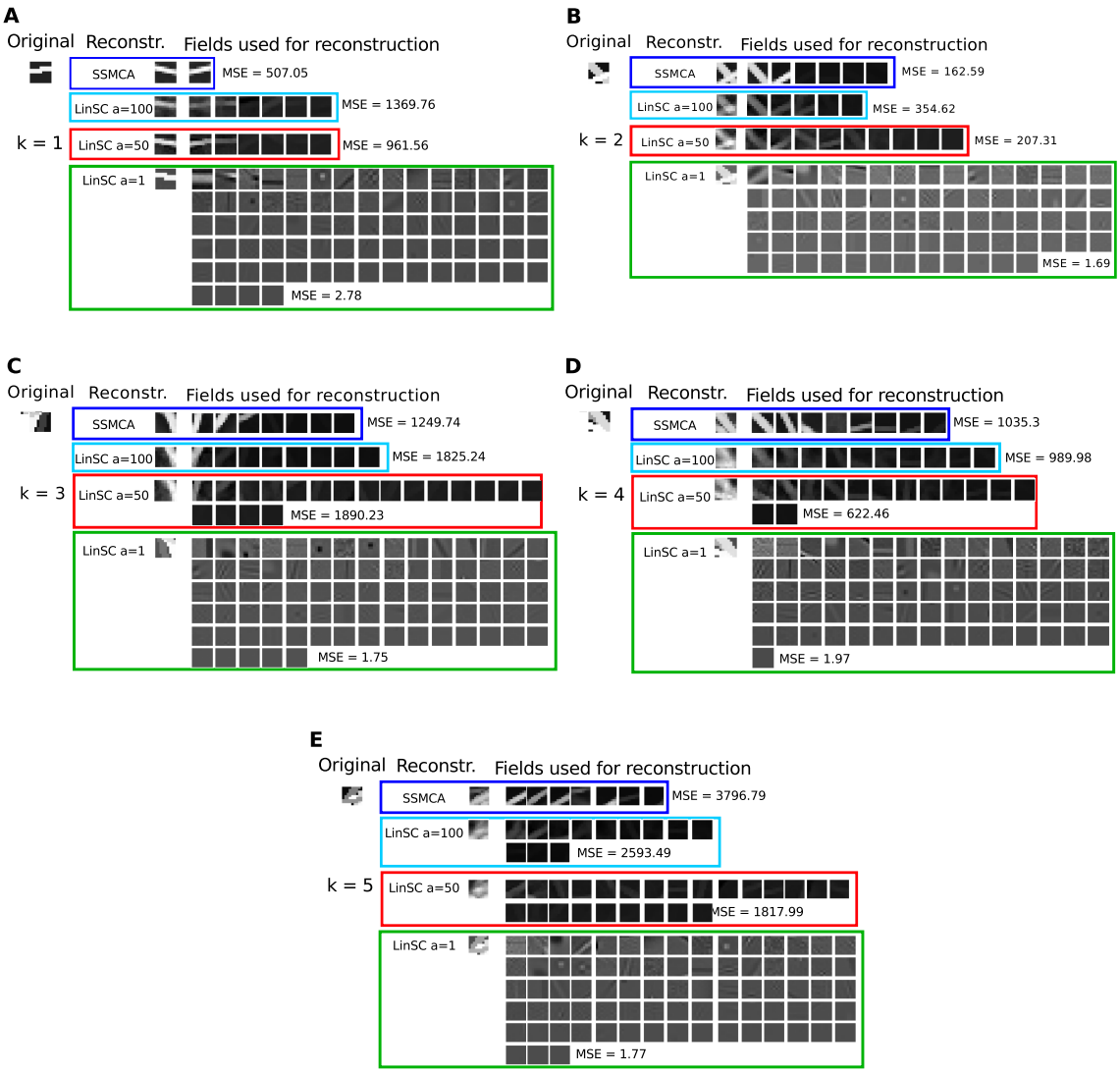


Figure 3.7: Comparison of linear and nonlinear sparse coding on image reconstruction. *Shown are a handfull of real data points of varying complexity in terms of the number of strokes k in each image ($k \in (1, 5)$ strokes per image), the components/fields learned by the various algorithms, the corresponding reconstruction of the given data point, and the mean squared error (MSE) of each reconstruction. Image complexities shown are:* **A** *$k = 1$ stroke,* **B** *$k = 2$ strokes,* **C** *$k = 3$ strokes,* **D** *$k = 4$ strokes, and* **E** *$k = 5$ strokes. Regardless of image complexity k, 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 data point (and all other shown cases), hence the reconstruction error (MSE) is very low.*

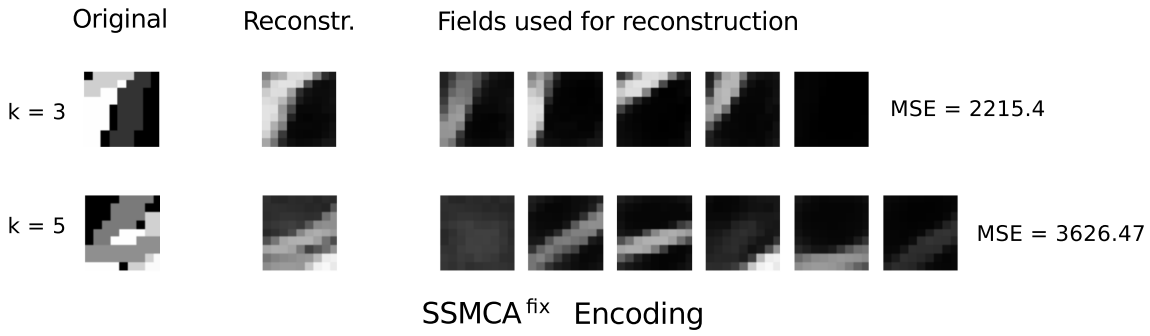


Figure 3.8: Results of nonlinear sparse coding using a binary prior on image reconstruction. *The nonlinear sparse coding model if applied to artificial strokes using a fixed narrow slab (SSMCA$^{\text{fix}}$). The two figure columns show image reconstruction results for SSMCA$^{\text{fix}}$ with $\sigma_{\text{pr}} = 0.25$ for two different ground-truth stroke numbers: $k = 3$ in the first row and $k = 5$ in the second row. SSMCA$^{\text{fix}}$ was first trained with fixed σ_{pr} and then applied to the data. Reconstructions were computed as described for Figure 3.7.*

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 gray-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 reconstruction 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 3.6 and 3.7.

Natural Image Occlusions

We have shown that our approach can model realistic artificial occlusions well. Now we are interested in investigating the performance of the linear and nonlinear approaches on naturally occurring occlusions. We use an image of underbrush in a forest (taken from 'bridge.jpg', which has been used for denoising benchmarking (Mairal et al., 2009b)),

which is rich with occluding branches and twigs. See Figure 3.9**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 data set, shown in Figure 3.9**B**. To compose the data set as in the previous experiments, we cut the 256×256 image, with pixel values x_i ranging from $(0, 255)$, into $N = 61,009$ 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 data set, with both the nonlinear and linear methods learning a dictionary size of $H = 100$ latent variables. For SSMCA we again draw 40 samples per data point, per variable (i.e. 40×100 samples per data point), 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 $a = (1, 50, 100)$.

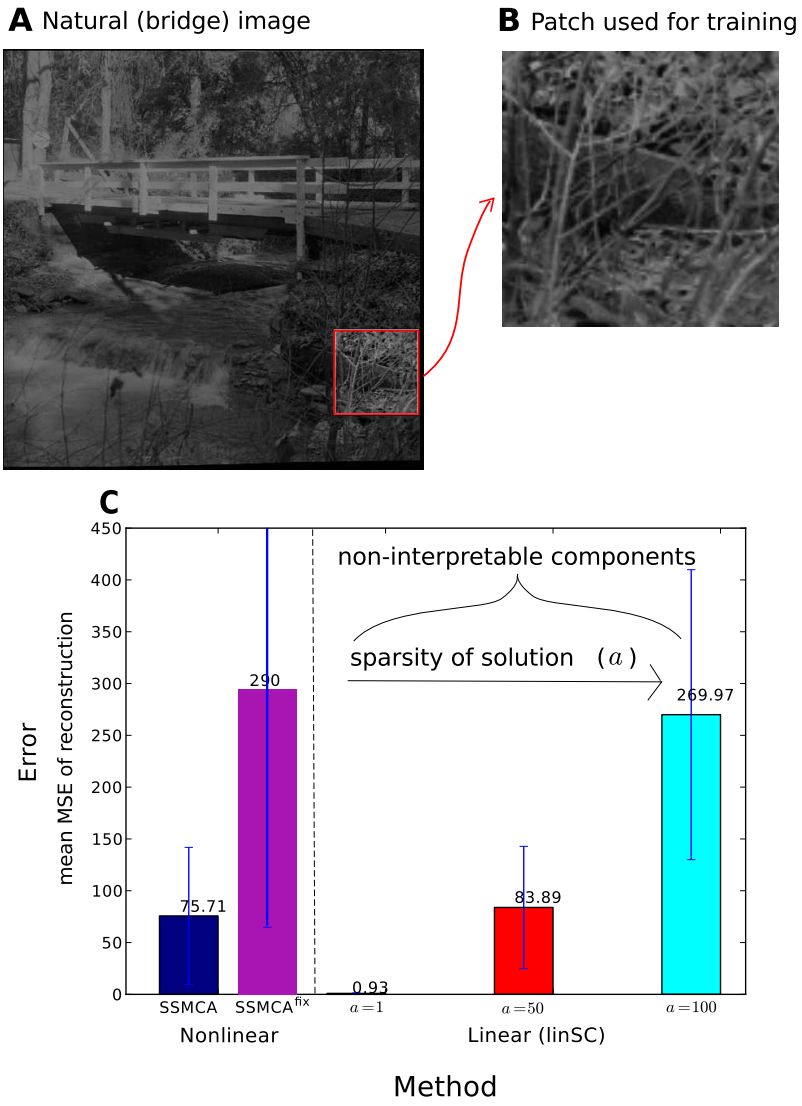


Figure 3.9: Results of comparative experiments of linear and nonlinear sparse coding methods on component learning/image reconstruction on natural image patches. *Experiments evaluated are the nonlinear SSMCA and SSMCA*$^{\text{fix}}$ *(where the 'slab' is fixed to be narrow, i.e. resulting in a spike-and-slab prior distribution closer to a binary distribution), and the linear model with a range of regularization parameters* α *to enforce the sparsity of a solution (small* α *leads to less sparse solutions and larger* α *to more sparse solutions).* ***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 data set.* ***C*** *shows the mean squared error (MSE) of the compared nonlinear and linear methods' reconstruction averaged over the entire data set, with the standard deviation indicated with error bars. The trend is the same as in the artificial occlusions data experiments: the nonlinear method maintains reasonably low MSE, while learning a sparse set of interpretable components, whereas the linear method achieves a very low MSE only when it does not learn a sparse (and never interpretable) solution of components.*

Because we do not have any ground-truth associated with this data set as to how many strokes/components are in a given image, we can only compare the average reconstruction error (MMSE) for the entire data set across methods. Figure 3.9**C** shows the mean MSE of each method with the associated standard deviation. The results follow the trend outlined in the previous set of experiments (in Figure 3.6**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 components for reconstruction), it can fine-tune its reconstruction to achieve very low error (MMSE $= 0.93$). However, the components each linear model uses for reconstruction are non-interpretable (i.e. 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 3.6 for comparison). When we make the slab narrower still, 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.

3.4.3 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 basis 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 variables/generative fields, and run 50 EM iterations with 100

samples per data point. 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 data point 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 3.10. In Figure 3.10**A**, we have a handful of the learned dictionary elements $\mathbf{W}_h$ (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 the Supplementary Material Results 3.6.2 for details). Next, we classify the fields as either orientation-sensitive Gabor wavelets or 'globular' fields best matched by DoGs. In Figure 3.10**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 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 3.10**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 3.10**D** shows a handfull of the inferred latent variables (coefficients) s_h. These correspond to the actual activations of the diverse dictionary elements $\mathbf{W}_h$, each of which is visualized in the upper right of each subfigure. Please see the Supplementary Material 3.6.2 for the complete set of generative fields learned and for a larger set of the learned prior activations.

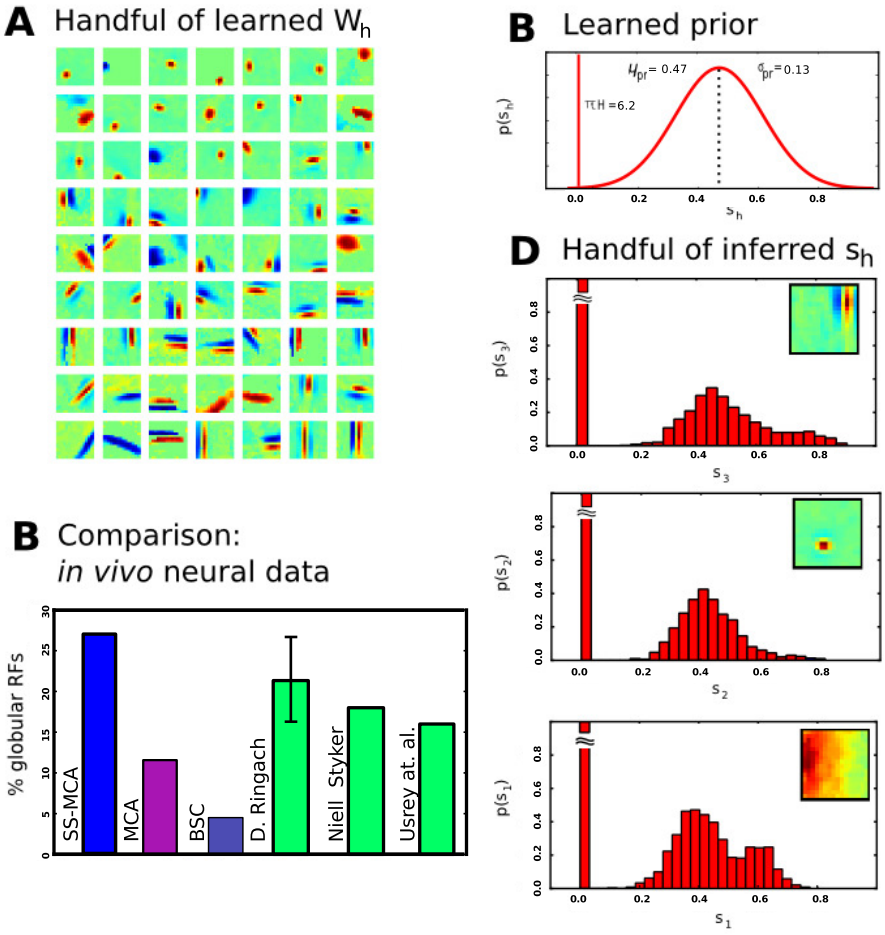


Figure 3.10: Analysis of dictionary components learned by the SSMCA algorithm on natural image patches. **A** *Example dictionary elements* $\mathbf{W}_h$ *after learning.* **B** *Fraction of globular fields estimated from in vivo measurements, compared to ours (after fitting with Gabor wavelets and 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).*

Since we have shown consistent predictions with neural recordings, we finally test the model for consistency with the natural image patches data set. 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 data points $\mathbf{y}^{(n)}$ matches the prior (compare e.g. (Berkes et al., 2011a)):

$$\lim_{N\to\infty} \frac{1}{N} \sum_n p(\mathbf{s} \,|\, \mathbf{y}^{(n)}, \Theta) = p(\mathbf{s} \,|\, \Theta). \tag{3.26}$$

After the learning on image patches as described above, we observed that posteriors averaged over data points closely resemble the learned prior (see Figure 3.10E for examples). Linear sparse coding has reportedly struggled with this consistency condition (see (Olshausen and Millman, 2000) for a discussion).

3.5 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 properties. Particularly, the nonlinearity of the $\max$ equips the approach to well-approximate occlusions.

The main technical contribution of this work was a method to make inference possible in a model with these two modifications. We proposed an exact Gibbs sampler (to use in the context of Select and Sample) that constructed 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.

A main focus of 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 have demonstrated 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 showed, 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*. Finally, 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 level of corresponding reconstruction error (MSE).

Our results consequently show that the $\max$ nonlinearity is sufficient to reproduce many properties desired from a latent variable approach to image patch modelling – especially "interpretable" encoding. While explicit occlusion models can be developed based on similar methods 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) is still a research domain posing many difficult scientific questions.

Although we have observed that SSMCA can successfully model the generation process of some occlusion-rich data, it is currently limited in its applicability to functional tasks. SSMCA extracts components that resemble the low-level image effects that generate occlusions in images, e.g. the overlapping edges/branches in a picture of tree branches. However, when the goal is to achieve consistently nominal error in reconstructing the original image (here this was measured by the mean squared error), the SSMCA model is a poor choice. SSMCA is catered to find a holistic sparse solution set that is easily interpretable, such as an individual branch edges in a tree. Linear models on the other are capable of learning a less sparse solution set that, albeit non-interpretable, yields consistently very low reconstruction error – summing as many components as necessary for optimal image reconstruction. From this it follows that, in its current form at least, SSMCA cannot outperform leading linear models on tasks that have image reconstruction at its core such as inpainting or denoising. Confirming this, in preliminary denoising and inpainting experiments (not included in this thesis), SSMCA did not outperform current methods but was nonetheless comparable to the results of some leading linear models (Lázaro-gredilla and Titsias, 2011; Zhou et al., 2009). Extensions targeting image restoration and reconstruction could be explored in future work.

The model is somewhat limited in terms of the complexity of the data it of which it can successfully model. Generalizations of SSMCA could, for example, take into account object depths and ordering of components for a more explicit occlusion modelling. The unconstrained object permutations in that case, however, lead to super-exponential scaling of hidden states, making inference even more challenging. Additionally, further refinements to the model, by e.g. learning individual means for each spike-and-slab and/or learning different noise estimates for each pixel in the image patch could drastically improve its applicability to a wider variety of data and tasks.

3.6 Supplementary Material

3.6.1 M-step Parameter Equation Derivations

The equations computed every EM iteration in the M-step to update the model parameters to the current maximum likelihood solution are shown here with their derivations.

The optimal parameters that maximize the data log likelihood under the generative model can be found using EM algorithm (see e.g. (Neal and Hinton, 1998)), which iteratively optimizes a lower bound $\mathcal{F}(\Theta, q)$ of the likelihood with respect to the parameters Θ and a distribution q:

$$\mathcal{L}(\Theta) \geq \mathcal{F}(\Theta, q_{\Theta'}) = \sum_{n=1}^{N} \sum_{s} q_n(\mathbf{s}|\Theta') \log \frac{p(y^{(n)}, \mathbf{s}|\Theta)}{q_n(\mathbf{s}|\Theta')} \tag{3.27}$$

$$= \langle \log p(\mathbf{y}, \mathbf{s} \,|\, \Theta) \rangle_{q(\mathbf{s}|\Theta')} + \mathrm{H}[q(\mathbf{s}|\Theta')]. \tag{3.28}$$

Each iteration consists of an E-step and an M-step. The E-step optimizes the lower bound with respect to 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 $\mathcal{F}(\Theta, q_{\Theta'})$ with respect to 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:

$$\Theta^{(t+1)} = \mathrm{argmax}_{\Theta} \mathcal{F}(\Theta, q_{\Theta^{(t)}}). \tag{3.29}$$

This is maximised with the maximum likelihood estimate:

$$\Theta^{(t+1)} = \mathrm{argmax}_{\Theta} \langle \log p(\mathbf{y}, \mathbf{s} \,|\, \Theta) \rangle_{q(\mathbf{s}|\Theta^{(t)})}. \tag{3.30}$$

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:

$$y^{(n)} = w s^{(n)} + \varepsilon \tag{3.31}$$

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

$$p(y^{(n)} \,|\, s^{(n)}, w) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2} \left(\frac{y^{(n)} - w s^{(n)}}{\sigma} \right)^2 \right) \tag{3.32}$$

In log space this is a quadratic function:

$$\log p(y^{(n)} \mid s^{(n)}, w) = c - \log \sigma - \frac{1}{2}\left(\frac{y^{(n)} - w s^{(n)}}{\sigma}\right)^2 \tag{3.33}$$

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:

$$\frac{d}{dw}\left[\sum_n \left(y^{(n)} - s^{(n)} w\right)^2\right] = 0. \tag{3.34}$$

From which the maximum is given by:

$$w = \frac{\sum_n s^{(n)} w^{(n)}}{\sum_n s^{(n)2}}. \tag{3.35}$$

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

$$y^{(n)} = \max_h \left\{W_h s_h^{(n)}\right\} + \varepsilon \tag{3.36}$$

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:

$$W_{hd} = \frac{\sum_n^N \sum_k^K \delta(\text{h is max}) s_{hn}^{(k)} y_d^{(n)}}{\sum_n^N \sum_k^K \delta(\text{h is max})\ {s_{hn}^{(k)}}^2} \tag{3.37}$$

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.

3.6.2 Experiments: Natural Image Patches

The complete set of generative fields learned W learned in the experiments on natural image patches and a larger set of the learned prior activations are shown here.

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 corre-

sponding 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 3.11**A** we show the generative fields that are learned with our method. Figure 3.11**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 3.11**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 3.11 we used the receptive field estimates in Figure 3.11**B**. Please see the preliminary study (Shelton et al., 2012a) for more details and analysis of the model's application to response properties in primary visual cortex.

In Figure 3.12, we see a histogram of the latent activations of the 20 most often active dictionary elements, all of which follow a spike-and-slab distribution, consistent with our model's prior beliefs, as defined in Equation (3.26).

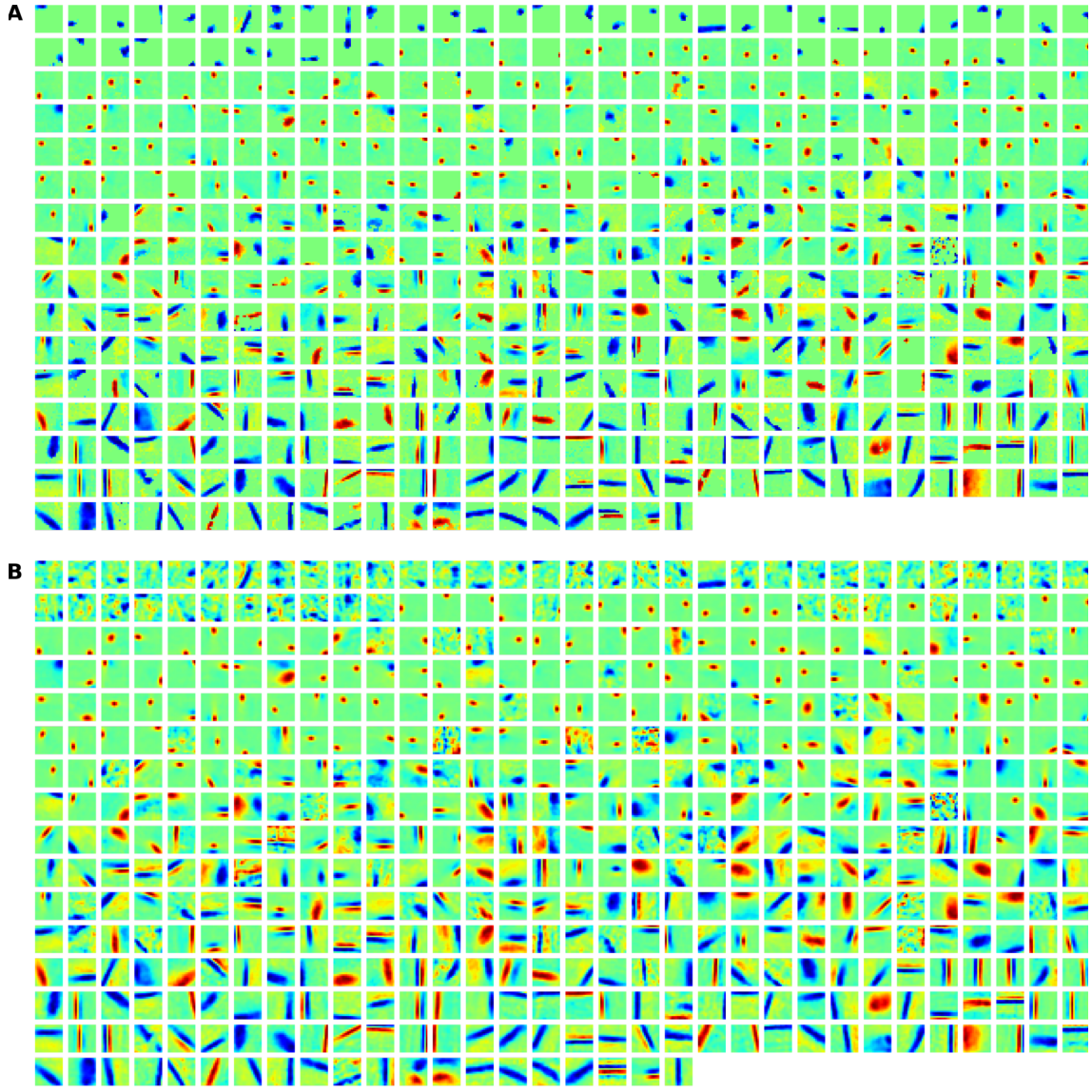


Figure 3.11: **A** *Full set of* $H = 500$ *learned generative fields* ($\mathbf{W}_h$). **B** *Fields after reverse correlation with preprocessed input patches.*

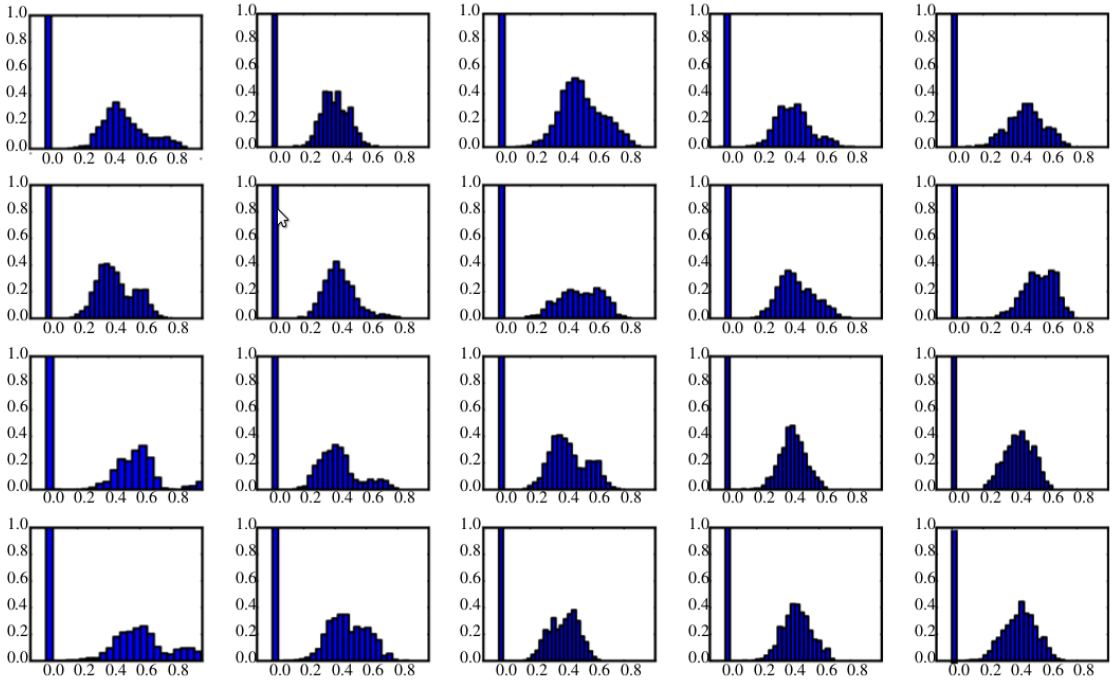


Figure 3.12: *Histogram of the latent activations of the 20 most often active dictionary elements. This corresponds to the prior over latent units that we have assumed in our model, thus supporting the consistency of the model as defined in Equation* (3.26).

Chapter 4

GP-select: Accelerating EM using Adaptive Subspace Preselection

In this Chapter, we introduce a generalization of the preselection optimization method introduced in Chapter 2 and further applied in Chapter 3. Whereas in those chapters, the selection function used to preselect latent variables was hand-engineered for each individual model, here we propose a model-independent nonparametric black-box way to define a suitable selection function efficiently.

The work presented in this Chapter can be found in the following publications: Shelton et al. (2014, 2017).

4.1 Introduction

We propose a nonparametric procedure to achieve fast inference in generative graphical models when the number of latent states is very large. The approach is based on iterative latent variable preselection, where we alternate between learning a 'selection function' to reveal the relevant latent variables, and using this to obtain a compact approximation of the posterior distribution for EM; this can make inference possible where the number of possible latent states is e.g. exponential in the number of latent variables, whereas an exact approach would be computationally infeasible. We learn the selection function entirely from the observed data and current EM state via Gaussian process regression. This is by contrast with earlier approaches, where selection functions were manually-designed for each problem setting. We show that our approach performs as well as these bespoke selection functions on a wide variety of inference problems: in particular, for the challenging case of a hierarchical model for object localization with occlusion, we

achieve results that match a customized state-of-the-art selection method, at a far lower computational cost.

4.2 Related Work

The general idea of aiding inference in graphical models by learning a function that maps from the observed data to a property of the latent variables is quite old. Early work includes the Helmholtz machine (Dayan et al., 1995) and its bottom-up connections trained using the wake-sleep algorithm (Hinton et al., 1995). More recently, the idea has surfaced in the context of learning variational distributions with neural networks (Kingma and Welling, 2014). A two-stage inference procedure has been discussed in the context of computer vision (Yuille and Kersten, 2006) and neural inference (Körner et al., 1999). Recently, researchers (Mnih and Gregor, 2014) have generalized this idea to learning in arbitrary graphical models by training an "inference network" that efficiently implements sampling from the posterior distribution.

GPs have recently been widely used to "learn" the results of complicated models in order to accelerate inference and parameter selection. GP approximations have been used in lieu of solving complex partial differential equations (Sacks et al., 1989; Currin et al., 1991), to learn data-driven kernel functions for recommendation systems (Schwaighofer et al., 2004), and recently for quantum chemistry (Rupp et al., 2012). Other work has used GPs to simplify computations in approximate Bayesian computation (ABC) methods: namely to model the likelihood function for inference (Wilkinson, 2014), to aid in making Metropolis-Hastings (MH) decisions (Meeds and Welling, 2014), and to model the discrepancies between simulated/observed data in parameter space simplification (Gutmann and Corander, 2015). Recently, instead of the typical choice of GPs for large scale Bayesian optimization, neural networks have been used to learn an adaptive set of basis functions for Bayesian linear regression (Snoek et al., 2015).

Our work follows the same high level philosophy in that we use GPs to approximate complex/intractable probabilistic models. None of the prior work cited addresses our problem setting, namely the selection of relevant latent variables by learning a nonparametric relevance function, for use in Expectation Truncation (ET).

4.3 Variable Selection for Accelerated Inference

Notation. We denote the observed data by the $D \times N$ matrix $\mathbf{Y} = (\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(N)})$, where each vector $\mathbf{y}^{(n)} = (y_1^{(n)}, \ldots, y_D^{(n)})^{\mathrm{T}}$ is the nth observation in a D-dimensional space. Similarly we define corresponding binary latent variables by the matrix $\mathbf{S} = (\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(N)}) \in \{0,1\}^{H \times N}$ where each $\mathbf{s}^{(n)} = (s_1^{(n)} \ldots, s_H^{(n)})^{\mathrm{T}} \in \{0,1\}^H$ is the nth vector in the H-dimensional latent space, and for each individual hidden variable $h = 1, \ldots, H$, the vector $\mathbf{s}_h = (s_h^{(1)} \ldots, s_h^{(N)}) \in \{0,1\}^N$. Reduced latent spaces are denoted by H', where $H' \ll H$. Note that although we restrict ourselves to binary latent variables here, the procedure could in principle be generalized to variables with higher cardinality (e.g. see (Exarchakis et al., 2012)). We denote the prior distribution over the latent variables as $p(\mathbf{s}|\theta)$ and the likelihood of the data as $p(\mathbf{y}|\mathbf{s}, \theta)$. Using these expressions, the posterior distribution over latent variables is

$$p(\mathbf{s}^{(n)}|\mathbf{y}^{(n)}, \Theta) = \frac{p(\mathbf{s}^{(n)}|\Theta)\, p(\mathbf{y}^{(n)}|\mathbf{s}^{(n)}, \Theta)}{\sum_{\mathbf{s}'^{(n)}} p(\mathbf{s}'^{(n)}|\Theta)\, p(\mathbf{y}^{(n)}|\mathbf{s}'^{(n)}, \Theta)}. \tag{4.1}$$

4.3.1 Selection via Expectation Truncation in EM

As discussed in the previous chapters, EM is an iterative algorithm to optimize the model parameters of a given graphical model. EM iteratively optimizes a lower bound on the data likelihood by inferring the posterior distribution over hidden variables given the current parameters (the E-step), and then adjusting the parameters to maximize the likelihood of the data averaged over this posterior (the M-step). When the number of latent states to consider is large (e.g. exponential in the number of latent variables), the computation of the posterior distribution in the E-step becomes intractable and approximations are required.

ET is a meta algorithm, which improves convergence of the EM algorithm (Lücke and Eggert, 2010). The main idea underlying ET is that the posterior probability mass is concentrated in a small subspace of the full latent space. This is the case, for instance, if for a given data point $\mathbf{y}^{(n)}$ only a subset of the H latent variables $s_h^{(n)}, s_1^{(n)}, s_2^{(n)}, \ldots, s_H^{(n)}$ are relevant. Even when the probability mass is supported everywhere, it may still be largely concentrated on a small number of the latents.

A *selection function* can be used to identify a subset of salient variables, denoted by H' where $H' \ll H$, which in turn is used to define a subset of states, denoted $\mathcal{K}_n$, of the possible state configurations of the space per data point. State configurations not in this

space (of variables deemed to be non-relevant) are fixed to 0 (assigned zero probability mass). The posterior distribution in Equation (4.1) can then be approximated by a truncated posterior distribution, computed on the reduced support,

$$p(\mathbf{s}^{(n)}|\mathbf{y}^{(n)},\Theta) \approx q_n(\mathbf{s}^{(n)};\Theta) = \frac{p(\mathbf{s}^{(n)},\mathbf{y}^{(n)}|\,\Theta)\,\mathbb{I}(\mathbf{s}^{(n)} \in \mathcal{K}_n)}{\sum_{\mathbf{s}'^{(n)}\in\mathcal{K}_n} p(\mathbf{s}'^{(n)},\mathbf{y}^{(n)}|\,\Theta)}, \tag{4.2}$$

where $\mathcal{K}_n$ contains the latent states of the H' relevant variables for data point $\mathbf{y}^{(n)}$, and $\mathbb{I}(\mathbf{s} \in \mathcal{K}_n) = 1$ if $\mathbf{s} \in \mathcal{K}_n$ is true, and 0 otherwise. In other words, Equation (4.2) is proportional to Equation (4.1) if $s \in \mathcal{K}_n$ (and zero otherwise). The set $\mathcal{K}_n$ contains only states for which $s_h = 0$ for all h that are not selected, i.e. all states where $s_h = 1$ for non-selected h are assigned zero probability, and the indices of the selected H' variables are collected in $\mathcal{I}$ such that we can define $\mathcal{K}_n = \{\mathbf{s}\,|\,\text{for all } h \notin \mathcal{I}:\ s_h = 0\}$. This means that there are fewer terms in the denominator of Equation (4.2) compared with Equation (4.1), which affects the overall scaling of the terms. Equation (4.2) still remains proportional to Equation (4.1) for the selected terms $s \in K_n$, however. The sum over $\mathcal{K}_n$ is much more efficient than the sum for the full posterior, since it only needs to be computed over the reduced set of latent variable states deemed relevant: the state configurations of the irrelevant variables are fixed to be zero. The larger the variable selection parameter H' is, the closer the approximation will be to true EM, but available computational resources dictate how large it can be. However as mentioned in Chapter 1, much smaller values of H' have been empirically shown to achieve high accuracy.

4.3.2 ET with Affinity

One way of constructing a selection function is by first ranking the latent variables according to an *affinity function* $f_h(\mathbf{y}^{(n)}) : \mathrm{R}^D \mapsto \mathrm{R}$ which directly reflects the relevance of the latent variable s_h. A natural choice for such a function is the one that approximates the marginal posterior probability of each variable. We try to learn f as follows:

$$f_h(\mathbf{y}^{(n)}) = \hat{p}_h^{(n)} \approx p_h^{(n)} \equiv p(s_h^{(n)} = 1|\mathbf{y}^{(n)},\Theta), \tag{4.3}$$

meaning that the relevant variables will have greater marginal posterior probability p_h. See Figure 4.1 for a simplified illustration. When the latent variables $s_{h=1}^{(n)},\ldots,s_H^{(n)}$ in the marginal posterior probability $\hat{\mathbf{p}}^{(n)} = \hat{p}_{h=1}^{(n)},\ldots,\hat{p}_H^{(n)}$ are conditionally independent given a data point $\mathbf{y}^{(n)}$, this affinity function correctly isolates the most relevant variables in the posterior. To see this, consider the full joint $p(s_1,...s_h\,|\,\mathbf{y},\Theta)$ in the case when a subset of

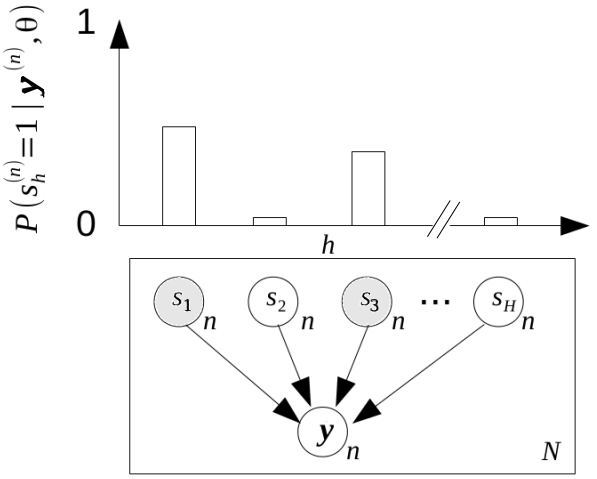


Figure 4.1: *Illustration of the affinity function for selection. The affinity approximates the marginal posterior probability of each $h = 1, \ldots, H$ latent variable (top), which identifies the most relevant variables for a given data point $\mathbf{y}^{(n)}$ (bottom). Here, the variables s_1 and s_3 yield high affinity and would thus be considered relevant for $\mathbf{y}^{(n)}$.*

latents has values clamped to zero, i.e., $s_h = 0$ for all $h \notin \mathcal{I}$ (compare Equation (4.2)). We can then ask what the overall joint posterior mass is in this case. If we suppose the latents to be conditionally independent, this total mass is given by a product of marginals as follows:

$$\sum_{\mathbf{s} \text{ with } s_h=0 \text{ for all } h \notin \mathcal{I}} p(s_1, \ldots s_H \,|\, \mathbf{y}, \Theta) = \prod_{h \notin \mathcal{I}} (1 - p(s_h = 1 \,|\, \mathbf{y}, \Theta)). \tag{4.4}$$

We want this mass to be as large as possible as its complement is the posterior mass that we discard with our approximation. If the affinity function correctly estimates the marginals $p(s_h = 1 \,|\, \mathbf{y}, \Theta)$, then discarding those $(H - H')$ marginals with the lowest values is equivalent to discarding the space with the least posterior mass (compared to discarding w.r.t. all alternative choices with the same number of latents). Even when this strong assumption does not hold in practice (which is often the case) however, the affinity can still correctly highlight relevant variables, and has been empirically shown to be quite effective when dependencies exist (see e.g. the source separation tasks in (Sheikh et al., 2014)).

Next, using all $\hat{p}^{(n)}_{h=1}, \ldots, \hat{p}^{(n)}_{H}$ from the affinity function $\mathbf{f}(\mathbf{y}^{(n)}) = (f_1(\mathbf{y}^{(n)}), \ldots, f_H(\mathbf{y}^{(n)}))$, we define $\gamma\left(\hat{\mathbf{p}}^{(n)}\right)$ to simultaneously sort the indices of the latent variables in descending order [of probability $\hat{p}^{(n)}$] and *reduce* the sorted set to the H' highest (most relevant) variables' indices. To ensure that there is a non-zero probability of selecting each variable per EM iteration, 10% of the H' indices are uniformly chosen from H at random (as noted in Section 2.5). This prevents the possible propagation of errors from $q_{(n)}$ continuously

assigning small probabilities to a variable s_h in early EM iterations, because the optimization of $q_{(n)}$ in early iterations starts from randomly initialized $\mathbf{s}_h$. $\gamma(\hat{\mathbf{p}}^{(n)})$ thus returns the H' selected variable indices I deemed by the affinity to be relevant to the nth data point. Finally, using the indices I from γ, we define $\mathcal{I}(I)$ to return an H'-dimensional subset of selected relevant latent states $\mathcal{K}_n$ for each data point $\mathbf{y}^{(n)}$. All 'non-relevant' variable states s_h for all variables $h \notin I$ are effectively set to 0 in Equation (4.2) by not being present in the state set $\mathcal{K}_n$. For example, let's say that there are five s_h, where $h \in \{1, ..., 5\}$. We consider the case where only s_1 and s_2 are selected. The $\mathcal{I}$ function will then return zeros for s_3, s_4, and s_5, but will return both allowed possibilities 0 or 1 for s_1 and s_2. Thus a valid setting for the entire vector $\mathbf{s}$ can be $\mathbf{s} = [01000]$, but not $\mathbf{s} = [01100]$.

Using $\mathbf{f}$, $\mathcal{I}$, and γ, we can define a *selection function* $\mathcal{S} : \mathrm{R}^D \mapsto 2^{\{1,...,H\}}$ to select subsets $\mathcal{K}_n$ per data point $\mathbf{y}^{(n)}$. Again, the goal is for the states $\mathcal{K}_n$ to contain most of the probability mass $p(\mathbf{s} \,|\, \mathbf{y})$ and to be significantly smaller than the entire latent space. The *affinity based selection function* to obtain the set of states $\mathcal{K}_n$ can be expressed as

$$\mathcal{S}(\mathbf{y}^{(n)}) \;=\; \mathcal{I}\left[\gamma\left[\mathbf{f}(\mathbf{y}^{(n)})\right]\right] \;=\; \mathcal{K}_n. \tag{4.5}$$

To summarize, the main task is to formulate a general data-driven function to identify relevant latent variables and to select the corresponding set of states $\mathcal{K}_n$. This is performed using GP regression in order to compute the truncated posterior in Equation (4.2) on the reduced support $\mathcal{K}_n$. With the combined effort of the above utility functions, we have concisely defined the function $\mathcal{S}(\mathbf{y}^{(n)})$ in Equation (4.5) to perform this selection.

4.3.3 Inference in EM with Selection

In each iteration of EM, the following occurs: prior to the E-step, the selection function $\mathcal{S}(\mathbf{y}^{(n)})$ in (4.5) is computed to select the most relevant states $\mathcal{K}_n$, which are then used to compute the truncated posterior distribution $q_n(\mathbf{s})$ in (4.2). The truncated posterior can be computed using any standard inference method, such as exact inference or e.g. Gibbs sampling from $q(\mathbf{s})$ if inference is still intractable or further computational acceleration is desired. The result of the E-step is then used to update the model parameters with maximum likelihood in the M-step.

4.4 GP-select

In previous work, the selection function $\mathcal{S}(\mathbf{y}^{(n)})$ was a deterministic function derived individually for each model (see e.g. Shelton et al., 2011b, 2012a; Dai and Lücke, 2012a,b; Bornschein et al., 2013; Sheikh et al., 2014; Shelton et al., 2015), specific examples of which will be shown in Section 5.1. We now generalize the selection approach: instead of predefining the form of $\mathcal{S}$ for variable selection, we want to learn it in a black-box and model-free way based on the data. We learn $\mathcal{S}$ using Gaussian process (GP) regression (e.g. Rasmussen and Williams, 2005), which is a flexible nonparametric model and scales cubically[1] with the number of data points N but linearly with the number of latent variables H. We define the affinity function f_h as being drawn from a Gaussian process model: $f_h(\mathbf{y}^{(n)}) \sim \mathrm{GP}\,(0,\, k(\cdot,\cdot))$, where $k(\cdot,\cdot)$ is the covariance kernel, which can be flexibly parameterized to represent the relationship between variables. Again, we use f_h to approximate the marginal posterior probability p_h that $s_h^{(n)} = 1$. A nice property of Gaussian processes is that the kernel matrix K need only be computed once (until the kernel function hyperparameters are updated) to approximate $p_h^{(n)}$ for the entire $H \times N$ set of latent variables $\mathbf{S}$.

Thus, prior to each E-step in each EM iteration, within each calculation of the selection function, we calculate the affinity using a GP to regress the expected values of the latent variables $\langle \mathbf{S} \rangle$ onto the observed data $\mathbf{Y}$. Specifically, we train on p_h from the previous EM iteration (where p_h is equal to $\langle s_h \rangle$), for training data of $\mathcal{D} = \{(\mathbf{y}^{(n)}, \langle \mathbf{s}^{(n)} \rangle_q) | n = 1, \ldots, N\}$, where we recall that $q_n(\mathbf{s}^{(n)})$ is the approximate posterior distribution for $\mathbf{s}^{(n)}$ in Equation (4.2). Note that we do not use a sigmoid link, hence this is clearly not a correct estimate of a probability (it can be negative, or greater than one). From the selection perspective, however, it is not necessary to avoid these pathologies, as we only want an ordering of the variables. A correct GP classification approach with a properly defined likelihood will no longer have a marginal Gaussian distribution, and we would no longer be able to trivially express the posterior means of different functions with the same inputs, without considerable extra computation.

In the first EM iteration, the expectations $\langle \mathbf{s}^{(n)} \rangle_q$ are initialized randomly; in each subsequent EM iteration, the expectations w.r.t. the $\mathcal{K}_n$-truncated posterior $q(\mathbf{s})$ are used. The EM algorithm is run for T iterations and the hyperparameters of the kernel are optimized by maximum likelihood every T^* EM iterations.

For each data point $\mathbf{y}^{(n)}$ and latent variable s_h we compute the predicted mean of the GP by leaving this data point out of the training set and considering all others, which is called

[1] If the scaling with N is still too expensive, an incomplete Cholesky approximation is used, with cost linear in N and quadratic in the rank Q of the approximation (see Section 4.5.3 for details).

leave-one-out (LOO) prediction. It can be shown that this can be implemented efficiently (see Section 5.4.2 in Rasmussen and Williams, 2005), and we use this result to update the predicted affinity as follows:

$$\hat{p}_h^{(n)} \leftarrow \langle s_h^{(n)} \rangle_{q_n} - \frac{[K^{-1}\langle \mathbf{s}_h \rangle_{q_n}]_{nn}}{[K^{-1}]_{nn}}. \tag{4.6}$$

Equation (4.6) can be efficiently implemented for all latent variables $h = 1, \ldots, H$ and all data points $n = 1, \ldots, N$ using matrix operations, thereby requiring only one kernel matrix inversion for the entire data set.

Substituting Equation (4.6) for $\mathbf{f}$ in the affinity based selection function (4.5) ,

$$\mathcal{S}(\mathbf{y}^{(n)}) \;=\; \mathcal{I}\left[\gamma\left[\langle s_h^{(n)} \rangle_{q_n} - \frac{[K^{-1}\langle \mathbf{s}_h \rangle_{q_n}]_{nn}}{[K^{-1}]_{nn}}\right]\right] \;=\; \mathcal{I}\left[\gamma\left[\mathbf{f}(\mathbf{y}^{(n)})\right]\right] \;=\; \mathcal{K}_n$$

we call the entire process *GP-select*. An outline is shown in Algorithm 1.

Algorithm 1 GP-select to accelerate inference in Expectation Maximization

for EM iterations $t = 1, \ldots, T$ **do**
  initialize all latent variables expectations $\langle s_h^{(n)} \rangle_{q_{n,t}}$
  for data point $n = 1, \ldots, N$ **do**
    compute affinity of all latent variables $\hat{\mathbf{p}}_t^{(n)}$ using Eq. (4.6)
    compute subset of relevant states $\mathcal{K}_n$ using Eq. (4.5)
    compute truncated posterior $q_{n,t}(\mathbf{s}^{(n)})$ in E-step: Eq. (4.2)
    update model parameters in M-step, e.g. as in Sec. 4.5.1
    store $\langle s_h^{(n)} \rangle_{q_{n,t}}$ for $\mathbf{p}^{(n)}$ in EM iteration $t + 1$
  end for
  optimize kernel hyperparams every T^* EM iterations
end for

4.5 Experiments

We apply our GP-select inference approach to five different probabilistic generative models. First, we considered three sparse coding models (binary sparse coding, spike-and-slab, nonlinear spike-and-slab), where the relationship between the observed and latent variables is known to be complex and nonlinear. Second, we apply GP-select to a simple Gaussian Mixture Model (GMM), to both provide functional intuition of approach and to explicitly visualize the form of the learned regression function. Finally, we apply our approach to a recent hierarchical model for translation invariant occlusive components analysis (Dai and Lücke, 2012a; Dai et al., 2013; Dai and Lücke, 2014).

4.5.1 Sparse Coding Models

Many types of natural data are composed of potentially many component types, but any data point often only contains a very small number of this potentially large set of components. For the introductory example of toys on the nursery floor, for instance, there are many different toys that can *potentially* be in a given image but there is typically only a relatively small number of toys *actually* appearing in any one image. Another example is a sound played by a piano at a given time t. While the sound can contain waveforms generated by pressing any of the 88 piano keys, there are only relatively few keys (typically much smaller than ten) that actually generated the sound. Sparse Coding algorithms model such data properties by providing a large number of hidden variables (potential data components) but assigning non-zero (or significantly different from zero) values only to a small subset of components (those actually appearing). Sparse coding algorithms are typically used for tasks such as denoising (Elad and Aharon, 2006; Mairal et al., 2009b), inpainting (Mairal et al., 2009b,a; Lázaro-gredilla and Titsias, 2011), classification (LeCun and Cortes, 2010; Lázaro-gredilla and Titsias, 2011; Raina et al., 2007, e.g. MNIST data set[2], the flowers data set[3]), transfer learning (Raina et al., 2007), collaborative filtering (Lázaro-gredilla and Titsias, 2011) and are important models for neurosensory processing (Olshausen and Field, 1997; Zylberberg et al., 2011; Bornschein et al., 2013; Sheikh et al., 2014, and many more). A variety of sparse coding models have been successfully scaled to high-dimensional latent spaces with the use of selection (Henniges et al., 2010; Bornschein et al., 2013; Sheikh et al., 2014) or selection combined with Gibbs sampling (Shelton et al., 2011b, 2012a, 2015) inference approaches. Latent variables were selected in these earlier works using selection functions that were individually defined for each model. In order to demonstrate our method of autonomously learned selection functions, we consider three of these sparse generative models, and perform inference in EM with our GP-select approach instead of a hand-crafted selection function. The models are relevant for different tasks such as classification (e.g., binary sparse coding), source separations and denoising (linear spike-and-slab sparse coding) or sparse encoding and extraction of interpretable image components (nonlinear sparse coding). Note that when it is obvious from context, we drop the notation referring to each data point n in order to make the equations more concise.

[2]The MNIST data set (LeCun and Cortes, 2010): *http://yann.lecun.com/exdb/mnist/*

[3]The flowers data set (Nilsback and Zisserman, 2006): *http://www.robots.ox.ac.uk/∼vgg/data/flowers/*

The models and their parameters are:

A. *Binary sparse coding*:

$$\begin{aligned}
\text{latents:} \quad & \mathbf{s} \sim \mathrm{Bern}(\mathbf{s}|\pi) = \textstyle\prod_{h=1}^{H} \pi^{s_h} \big(1-\pi\big)^{1-s_h} \\
\text{observations:} \quad & \mathbf{y} \sim \mathcal{N}(\mathbf{y}; W\mathbf{s}, \sigma^2 I) \\
\text{parameters:} \quad & W = \Big(\sum_{n=1}^{N} \mathbf{y}^{(n)} \langle \mathbf{s} \rangle_{q_n}^{T}\Big) \Big(\sum_{n=1}^{N} \big\langle \mathbf{s}\,\mathbf{s}^{T} \big\rangle_{q_n}\Big)^{-1} \\
& \sigma^2 = \frac{1}{ND} \sum_{n} \Big\langle \big|\big|\mathbf{y}^{(n)} - W\,\mathbf{s}\big|\big|^2 \Big\rangle_{q_n} \\
& \pi = \frac{1}{N} \sum_{n} |\big\langle \mathbf{s} \big\rangle_{q_n}|, \;\; \text{where } |\mathbf{x}| = \frac{1}{H} \sum_{h} x_h
\end{aligned}$$

where $W \in \mathrm{R}^{D\times H}$ denotes the components / dictionary elements and π parameterizes the sparsity (see e.g. (Henniges et al., 2010)).

B. *Spike-and-slab sparse coding*:

$$\begin{aligned}
\text{latents:} \quad & \mathbf{s} = \mathbf{b} \odot \mathbf{z} \quad \text{where} \quad \mathbf{b} \sim \mathrm{Bern}(\mathbf{b}|\pi) \text{ and } \mathbf{z} \sim \mathcal{N}(\mathbf{z};\, \mu, \Sigma_h) \\
\text{observations:} \quad & \mathbf{y} \sim \mathcal{N}(\mathbf{y}; W\mathbf{s}, \sigma^2 I) \\
\text{parameters:} \quad & W = \frac{\sum_{n=1}^{N} \mathbf{y}^{(n)} \langle \mathbf{s} \odot \mathbf{z} \rangle_n^{\mathrm{T}}}{\sum_{n=1}^{N} \langle (\mathbf{s} \odot \mathbf{z})(\mathbf{s} \odot \mathbf{z})^{\mathrm{T}} \rangle_n} \\
& \pi = \frac{1}{N} \sum_{n=1}^{N} \langle \mathbf{s} \rangle_n \\
& \sigma^2 = \sum_{n=1}^{N} \Big[\big\langle (\mathbf{s} \odot \mathbf{z})(\mathbf{s} \odot \mathbf{z})^{\mathrm{T}} \big\rangle_n - \big\langle \mathbf{s}\,\mathbf{s}^{\mathrm{T}} \big\rangle_n \odot \mu\mu^{\mathrm{T}} \Big] \odot \Big(\sum_{n=1}^{N} \Big[\big\langle \mathbf{s}\,\mathbf{s}^{\mathrm{T}} \big\rangle_n \Big] \Big)^{-1} \\
& \mu_{pr} = \frac{\sum_{n=1}^{N} \langle \mathbf{s} \odot \mathbf{z} \rangle_n}{\sum_{n=1}^{N} \langle \mathbf{s} \rangle_n} \\
& \sigma_{pr}^2 = \frac{1}{N} \sum_{n=1}^{N} \Big[\mathbf{y}^{(n)} (\mathbf{y}^{(n)})^{\mathrm{T}} - W \big[\langle (\mathbf{s} \odot \mathbf{z}) \rangle_n \langle (\mathbf{s} \odot \mathbf{z}) \rangle_n^{\mathrm{T}} \big] W^{\mathrm{T}} \Big]
\end{aligned}$$

where the point-wise multiplication of the two latent vectors, i.e., $(\mathbf{s} \odot \mathbf{z})_h = s_h\, z_h$ generates a 'spike-and-slab' distributed variable ($\mathbf{s} \odot \mathbf{z}$), that has either continuous values or exact zero entries (e.g. (Lázaro-gredilla and Titsias, 2011; Goodfellow et al., 2013; Sheikh et al., 2014)).

C. *Nonlinear Spike-and-slab sparse coding*:

$$
\begin{aligned}
\text{latents:} \quad & \mathbf{s} = \mathbf{b} \odot \mathbf{z} \quad \text{where} \quad \mathbf{b} \sim \text{Bern}(\mathbf{b}|\pi) \\
& \text{and } \mathbf{z} \sim \mathcal{N}(\mathbf{z};\, \mu_{\mathbf{pr}}, \sigma^2) \\
\text{observations:} \quad & \mathbf{y} \sim \mathcal{N}(\mathbf{y}; \max_h\{s_h \mathbf{W}_h\}, \sigma^2 I) \\
\text{parameters:} \quad & \hat{W}_{hd} = \frac{\langle s_h y_d \rangle^*}{\langle s_d^2 \rangle^*} \qquad\qquad \hat{\pi} = \langle \mathbb{I}(s) \rangle \\
& \hat{\sigma}^2 = \left\langle W_{dh} s_h - y_d^{(n)} \right\rangle^* \\
& \hat{\mu}_{\text{pr}} = \langle s_h \rangle^* \qquad\qquad \hat{\sigma}^2_{\text{pr}} = \langle (s_h - \hat{\mu}_{\text{pr}})^2 \rangle^* \\
& \text{Where expectations } \langle\, . \,\rangle^* \text{ mean:} \\
& \langle f(s) \rangle^* = \sum_n \frac{\int_s p(\mathbf{s}|\mathbf{y}^{(n)}, \Theta)\, f(\mathbf{s})\, \mathbb{I}(\text{h is max})\, d\mathbf{s}}{\int_s p(\mathbf{s}|\mathbf{y}^{(n)}, \Theta)\, \mathbb{I}(\text{h is max})\, d\mathbf{s}}
\end{aligned}
$$

where $\mathbb{I}$ is the indicator function denoting the domain to integrate over, namely where h is the maximum. Using $\langle f(s) \rangle^*$ allows for the condensed expression of the update equations shown above. The mean of the Gaussian for each $\mathbf{y}^{(n)}$ is centered at $\max_h\{s_h \mathbf{W}_h\}$, where $\max_h$ is a nonlinearity that considers all H latent components and takes the h yielding the maximum value for $s_h \mathbf{W}_h$ (Lücke and Sahani, 2008; Shelton et al., 2012a; Bornschein et al., 2013; Shelton et al., 2015), instead of centering the data at the linear combination of $\sum_h s_h \mathbf{W}_h = W\mathbf{s}$.

In the above models, inference with the truncated posterior of Equation (4.2) using hand-crafted selection functions $\mathcal{S}_h(\mathbf{y}^{(n)})$ to obtain the subset of states $\mathcal{K}_n$ [of selected relevant variables $\mathbf{s}(\mathbf{y}^{(n)})$], shown in Equation (4.5), has yielded results as good or more robust performance than exact inference (converging less frequently to local optima than exact inference; see earlier references for details). For models **A** and **C**, the hand-constructed function approximating $\mathbf{f}(\mathbf{y}^{(n)})$, for substitution in Equation (4.5), was the cosine similarity between the weights $\mathbf{W}_h$ (e.g. dictionary elements, components, etc.) associated with each latent variable s_h and each data point $\mathbf{y}^{(n)}$: $\mathbf{f}(\mathbf{y}^{(n)}) = (\mathbf{W}_h^{\mathrm{T}} \,/\, ||\mathbf{W}_h||)\, \mathbf{y}^{(n)}$. For model **B**, the constructed affinity function was the data likelihood given a singleton state: $\mathbf{f}(\mathbf{y}^{(n)}) = p(\mathbf{y}^{(n)}|\mathbf{s} = \mathbf{s}_h, \Theta)$, where $\mathbf{s}_h$ represents a singleton state in which only the entry h is non-zero.

The goal of these experiments is to demonstrate the performance of GP-select and the effects/benefits of using different selection functions. To do this, we consider artificial data generated according to each sparse coding model, and thus with known ground-truth parameters. As discussed above, we could also apply the sparse coding models using GP-select to other application domains listed, but that is not the focus of these experiments. We generate $N = 2,000$ data points consisting of $D = 5 \times 5 = 25$ observed

dimensions and $H = 10$ latent components according to each of the models **A-C**: N images of randomly selected overlapping 'bars' with varying intensities for models **B** and **C**, and additive Gaussian noise parameterized by ground-truth $\sigma^2 = 2$ and we choose $H' = 5$, (e.g. following the spike-and-slab prior). On average, each data point contains 2 bars, i.e. ground-truth is $\pi H = 2$, and we choose $H' = 5$. With this choice, we can select sufficiently many latents for virtually all data points.

For each of the models considered, we run 10 repetitions of each of the following set of experiments: (1) selection using the respective hand-crafted selection function, (2) GP-select using a linear covariance kernel, (3) GP-select using a Radial Basis Function (RBF) covariance kernel, and (4) GP-select using a kernel composed by adding the following kernels: RBF, linear, bias and white noise kernels, which we will term the *composition kernel*. As hyperparameters of kernels are learned, the composition kernel (4) can adapt itself to the data and only use the kernel components required. See (Rasmussen and Williams, 2005, Chapter 4, Section 4.2.4) for a discussion on kernel adaptation. Kernel parameters were model-selected via maximum marginal likelihood every 10 EM iterations. For models **A** and **B**, inference was performed exactly using the truncated posterior (4.2), but as exact inference is analytically intractable in model **C**, inference was performed by drawing Gibbs samples from the truncated space (Shelton et al., 2011b, 2012a, 2015). We run all models until convergence.

Results are shown in Figure 4.2. In all experiments, the GP-select approach was able to infer ground-truth parameters as well as the hand-crafted function. For models where the cosine similarity was used (in **A** and **C**), GP regression with a linear kernel quickly learned the ground-truth parameters, and hence fewer iterations of EM were necessary. In other words, even without providing GP-select explicit weights W as required for the hand-crafted function, its affinity function using GP regression (4.6) learned a similar enough function to quickly yield identical results. Furthermore, in the model with a less straight-forward hand-crafted function (in the spike-and-slab model of **B**), only GP regression with an RBF kernel was able to recover ground-truth parameters. In this case (model **B**), GP-select using an RBF kernel recovered the ground-truth 'bars' in 7 out of 10 repetitions, whereas the hand-crafted function recovered the bases in 8 instances. For the remaining models, GP-select converged to the ground-truth parameters with the same average frequency as the hand-crafted functions.

Finally, we have observed empirically that the composition kernel is flexible enough to subsume all other kernels: the variance of the irrelevant kernels dropped to zero in simulations. This suggests the composition kernel is a good choice for general use.

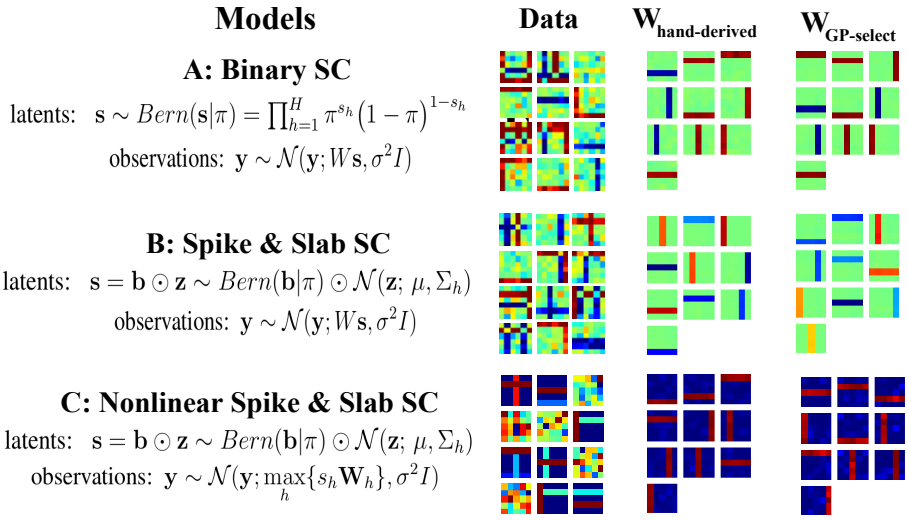


Figure 4.2: *Sparse coding models results comparing GP-select with a successful hand-derived selection function. Results are shown on artificial ground-truth data with $H = 10$ latent variables and $H' = 5$ preselected variables for:* ***A*** *Binary sparse coding,* ***B*** *Spike-and-slab sparse coding, and* ***C*** *Nonlinear spike-and-slab sparse coding. First column: Example data points $\mathbf{y}^{(n)}$ generated by each of the models. Middle column: Converged dictionary elements W learned by the hand-crafted selection functions. Third column: Converged dictionary elements W learned by GP-select with $H' = 5$ using the kernel with best performance (matching that of inference with hand-crafted selection function). In all cases, the model using the GP-select function converged to the ground-truth solution, just as the hand-crafted selection functions did.*

4.5.2 Gaussian Mixture Model

Next, we apply GP-select to a simple example, a Gaussian Mixture Model, where the flexibility of the approach can be easily and intuitively visualized. Furthermore, the GMMs flexibility allows us to explicitly visualize the effect of different selection functions. A demonstration and code for the GMM application is provided in (Dai, 2016).

The model of the data likelihood is

$$p(\mathbf{y}^{(n)}|\mu_{\mathbf{c}}, \sigma_{\mathbf{c}}, \pi) = \sum_{c=1}^{C} \mathcal{N}(\mathbf{y}^{(n)}; \mu_{\mathbf{c}}, \sigma_{\mathbf{c}})\, \pi_c, \tag{4.7}$$

where C is the number of mixture components and π_c is the prior probability of a given cluster; the task is to assign each data point to its latent cluster.

The training data used for GP regression was $\mathcal{D} = \{(\mathbf{y}^{(n)}, \langle s_h^{(n)} \rangle_{q_n}) | n = 1, \ldots, N\}$, where the targets were the expected cluster responsibilities (posterior probability distribution for each cluster) for all data points, $\langle s_h \rangle_q$, and we use one-hot encoding for cluster identity. With this, we apply our GP-select approach to this model, computing the selection

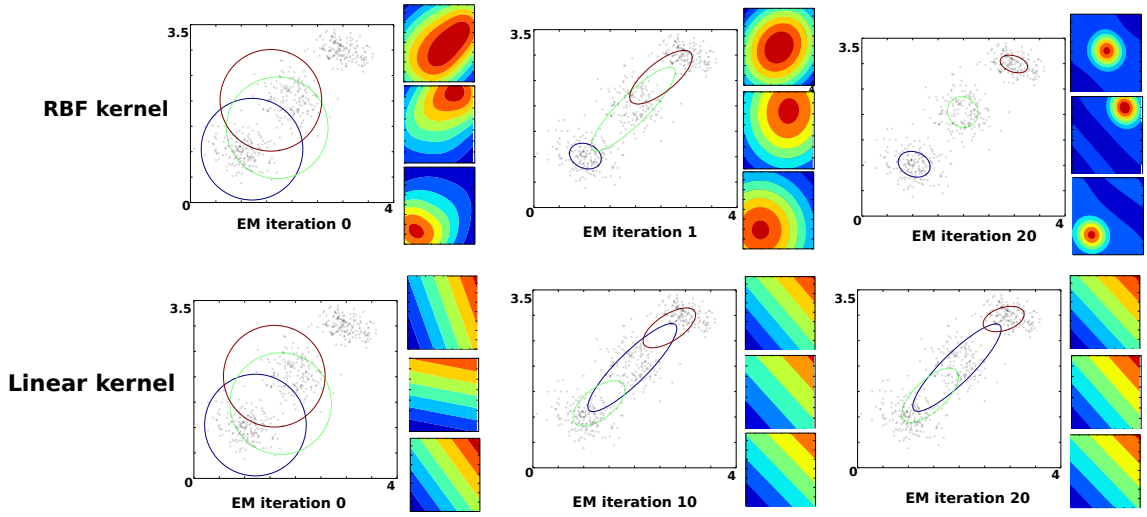


Figure 4.3: *Gaussian Mixture Model results using GP-select (selection of $C' = 2$ in a $C = 3$ cluster scenario) for inference on 2-dimensional observed data (for illustrative visualization). Progress of the inference is shown using (row one) an RBF covariance kernel in the regression, and (row two) a linear covariance kernel. For each iteration shown, we see (1) the observed data and their inferred cluster assignments and (2) the C corresponding GP regression functions learned/used for GP-select in that iteration. Different iterations are pictured due to different convergence rates. As shown, inference with GP-select using a linear kernel is unable to assign the data points to the appropriate clusters, whereas GP-select with an RBF kernel succeeds.*

function according to Equation (4.5) with affinity f defined by GP regression (4.6) and following the approximate EM approach as in the previous experiments. In these experiments we consider two scenarios for EM learning of the data likelihood in Equation (4.7): GP-select with an RBF covariance kernel and a linear covariance kernel. We do not include the composition kernel suggested (based on experiments) in Section 4.1, as the goal of the current experiments is to show the effects of using the 'wrong' kernel. These effects would further support the use of the flexible composition kernel in general, as it can subsume both kernels considered in the current experiments (RBF and linear).

To easily visualize the output, we generate 2-dimensional observed data ($\mathbf{y}^{(n)} \in \mathrm{R}^{D=2}$) from $C = 3$ clusters – first with randomly assigned cluster means, and second such that the means of the clusters lie roughly on a line. In the GP-select experiments, we select $C' = 2$ clusters from the full set, and run 40 EM iterations for both kernel choices (linear and RBF). Note that for mixture models, the notation of C' selected clusters of the C set is analogous to the H' selected latent variables from the H full set, as described in the non-mixture model setting, and the GP-select algorithm proceeds unchanged. We randomly initialize the variance of the clusters $\sigma_{\mathbf{c}}$ and initialize the cluster means $\mu_{\mathbf{c}}$ at randomly selected data points. Results are shown in Figure 4.3.

With cluster parameters initialized randomly on these data, the linear GP regression prediction cannot correctly assign the data to their clusters (as seen in Figure 4.3**B**), but the nonlinear approach successfully and easily finds the ground-truth clusters (Figure 4.3**A**). Furthermore, even when both approaches were initialized in the optimal solution, the cluster assignments from GP-select with a linear kernel quickly wandered away from the optimal solution and were identical to random initialization, converging to the same result shown in iteration 20 of Figure 4.3**B**). The RBF kernel cluster assignments remained at the optimal solution even with number of selected clusters set to $C' = 1$.

These experiments demonstrate that the selection function needs to be flexible even for very simple models, and that nonlinear selection functions are an essential tool even in such apparently straightforward cases.

4.5.3 Translation Invariant Occlusive Models

Now that we have verified that GP-select can be applied to various generative graphical models and converge to ground-truth parameters, we consider a more challenging model that addresses a problem in computer vision: *translations of objects in a scene*.

Model. Translation invariant models address the problem that, e.g. visual objects can appear in any location of an image. Probabilistic models for translation invariance are particularly appealing as they allow to separately infer object positions and object type, making them very interpretable and powerful tools for image processing.

Translation invariant models are particularly difficult to optimize, however, because they must consider a massive latent variable space: evaluating multiple objects and locations in a scene leads a *latent space complexity of the number of locations exponentiated by the number of objects*. Inference in such a massive latent space heavily relies on the idea of variable selection to reduce the number of candidate objects and locations. In particular, hand-engineered selection functions that consider *translational invariance* have been successfully applied to this type of model (Dai and Lücke, 2012b, 2014; Dai et al., 2013). The model we consider in these experiments is the *Invariant Exclusive Component Analysis (InvECA) model* (Dai and Lücke, 2012b; Dai et al., 2013). In contrast to linear models, the InvECA model requires two sets of parameters for the encoding of image components: component masks and component features. Component masks describe where an image component is located and component features describe what a component encodes. High values of mask parameters encode the pixels most associated with a component h but the encoding has to be understood relative to a global component position. Until now, the selection function used to reduce latent space complexity in this model has been constructed as follows. First, the candidate locations of all the objects in the model

are predicted. Then, a subset of candidate objects that might appear in the image are selected according to those predicted locations. Next, the subset of states $\mathcal{K}_n$ is constructed according to the combinations of the possible locations and numbers of candidate objects. The posterior distribution is then computed following Equation (4.2).

This selection system is very costly: the selection function has parameters which need to be hand-tuned, e.g. the number of representative features, and it needs to scan through the entire image, considering all possible locations, which becomes computationally demanding for large-scale experiments. To maximally exploit the capabilities of GP-select's flexibility, we directly use the GP regression model to *predict the possible locations* of a component *without introducing any knowledge of translation invariance* into the selection function. In this work, a GP regression model is fitted from the input image to marginal posterior probabilities of individual objects appearing at all possible locations. Therefore, the input to the GP is the image to be inferred and the output is a score for each possible location of each object in the model. For example, when learning 10 objects in a $D = 30 \times 30$ pixel image patch, the output dimensionality of GP-select is $9,000$. This task is computationally feasible, since GP models scale linearly with output dimensionality. The inference of components' locations with GP-select is significantly faster than the selection function in the original work, as it avoids explicitly scanning through the image.

Although GP-select has some additional computations, e.g. parameters to tune, there are many options to reduce these computational costs. First, we can approximate the full $N \times N$ Gram matrix by an incomplete Cholesky approximation (Fine and Scheinberg, 2002) resulting in a cost of $O(N \times Q)$, where $Q \ll N$ is the rank of the Cholesky approximation. Second, we may reduce the update frequency of the kernel hyperparameters to be computed only every T^* EM iterations, where a $T^* > 1$ represents a corresponding computation reduction. The combination of the Cholesky approximation plus infrequent updates will have the following benefits: a factor of five speedup for infrequent updates, and a factor of $(N - Q)^2$ speedup from incomplete Cholesky, where Q is the rank of the Cholesky approximation and N is the number of original data points.

COIL data set. In all experiments, we consider an image data set used in previous work: data were generated using 16 objects extracted from the COIL-100 image data set (Nene et al., 1996), downscaled to $D = 10 \times 10$ pixels, and segmented out from the black background. A given image was generated by randomly selecting a subset of the 16 objects, where each object has a probability of 0.2 of appearing. These objects were then placed at random positions on a 30×30 black image. When the objects overlap, they occlude each other with a different random depth order for each image. In total, $N = 2,000$ images were generated for the data set (examples shown in Figure 4.4).

Experiments. The task of the InvECA model is to discover the visual components (i.e.

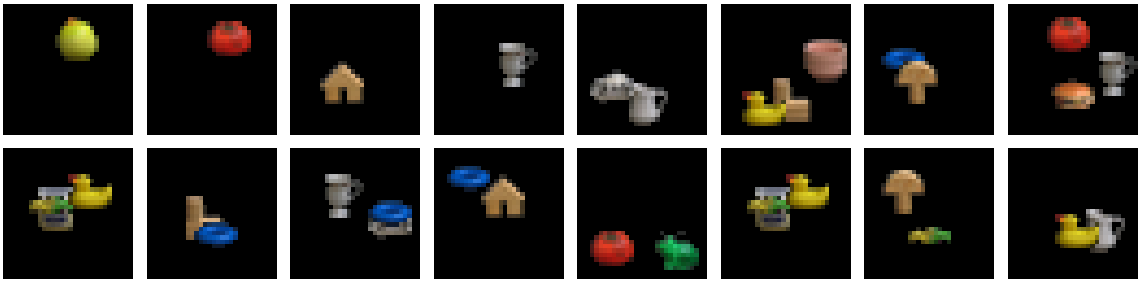


Figure 4.4: *COIL-100 data set (Nene et al., 1996): A handful of data points generated using objects from the COIL-100 data set and used in experiments with the Invariant Exclusive Component Analysis (InvECA) model, showing the occluding objects to be learned.*

the images of 16 objects) from the image set without any label information. We compare the visual components learned by using *four different selection functions in the InvECA model*: the hand-crafted selection function used in the original work by Dai and Lücke (2012b), GP-select updated every iteration, GP-select updated every $T* = 5$ iterations, and GP-select with incomplete Cholesky decomposition updated every iteration, or $T^* = 1$ (in this manner we isolate the improvements due to Cholesky from those due to infrequent updates). In these experiments, the parameters of GP-select are optimized at the end of each $T*$ EM iteration(s), using a maximum of 20 gradient updates. The number of objects to be learned is $H = 16$ and the algorithm preselects $H' = 5$ objects for each data point. The kernel used was the composition kernel, as suggested in Section 4.1, although after fitting the hyperparameters only the RBF kernel remained with large variance (i.e. a linear kernel alone would not have produced good variable selection, thus the flexible composition kernel was further shown to be a good choice).

Results.

All four versions of the InvECA model with their respective selection functions successfully recovered all 16 objects in our modified COIL image set. The learned object representations with GP-select are shown in Figure 4.5. Four additional components developed into representations, however these all had very low mask values, allowing them to easily be distinguished from other true components.

Next, we compare the accuracy of the four selection functions. For this, we collected the object locations (pixels) indicated by each selection function after all EM iterations, applied the selection functions (for the GP selection functions, this was using the final function learned after all EM iterations) to the entire image data set again, then compared these results with the ground-truth location of all of the objects in the data set. The accuracy of the predicted locations was then computed by comparing the distance of all ground-truth object location to the location of the top candidate locations from each selection function. See Figure 4.6 for a histogram of these distances and the corresponding

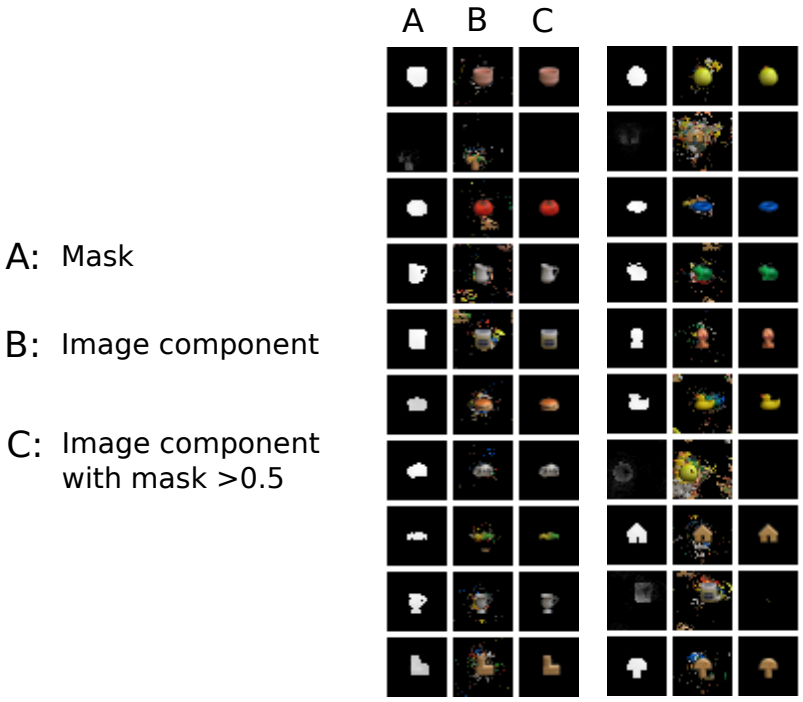


Figure 4.5: *Image components and their masks learned by GP-select with the Translation Invariant model. GP-select learned all objects in the data set. The figure shows:* ***A*** *the mask of each component,* ***B*** *the learned image components, and* ***C*** *only the area of the learned components that had a mask* > 0.5*. For the second three-column block of images, the same labels of the first three-column block hold.*

accuracy for all selection functions. Note that the percentages in the histogram are plotted in log scale. Also, as a baseline verification, we computed and compared the pseudo log likelihood (Dai et al., 2013) of the original selection function to the three GP-select based ones. The pseudo log likelihood for all selection functions is shown in Figure 4.7. Figures 4.6-4.7 show that all four selection functions can very accurately predict the locations of all the objects in the data set – the GP-select selection functions yields no loss in inference performance in comparison to the original hand-engineered selection function. Even those using speed-considerate approximations (incomplete Cholesky decomposition of the kernel matrix (GP IChol) and updating kernel hyperparameters only every 5 EM iterations (GP every5)) have indistinguishable prediction accuracy on the task.

An analysis of the benefits indicate that, as GP-select avoids explicitly scanning through the image, the time to infer the location of an object is significantly reduced compared to the hand-crafted function. GP-select requires 22.1 seconds on a single CPU core to infer the locations of objects across the whole image set, while the hand-crafted function requires 1830.9 seconds. In the original work, the selection function was implemented with GPU acceleration and parallelization. Although we must compute the kernel hy-

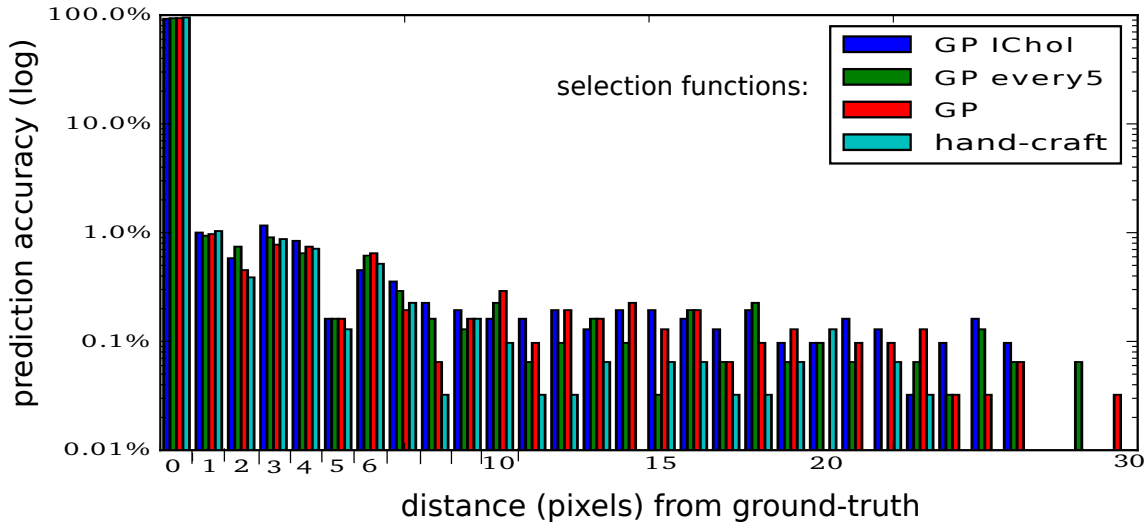


Figure 4.6: *Prediction accuracy of the four selection functions in the InvECA model. Functions depicted in the figures: GP-select with no modifications (GP, red), the incomplete Cholesky decomposition (GP IChol, blue), with updated kernel hyperparameters every 5 EM iterations (GP every5, green), and with hand-crafted selection (hand-craft, cyan). Shown: the log-scale histogram of the prediction accuracy for the four selection functions, measured by the distance each function's predicted object location was to the ground-truth object location. All bars of the selection functions show very similar accuracy for the various distances.*

perparameters for GP-select, it is important to note that the hyperparameters need not be fit perfectly each iteration – for the purposes of our approach, a decent approximation suffices for excellent variable selection. In this experiment, updating the parameters of GP-select with 10 gradient steps took about 390 seconds for the full-rank kernel matrix. When we compute the incomplete Cholesky decomposition while inverting the covariance matrix, compute time was reduced to 194 seconds (corresponding to the $(N-Q)^2$ speedup, where Q is the rank of the Cholesky approximation), with minimal loss in accuracy. Furthermore, when updating the GP-select hyperparameters only every 5 iterations, average compute time was reduced by another one fifth, again without loss in accuracy.

4.6 Discussion

We have proposed a means of achieving fast EM inference in Bayesian generative models, by learning an approximate selection function to determine relevant latent variables for each observed variable. The process of learning the relevance functions is interleaved with the EM steps, and these functions are used in obtaining an approximate posterior distribution in the subsequent EM iteration. The functions themselves are learned via Gaussian process regression, and do not require domain-specific engineering, unlike previous selec-

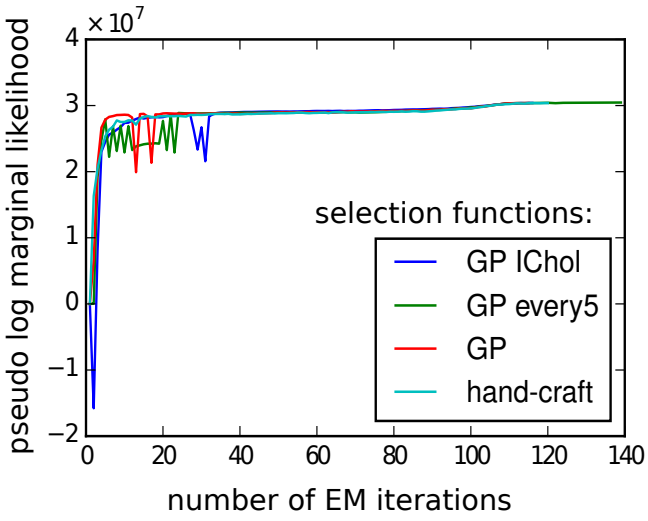


Figure 4.7: *Baseline comparison of the four selection functions in the InvECA model. Functions depicted in the figures are identical to those in Figure 4.6. Shown: the convergence of the pseudo log marginal likelihood [of the model parameters learned at each EM iteration] for the four selection functions over all EM iterations. After about 40 EM iterations, all selection function versions of the algorithm converge to the same likelihood solution. Simultaneously, the GP-select approaches exhibit no loss of accuracy compared to the hand-crafted function, and 'GP IChol' represents a factor of* 100 *speedup vs. 'GP', and 'GP every5' represents a factor of* 5 *speedup.*

tion functions. In experiments on mixtures and sparse coding models with interpretable output, the learned selection functions behaved in accordance with our expectations for the posterior distribution over the latents.

The significant benefit we show empirically is that by learning the selection function in a general and flexible nonparametric way, we can avoid using potentially expensive hand-engineered selection functions. Cost reduction is both in terms of required expertise in the problem domain, and computation time in identifying the relevant latent variables. Inference using our approach required 22.1 seconds on a single CPU core, versus 1830.9 seconds with the original hand-crafted function for the complex hierarchical model of (Dai et al., 2013).

Very large data sets would however pose a challenge to the efficiency of our approach. Namely, in computing the GP, the matrix inversion in the LOO Equation (4.6) would be resource-heavy. We were able to improve the efficiency of this step by using the incomplete Cholesky approximation, but further measures would be necessary for very large scale applications. Possibilities to accomplish this are discussed in the final Section.

A possible limitation to our approach is that GP regression assumes a smooth relation between the input and target variables, but if the real relation has discontinuities then a GP

will be a poorer approximation than a selection function that is explicitly discontinuous. Furthermore, GPs have parameters that need to be adjusted, e.g. after some amount of EM iterations, whereas a prebaked hand-crafted selection function might not. Although the selection function explicitly crafted for the InvECA model indeed has parameters that need to be tuned, this may not always be the case. Thus, in such scenarios a GP based selection function could be more costly.

Chapter 5

Conclusion and Discussion

In this work, we have proposed a means of achieving fast EM inference in Bayesian generative models. We do this by learning an approximate selection function to determine relevant latent variables for each observed variable prior to each E-step and then compute the posterior distribution in the E-step using the set of just these selected latent variables. The process of learning the relevance functions is interleaved with the EM steps, and these functions are used in obtaining an approximate posterior distribution in the subsequent EM iteration.

In Chapter 2, we introduced *Select and Sample*, the basic approximate inference approach which was built upon and applied in the subsequent Chapters. This approach combines latent variable preselection with Markov Chain Monte Carlo (MCMC) sampling methods for the acceleration of inference and learning with EM, in order to capture the strengths of each approach in representing complex posterior distributions and simultaneously reduce computational costs. There, the selection function used to identify the relevant latent variables was hand-derived for each individual model and required expertise with the problem domain. Our results with a simple sparse coding model using our approach show significant reduction in computational resources. In Chapter 3, we applied the Select and Sample approach to a more complex and novel sparse coding model designed to model low-level image components (such as edge-like structures and their occlusions). The model used a complex prior distribution (spike-and-slab) – to model the presence/absence of e.g. an edge as well as its pixel intensity – and had a nonlinearity in the data likelihood (the nonlinear max combination rule) to target occlusions, i.e. dictionary elements correspond to image components that can occlude each other. We called this model *SSMCA*. The nonlinearity in the data likelihood lead to a highly multi-modal complex posterior distribution, thus in order to adequately sample this distribution we developed an exact Gibbs sampler based on the exact form of the posterior distribution. Results showed that SSMCA can

model the generating process of images with occlusions, including extracting individual edge-like structures that occlude each other, and produces results that are neurally consistent. Finally, in Chapter 4 we introduced a generalization of the Select and Sample method used in the previous chapters. There, the selection function used to preselect relevant latent variables was hand-engineered for each individual model, here we propose a model-independent non-parametric black-box way to define a suitable selection function efficiently. Namely, we learned the selection function entirely from the observed data and current EM state using Gaussian process regression. We have named this approach *GP-select*. Empirical experiments showed equivalent performance between our inference algorithm (using GP-select to preselect variable) and algorithms of previous work (using a complex hand-engineered selection function for preselection). At the same time, GP-select is straightforward to implement and had a far lower computational cost.

Ideally, the selection function would be further generalized such that it would not require a hand-chosen parameter to specify how many latent variables it can preselect. Instead, it would be advantageous to have the number of preselected variables H' be adaptive and set based on the actual sparsity in the data. An adaptive H' could improve the current work: computational resources could be spared should H' have been set 'too high' and more variables be preselected than the data necessitates, or likewise adaptation could improve inference accuracy should H' have been set too low and too few variables be preselected.

Considering future directions, a major area where further performance gains might be expected is in improving computational performance, since we expect the greatest advantages of GP-select to occur for complex models at large scale. For instance, kernel ridge regression may be parallelized (Zhang et al., 2014), or the problem may be solved in the primal via random Fourier features (Le et al., 2013). Furthermore, there are many recent developments regarding the scaling up of GP inference to large-scale problems, e.g. sparse GP approximation (Lawrence et al., 2002), stochastic variational inference (Hensman et al., 2013, 2012), using parallelization techniques and GPU acceleration (Dai et al., 2014), or in combination with stochastic gradient descent (Bottou and Bousquet, 2008). For instance, for very large data sets where the main model is typically trained with mini-batch learning, stochastic variational inference can be used for GP fitting as in (Hensman et al., 2013) and the kernel parameters can be efficiently updated each (or only every T^* few) iteration with respect to a mini-batch. Another way to reduce computational costs would be to replace the selection step in Select and Sample with the black-box approach of GP-select, namely to draw samples from the posterior distribution truncated to the reduced variable set selected by GP regression.

Bibliography

Beck, J. M., Ma, W. J., Kiani, R., Churchland, T. H. A. K., Roitman, J., Shadlen, M. N., Latham, P. E., Pouget, A., 2008. Probabilistic population codes for bayesian decision making. Neuron 60, 1142–1152.

Berkes, P., Orban, G., Lengyel, M., Fiser, J., Jan. 2011a. Spontaneous Cortical Activity Reveals Hallmarks of an Optimal Internal Model of the Environment. Science 331 (6013), 83–87.

Berkes, P., Turner, R., Fiser, J., 2011b. The army of one (sample): the characteristics of sampling-based probabilistic neural representations. In: Frontiers in Neuroscience. Computational and Systems Neuroscience.

Bornschein, J., Dai, Z., Lücke, J., 2010. Approximate EM learning on large computer clusters. In: NIPS Workshop: Learning on Cores, Clusters and Clouds.

Bornschein, J., Henniges, M., Lücke, J., 06 2013. Are V1 simple cells optimized for visual occlusions? A comparative study. PLoS Computational Biology 9 (6), 1–16.

Bottou, L., Bousquet, O., 2008. The tradeoffs of large scale learning. In: Platt, J. C., Koller, D., Singer, Y., Roweis, S. T. (Eds.), Advances in Neural Information Processing Systems 20. pp. 161–168.

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

Currin, C., Mitchell, T., Morris, M., Ylvisaker, D., 1991. Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. J. American Statistical Association 86 (416), 953–963.

Dai, Z., 2016. GP-select Demo on Gaussian mixture models. https://github.com/fatflake/GP-select-Code/blob/master/GMM_demo.ipynb.

Dai, Z., Damianou, A., Hensman, J., Lawrence, N., 2014. Gaussian process models with parallelization and gpu acceleration. In: NIPS Workshop on Modern non-parametrics: automating the learning pipeline.

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

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

Dai, Z., Lücke, J., 2012b. Unsupervised learning of translation invariant occlusive components. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 2400–2407.

Dai, Z., Lücke, J., 2014. Autonomous document cleaning – a generative approach to reconstruct strongly corrupted scanned texts. IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (10), 1950–1962.

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

Dayan, P., Hinton, G. E., Neal, R. M., Zemel, R. S., 1995. The helmholtz machine. Neural Computation 7 (5), 889–904.

Dayan, P., Zemel, R. S., 1995. Competition and multiple cause models. Neural Computation 7 (3), 565–579.

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

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

Elad, M., Aharon, M., Dec. 2006. Image denoising via sparse and redundant representations over learned dictionaries. Trans. Img. Proc. 15 (12), 3736–3745.

Eldar, Y., Kutyniok, G., 2012. Compressed Sensing: Theory and Applications. Cambridge University Press.

Ernst, M. D., Banks, M. S., 2002. Humans integrate visual and haptic information in a statistically optimal fashion. Nature 415 (6870).

Exarchakis, G., Henniges, M., Eggert, J., Lücke, J., 2012. Ternary sparse coding. In: LVA/ICA. Lecture Notes in Computer Science. Springer, pp. 204–212.

Fine, S., Scheinberg, K., 2002. Efficient SVM training using low-rank kernel representations. Journal of Machine Learning Research 2, 243–264.

Fiser, J., Berkes, P., Orban, G., Lengyel, M., 2010. Statistically optimal perception and learning: from behavior to neural representations. Trends in Cognitive Science 14, 119–130.

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

Goodfellow, I., Courville, A., Bengio, Y., 2011. Spike-and-slab sparse coding for unsupervised feature discovery. In: NIPS Workshop on Challenges in Learning Hierarchical Models.

Goodfellow, I., Courville, A., Bengio, Y., 2012. Large-scale feature learning with spike-and-slab sparse coding. In: International Conference on Machine Learning 29. pp. 1439–1446.

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

Gutmann, M. U., Corander, J., 2015. Bayesian Optimization for Likelihood-Free Inference of Simulator-Based Statistical Models. Tech. rep., University of Helsinki, *http://arxiv.org/abs/1501.03291*.

Haft, M., Hofman, R., Tresp, V., 2004. Generative binary codes. Pattern Anal Appl 6, 269–84.

Han, S., Mangasarian, O., 1979. Exact penalty functions in nonlinear programming. Vol. 17 (1). pp. 251–269.

Henniges, M., Puertas, G., Bornschein, J., Eggert, J., Lücke, J., 2010. Binary Sparse Coding. In: Proceedings LVA/ICA. LNCS 6365. Springer, pp. 450–57.

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

Hensman, J., Fusi, N., Lawrence, N., 2013. Gaussian processes for big data. In: Nicholson, A., Smyth, P. (Eds.), Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence 29. pp. 282–290.

Hensman, J., Rattray, M., Lawrence, N. D., 2012. Fast Variational Inference in the Conjugate Exponential Family. In: Bartlett, P. L., Pereira, F. C. N., Burges, C. J. C., Bottou, L., Weinberger, K. Q. (Eds.), Advances in Neural Information Processing Systems 25. pp. 2897–2905.

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

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

Hoyer, P. O., 2003. Modeling receptive fields with non-negative sparse coding. Neurocomputing 54, 547–52.

Hubel, D. H., Wiesel, T. N., 1959. Receptive fields of single neurones in the cat's striate cortex. The Journal of Physiology 148 (3), 574–591.

Jernite, Y., Halpern, Y., Sontag, D., 2013a. Discovering hidden variables in noisy-or networks using quartet tests. In: Burges, C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. (Eds.), Advances in Neural Information Processing Systems 26. pp. 2355–2363.

Jernite, Y., Halpern, Y., Sontag, D., 2013b. Discovering hidden variables in noisy-or networks using quartet tests. In: Burges, C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. (Eds.), Advances in Neural Information Processing Systems 26. pp. 2355–2363.

Kingma, D. P., Welling, M., 2014. Efficient gradient-based inference through transformations between bayes nets and neural nets. In: International Conference on Machine Learning 31. pp. 1782–1790.

Kording, K. P., Wolpert, D. M., 2004. Bayesian integration in sensorimotor learning. Nature 427, 244–247.

Körner, E., Gewaltig, M. O., Körner, U., Richter, A., Rodemann, T., 1999. A model of computation in neocortical architecture. Neural Networks 12, 989–1005.

Laplace, P., 1774. Memoir on the probability of causes of events. Mémoires de Mathématique et de Physique, Tome Sixième. English translation by Stigler, S. M., 1986. Statistical Science 1 (19), 364–378.

Lawrence, N., Seeger, M., Herbrich, R., 2002. Fast sparse gaussian process methods: The informative vector machine. In: Becker, S., Thrun, S., Obermayer, K. (Eds.), Advances in Neural Information Processing Systems 15. pp. 609–616.

Lázaro-gredilla, M., Titsias, M. K., 2011. Spike and slab variational inference for multi-task and multiple kernel learning. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. (Eds.), Advances in Neural Information Processing Systems 24. pp. 2339–2347.

Le, Q., Sarlos, T., Smola, A. J., 2013. Fastfood — computing hilbert space expansions in loglinear time. In: International Conference on Machine Learning 30. pp. 244–252.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D., 1989. Backpropagation applied to handwritten zip code recognition. Neural Computation 1 (4), 541–551.

LeCun, Y., Cortes, C., 2010. MNIST handwritten digit database.
URL `http://yann.lecun.com/exdb/mnist/`

Lee, H., Battle, A., Raina, R., Ng, A., 2007. Efficient sparse coding algorithms. In: Schölkopf, B., Platt, J. C., Hoffman, T. (Eds.), Advances in Neural Information Processing Systems 19. pp. 801–808.

Lee, T. S., Mumford, D., 2003a. Hierarchical Bayesian inference in the visual cortex. Journal of the Optical Society of America A 20 (7), 1434–1448.

Lee, T. S., Mumford, D., July 2003b. Hierarchical Bayesian inference in the visual cortex. J Opt Soc Am A Opt Image Sci Vis 20 (7), 1434–1448.

Lücke, J., 2009. Receptive field self-organization in a model of the fine-structure in V1 cortical columns. Neural Computation 21 (10), 2805–45.

Lücke, J., Eggert, J., 2010. Expectation truncation and the benefits of preselection in training generative models. Journal of Machine Learning Research 11, 2855–2900.

Lücke, J., Sahani, M., 2008. Maximal causes for non-linear component extraction. Journal of Machine Learning Research 9, 1227–67.

Ma, W. J., Beck, J. M., Latham, P. E., Pouget, A., 2006. Bayesian inference with probabilistic population codes. Nature Neuroscience 9, 1432–1438.

Mairal, J., Bach, F., Ponce, J., Sapiro, G., 2009a. Online dictionary learning for sparse coding. Vol. 25. pp. 689–696.

Mairal, J., Bach, F., Ponce, J., Sapiro, G., 2010. Online learning for matrix factorization and sparse coding. Journal of Machine Learning Research 11, 19–60.

Mairal, J., Bach, F., Ponce, J., Sapiro, G., Zisserman, A., 2009b. Non-local sparse models for image restoration. International Conference on Computer Vision 25, 2272–2279.

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

Mangasarian, O., 1969. Linear and nonlinear separation of patterns by linear programming. Operations research 13 (3), 444–452.

Meeds, E., Welling, M., 2014. GPS-ABC: gaussian process surrogate approximate bayesian computation. In: Conference on Uncertainty in Artificial Intelligence 13. pp. 593–602.

Mnih, A., Gregor, K., 2014. Neural variational inference and learning in belief networks. In: International Conference on Machine Learning 31. pp. 1791–1799.

Mohamed, S., Heller, K., Ghahramani, Z., 2012. Evaluating Bayesian and L1 approaches for sparse unsupervised learning. In: International Conference on Machine Learning 29. pp. 751–758.

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

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

Neal, R. M., Jul. 1992. Connectionist learning of belief networks. Artificial Intelligence 56 (1), 71–113.

Neal, R. M., 1993. Probabilistic inference using markov chain monte carlo methods. Tech. rep., Dept. of Computer Science, University of Toronto.

Nene, S. A., Nayar, S. K., Murase, H., 1996. Columbia object image library (coil-100). Tech. rep., CUCS-006-96.

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

Nilsback, M.-E., Zisserman, A., 2006. A visual vocabulary for flower classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Vol. 2. pp. 1447–1454.

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

Olshausen, B., Millman, K., 2000. Learning sparse codes with a mixture-of-Gaussians prior. In: Solla, S. A., Leen, T. K., Müller, K. (Eds.), Advances in Neural Information Processing Systems 12. pp. 841–847.

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

Opper, M., Winther, O., 2005. Expectation consistent approximate inference. Journal of Machine Learning Research 6, 2177–2204.

Puertas, G., Bornschein, J., Lücke, J., 2010. The maximal causes of natural scenes are edge filters. In: Lafferty, J., Williams, C. K. I., Zemel, R., Shawe-Taylor, J., Culotta, A. (Eds.), Advances in Neural Information Processing Systems 23. pp. 1939–1947.

Raina, R., Battle, A., Lee, H., Packer, B., Ng, A. Y., 2007. Self-taught learning: Transfer learning from unlabeled data. In: International Conference on Machine Learning 24. pp. 759–766.

Rao, R. P. N., Olshausen, B. A., Lewicki, M. S., 2002. Probabilistic Models of the Brain: Perception and Neural Function. MIT Press.

Rasmussen, C. E., Williams, C. K. I., 2005. Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press.

Riesenhuber, M., Poggio, T., 1999. Hierarchical models of object recognition in cortex. Nature Neuroscience 211 (11), 1019 – 1025.

Riesenhuber, M., Poggio, T., 07 2002. How visual cortex recognizes objects: The tale of the standard model (short title: Computational object vision). The Visual Neurosciences 2.

Ringach, D., 2002. Spatial structure and symmetry of simple-cell receptive fields in macaque primary visual cortex. Journal of Neurophysiology 88, 455–63.

Rosenblatt, F., 1958. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review 65 (6), 65–386.

Roweis, S. T., 2003. Factorial models and refiltering for speech separation and denoising. In: Eurospeech 8. pp. 1009–1012.

Rupp, M., Tkatchenko, A., Müller, K.-R., von Lilienfeld, O. A., Jan 2012. Fast and accurate modeling of molecular atomization energies with machine learning. Physical Review Letters 108, 058301.

Sacks, J., Welch, W. J., Mitchell, T. J., Wynn, H. P., 11 1989. Design and analysis of computer experiments. Statistical Science 4 (4), 433–435.

Saund, E., 1995. A multiple cause mixture model for unsupervised learning. Neural Computation 7 (1), 51–71.

Schwaighofer, A., Tresp, V., Yu, K., 2004. Learning gaussian process kernels via hierarchical bayes. In: Saul, L., Weiss, Y., Bottou, L. (Eds.), Advances in Neural Information Processing Systems 17. pp. 1209–1216.

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

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

Shelton, J., Bornschein, J., Sheikh, A.-S., Berkes, P., Lücke, J., 2011a. Select and sample - a model of efficient neural inference and learning. Women in Machine Learning Workshop (WiML 2011) in conjunction with NIPS, Malaga, Spain.

Shelton, J., Bornschein, J., Sheikh, A.-S., Berkes, P., Lücke, J., 2011b. Select and Sample - A Model of Efficient Neural Inference and Learning. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. (Eds.), Advances in Neural Information Processing Systems 24. pp. 2618–2626.

Shelton, J., Sheikh, A.-S., Sterne, P., Bornschein, J., Lücke, J., 2013. Nonlinear spike-and-slab sparse coding for interpretable image encoding. NIPS Workshop on High-dimensional Statistical Inference in the Brain.

Shelton, J., Sterne, P., Bornschein, J., Sheikh, A.-S., Lücke, J., 2012a. Why MCA? Nonlinear sparse coding with spike-and-slab prior for neurally plausible image encoding. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (Eds.), Advances in Neural Information Processing Systems 25. pp. 2285–2293.

Shelton, J., Sterne, P., Bornschein, J., Sheikh, A.-S., Lücke, J., 2012b. Why MCA? nonlinear spike-and-slab sparse coding with spike-and-slab prior for neurally plausible image encoding. Women in Machine Learning Workshop (WiML 2012) in conjunction with NIPS, Lake Tahoe, Nevada.

Shelton, J. A., Gasthaus, J., Dai, Z., Lücke, J., Gretton, A., 2014. Gp-select: Accelerating EM using adaptive subspace preselection. Women in Machine Learning Workshop (WiML 2014) in conjunction with NIPS, Montreal, Quebec.

Shelton, J. A., Gasthaus, J., Dai, Z., Lücke, J., Gretton, A., 2017. Gp-select: Accelerating EM using adaptive subspace preselection. Neural Computation 29 (8), 2177–2202.

Shelton, J. A., Sheikh, A.-S., Bornschein, J., Sterne, P., Lücke, J., 2015. Nonlinear spike-and-slab sparse coding for interpretable image encoding. PLoS ONE 10 (5), 1–25.

Singliar, T., Hauskrecht, M., 2006. Noisy-or component analysis and its application to link analysis. Journal of Machine Learning Research 7, 2189–2213.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Narayanan Sundaram, M., Ali, M., Patwary, P., Adams, R., 2015. Scalable bayesian optimization using deep neural networks. Tech. rep., Harvard University, *http://arxiv.org/abs/1502.05700*.

Tan, X., Li, J., Stoica, P., 2010. Efficient sparse Bayesian learning via Gibbs sampling. In: IEEE International Conference on Acoustics Speech and Signal Processing. pp. 3634–3637.

Tibshirani, R., 1996. Regression shrinkage and selection via the Lasso. Journal of the Royal Statistical Society. Series B 58 (1), 267–288.

Trommershäuser, J., Maloney, L. T., Landy, M. S., 2008. Decision making, movement planning and statistical decision theory. Trends in Cognitive Science 12, 291–297.

Turner, R., Berkes, P., Fiser, J., 2011. Learning complex tasks with probabilistic population codes. In: Frontiers in Neuroscience. Computational and Systems Neuroscience.

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

Valpola, H., Oja, E., Ilin, A., Honkela, A., Karhunen, J., 1999. Nonlinear blind source separation by variational bayesian learning. Digital Signal Processing 17 (5), 914–934.

van Hateren, J. H., van der Schaaf, A., 1998. Independent component filters of natural images compared with simple cells in primary visual cortex. Proceedings of the Royal Society of London B 265, 359–66.

Vul, E., Goodman, N. D., Griffiths, T. L., Tenenbaum, J. B., 2009. One and done? Optimal decisions from very few samples. In: Annual Meeting of the Cognitive Science Society 38 (4). pp. 599–637.

Wainwright, M., Jordan, M., 2003. Graphical models, exponential families, and variational inference. Tech. rep., University of California, Berkeley.

Weiss, Y., Simoncelli, E., Adelson, E., 2002. Motion illusions as optimal percepts. Nature Neuroscience 5, 598–604.

Wilkinson, R. D., Feb. 2014. Accelerating ABC methods using gaussian processes. Tech. rep., University of Sheffield, *http://arxiv.org/abs/1401.1436*.

Wood, F., Griffiths, T. L., Ghahramani, Z., 2006. A non-parametric bayesian method for inferring hidden causes. In: Uncertainty in Artificial Intelligence 22. pp. 536–543.

Yuille, A., Kersten, D., 2006. Vision as Bayesian inference: analysis by synthesis? Trends in Cognitive Sciences 10 (7), 301–308.

Zhang, Y., Duchi, J. C., Wainwright, M. J., 2014. Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates. Tech. rep., University of California, Berkeley, *ttp://arxiv.org/abs/1305.5029*.

Zhou, M., Chen, H., Paisley, J., Ren, L., Sapiro, G., Carin, L., 2009. Non-parametric Bayesian dictionary learning for sparse image representations 1. In: NIPS Workshop.

Zylberberg, J., Murphy, J., Deweese, M., 2011. A Sparse Coding Model with Synaptically Local Plasticity and Spiking Neurons Can Account for the Diverse Shapes of V1 Simple Cell Receptive Fields. PLoS Computational Biology 7 (10), e1002250.

Appendix A

Contributions

Many aspects of my thesis have been published previously in peer-reviewed venues in the forms of journal articles, conference proceedings, and (extended) abstracts. My contributions to each are listed below.

1. **Shelton, J.**, Gasthaus, J., Dai, Z., Lücke, J., Gretton, A., 2017. GP-select: Accelerating EM using adaptive subspace preselection. Neural Computation 29 (8), 2177–2202.
   I shared in the development of the main idea and its theoretical formulation, along with Gasthaus and Gretton. Lücke contributed later to theoretical aspects of the work and made suggestion for the experimental part. I wrote the original code that was adapted for each experiment, Gasthaus helped improve the efficiency of this code, Dai implemented this in the code for experiments using his model. I ran most experiments (all sparse coding and mixture model experiments) and created the corresponding graphics. Dai also ran a significant portion of the experiments and generated the corresponding graphics. I wrote a large part of the manuscript, and all authors took part in its composition.

2. **Shelton, J.**, Sheikh, A.-S., Bornschein, J., Sterne, P., Lücke, J., 2015. Nonlinear Spike-and-Slab Sparse Coding for Interpretable Image Encoding. PLOS ONE 10 (5), 1–25.
   This is the journal extension of [6] and [5], the author contributions for which are described below. For this extended work, experiments were run by myself and Sheikh. I created all of the new graphics and wrote all new parts of the manuscript with revisions and additions made by Lücke. Bornschein helped design the new experiments and contributed to code to generate the corresponding data. Sterne proof-read and polished the writing.

3. **Shelton, J.**, Gasthaus, J., Dai, Z., Lücke, J., Gretton, A., 2014. GP-select: Accelerating EM using adaptive subspace preselection. Women in Machine Learning Workshop (WiML 2014) in conjunction with NIPS, Montreal, Quebec.
   This abstract corresponds to preliminary versions of the work in [1], for which the same author contributions apply. I created the poster and Jan Gasthaus presented it.

4. Lücke, J., **Shelton, J.**, Bornschein, J., Sterne, P., Berkes, P., Sheikh, A.-S., 2013. Combining Feed-Forward Processing and Sampling for Neurally Plausible Encoding Models. Computational and Systems Neuroscience 12.
   This abstract was based off of the work in [6] and [8], the authors' contributions for which are listed below. Lücke wrote the abstract, created the poster and presented the poster.

5. **Shelton, J.**, Sheikh, A.-S., Sterne, P., Bornschein, J., Lücke, J., 2013. Nonlinear Spike-and-Slab Sparse Coding for Interpretable Image Encoding. NIPS Workshop on High-dimensional Statistical Inference in the Brain.
   This abstract was based off of the work in [6] and formed the preliminary basis for the work in [2]. Contributions as listed for [6] apply. For this abstract I ran additional experiments with help from Sheikh, and created the poster. I presented the poster jointly with Bornschein.

6. **Shelton, J.**, Sterne, P., Bornschein, J., Sheikh, A.-S., Lücke, J., 2012. Why MCA? Nonlinear Spike-and-Slab Sparse Coding with Spike-and-Slab Prior for Neurally Plausible Image Encoding. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (Eds.), Advances in Neural Information Processing Systems 25. pp. 2285–2293.
   Lücke proposed the model in discussions with myself, Sterne, Bornschein and Sheikh. The model parameter derivations were done largely by Sterne with contributions from Bornschein and myself. The sampling method was developed by Sterne, Bornschein, Sheikh, and myself. I designed and ran several experiments, created several of the graphics, and wrote large parts of the manuscript.

7. **Shelton, J.**, Sterne, P., Bornschein, J., Sheikh, A.-S., Lücke, J., 2012. Why MCA? Nonlinear Spike-and-Slab Sparse Coding with Spike-and-Slab Prior for Neurally Plausible Image Encoding. Women in Machine Learning Workshop (WiML 2012) in conjunction with NIPS, Lake Tahoe, Nevada.
   This abstract and its author contributions correspond to the work in [6]. I presented the poster.

8. **Shelton, J.**, Bornschein, J., Sheikh, A.-S., Berkes, P., Lücke, J., 2011. Select and Sample - A Model of Efficient Neural Inference and Learning. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. (Eds.), Advances in Neural

Information Processing Systems 24. pp. 2618–2626.
Berkes and Lücke proposed the basic idea in discussions primarily with myself and with contributions by all authors. I wrote the preliminary code which was optimized and parallelized by Bornschein. Bornschein and I ran all experiments and generated the corresponding graphics, with help from Sheikh. All authors contributed to the manuscript; Berkes and Lücke wrote the majority of the introduction and discussion, I described most of the experiments and results, the sampler with Bornschein, and Bornschein described neural comparisons experiments.

9. Dai, Z., **Shelton, J.**, Bornschein, J., Sheikh, A.-S., Lücke, J., 2011. Combining Approximate Inference Methods for Efficient Learning on Large Computer Clusters. NIPS workshop on Big Learning: Algorithms, Systems, and Tools for Learning at Scale.
This abstract included the results of [8] as one of our described tools for learning at scale. I wrote large parts of the manuscript; Dai, Bornschein, and Sheikh each contributed a section to the manuscript taken from their own research. I helped create the poster and every author added a section to the poster corresponding to their contributions to the manuscript. Zhenwen gave an oral presentation and presented the poster.

10. **Shelton, J.**, Bornschein, J., Sheikh, A.-S., Berkes, P., Lücke, J., 2011. Select and Sample - A Model of Efficient Neural Inference and Learning. Women in Machine Learning Workshop (WiML 2011) in conjunction with NIPS, Malaga, Spain.
This abstract and its author contributions correspond to the work in [8]. I presented the poster.