GP-Select: Accelerating EM Using Adaptive Subspace Preselection

Jacquelyn A. Shelton
jacquelyn.ann.shelton@gmail.com
Technical University Berlin, 10587 Berlin, Germany

Jan Gasthaus
jgasthaus@gmail.com
University College London, Gatsby Unit, London W1T 4JG, U.K.,
and Amazon Development Center, 10178 Berlin, Germany

Zhenwen Dai
z.dai@sheffield.ac.uk
University of Sheffield, Sheffield, South Yorkshire S10 2TN, U.K.

Jörg Lücke
joerg.luecke@uol.de
University of Oldenburg, 26129 Oldenburg, Germany

Arthur Gretton
arthur.gretton@gmail.com
University College London, Gatsby Unit, London W1T 4JG, U.K.

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, for example, 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 expectation-maximization state via gaussian process regression. This is in contrast to 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.

1 Introduction

Inference in probabilistic graphical models can be challenging in situations where there are large numbers of hidden variables, each of which may take on one of several state values. The expectation-maximization (EM) algorithm is widely applied to learn model parameters when hidden variables are present; however, inference can quickly become intractable as the dimensionality of hidden states increases. Consider, for instance, the floor of a nursery with different toys scattered on it 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 100 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 100 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 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 structures. Graphical models for such data are often a central building block for tasks such as denoising (Elad & Aharon, 2006; Titsias & Lázaro-Gredilla, 2011), inpainting (Mairal, Bach, Ponce, & Sapiro, 2009; Mairal, Bach, Ponce, Sapiro, & Zisserman, 2009; Titsias & Lázaro-Gredilla, 2011), classification (Raina, Battle, Lee, Packer, & Ng, 2007), and collaborative filtering (Titsias & Lázaro-Gredilla, 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).

Expectation truncation (ET) (Lücke & Eggert, 2010) is an approximate EM algorithm for accelerating inference and learning in graphical models with many latent variables. 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). For example, considering the toy example, we could 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 (e.g., combinations of a restricted number of toys) of the remaining toys. More features and more refined features would allow a reduction to still smaller sets of toys, until the combinatorics of these selected toys becomes computationally tractable. The selection function of expectation truncation mathematically models the process of selecting the relevant hidden variables (the relevant toys), while truncated posterior distributions then model their remaining combinatorics.

In previous work, functions to select states of high posterior mass were derived individually for each graphical model of interest, for example, by taking upper bounds or noiseless limits (Lücke & Eggert, 2010; Shelton, Sterne, Bornschein, Sheikh, & Lücke, 2012; Bornschein, Henniges, & Lücke, 2013; Henniges, Turner, Sahani, Eggert, & Lücke, 2014; Sheikh, Shelton, & Lücke, 2014). The crucial underlying assumption remains that when EM has converged, the posterior mass is concentrated in small volumes of the latent state space (see, e.g., Lücke & Eggert, 2010; Sheikh et al., 2014, for discussions). We can expect the approximation to be accurate only if restricting the combinatorics does not miss large parts of posterior mass. This property is observed to hold, however, for many types of data in the auditory, visual, or general pattern recognition domains.

The definition of appropriate selection functions for basic graphical models such as the nursery floor example is already nontrivial. For models incorporating more detailed data properties, the definition of selections functions becomes still more demanding. For visual data, models that also capture mutual object occlusions (Henniges et al., 2014) or the object position (Dai & 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).

In the work presented in this letter, we propose a generalization of the expectation truncation (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; it 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 & 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 retrained 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. 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, in section 5.3, we 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 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 modeling 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 & Lücke, 2014). The performance of our inference algorithm matches that of the complex hand-engineered selection function of previous work, while being straightforward to implement and having a far lower computational cost.

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, Hinton, Neal, & Zemel, 1995) and its bottom-up connections trained using the wake-sleep algorithm (Hinton, Dayan, Frey, & Neal, 1995). More

---

[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.

recently, the idea has surfaced in the context of learning variational distributions with neural networks (Kingma & Welling, 2014). A two-stage inference procedure has been discussed in the context of computer vision (Yuille & Kersten, 2006) and neural inference (Körner, Gewaltig, Körner, Richter, & Rodemann, 1999). Recently, researchers (Mnih & 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, Welch, Mitchell, & Wynn, 1989; Currin, Mitchell, Morris, & Ylvisaker, 1991), to learn data-driven kernel functions for recommendation systems (Schwaighofer, Tresp, & Yu, 2004), and recently for quantum chemistry (Rupp, Tkatchenko, Müller, & von Lilienfeld, 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), aid in making Metropolis-Hastings (MH) decisions (Meeds & Welling, 2014), and to model the discrepancies between simulated/or observed data in parameter space simplification (Gutmann & 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/or intractable probabilistic models. None of the cited prior work addresses our problem setting: the selection of relevant latent variables by learning a nonparametric relevance function, for use in ET.

3 Variable Selection for Accelerated Inference

We denote the observed data by the $D \times N$ matrix $\mathbf{Y} = (\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(N)})$, where each vector $\mathbf{y}^{(n)} = (y_1^{(n)}, \dots, y_D^{(n)})^{\mathrm{T}}$ is the nth observation in a D-dimensional space. Similarly, we define corresponding binary latent variables by the matrix $\mathbf{S} = (\mathbf{s}^{(1)}, \dots, \mathbf{s}^{(N)}) \in \{0, 1\}^{H \times N}$, where each $\mathbf{s}^{(n)} = (s_1^{(n)} \dots, s_H^{(n)})^{\mathrm{T}} \in \{0, 1\}^H$ is the nth vector in the H-dimensional latent space, and for each individual hidden variable $h = 1, \dots, H$, the vector $\mathbf{s}_h = (s_h^{(1)} \dots, s_h^{(N)}) \in \{0, 1\}^N$. The number of dimensions in the reduced latent space is 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, Henniges, Eggert, & Lücke, 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

$$p(\mathbf{s}^{(n)}|\mathbf{y}^{(n)}, \Theta) = \frac{p(\mathbf{s}^{(n)}|\Theta)\, p(\mathbf{y}^{(n)}|\mathbf{s}^{(n)}, \Theta)}{\sum_{\mathbf{s}'} p(\mathbf{s}'\,|\Theta)\, p(\mathbf{y}^{(n)}|\mathbf{s}', \Theta)}. \tag{3.1}$$

3.1 Selection via Expectation Truncation in EM. Expectation maximization (EM) is an iterative algorithm to optimize the model parameters of a given graphical model (see, e.g., Dempster, Laird, & Rubin, 1977; Neal & Hinton, 1998). 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.

Expectation truncation (ET) is a meta-algorithm, that improves convergence of the EM algorithm (Lücke & 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 $s_1^{(n)}, s_2^{(n)}, \ldots, s_H^{(n)}$ is 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 $I \subseteq \{1, 2, \ldots, H\}$ of size H' ($H' \ll H$) of salient variables, which in turn is used to define a subset $\mathcal{K}_n = \{\mathbf{s} \mid \mathbf{s} \in 2^H \wedge \forall h \notin I : s_h = 0\} \subseteq \{0, 1\}^H$ of the possible state configurations of the latent variables $\mathbf{s}^{(n)}$ for each data point. This subset contains only state configurations where the values of the variables not identified to be relevant by the selection function are fixed to 0. The posterior distribution in equation 3.1 can then be approximated by a truncated posterior distribution, computed on the reduced support,

$$p(\mathbf{s}^{(n)}|\mathbf{y}^{(n)}, \Theta)$$

$$\approx q_n(\mathbf{s}^{(n)}; \Theta) = \frac{p(\mathbf{s}^{(n)}, \mathbf{y}^{(n)}|\Theta)\, \mathbb{I}(\mathbf{s}^{(n)} \in \mathcal{K}_n)}{\sum_{\mathbf{s}' \in \mathcal{K}_n} p(\mathbf{s}', \mathbf{y}^{(n)}|\Theta)}, \tag{3.2}$$

where $\mathbb{I}(\mathbf{s} \in \mathcal{K}_n) = 1$ if $\mathbf{s} \in \mathcal{K}_n$ is true and 0 otherwise. In other words, equation 3.2 is proportional to equation 3.1 if $\mathbf{s}^{(n)} \in \mathcal{K}_n$ (and zero otherwise), so that the approximate posterior $q_n(\mathbf{s}^{(n)}; \Theta)$ assigns zero mass to states $\mathbf{s}^{(n)} \notin \mathcal{K}_n$. The set $\mathcal{K}_n$ contains only states for which $s_h = 0$ for all h that are not selected, that is, all states where $s_h = 1$ for some nonselected h are assigned zero probability. This means that there are fewer terms in the denominator of equation 3.2 compared with equation 3.1, thus reducing the computational complexity. Equation 3.2 still remains proportional to equation 3.1 for the remaining states $\mathbf{s} \in \mathcal{K}_n$, however. As there are only $2^{H'}$ terms
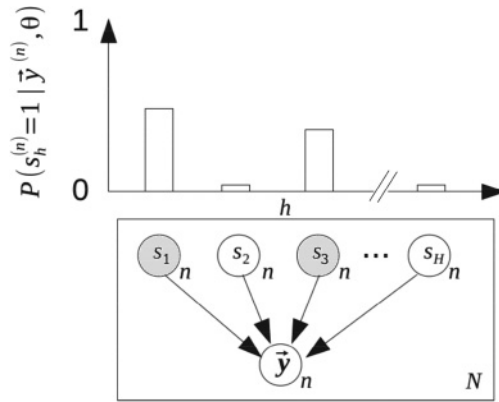
Figure 1: Illustration of the affinity function for selection. The affinity approximates the marginal posterior probability of each $h = 1, \ldots, H$ latent variable (top), which corresponds to the most relevant variables for a given data point $\mathbf{y}^{(n)}$ (bottom). Here, the variables s_1 and s_3 yield high affinity and would thus be considered relevant for $\mathbf{y}^{(n)}$.

in the sum over $\mathcal{K}_n$, computing this posterior approximation is much more efficient than computing the exact normalizing constant for the full posterior (containing 2^H terms). The number of latent dimensions to select, H', is chosen based on the computing resources available: as large as resources allow in order to be closer to true EM, although empirically it has been shown that much smaller values suffice (see, e.g., Sheikh et al., 2014, on complexity-accuracy trade-offs).

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

$$f_h(\mathbf{y}^{(n)}) = \hat{p}_h^{(n)} \approx p_h^{(n)} \equiv p(s_h^{(n)} = 1|\mathbf{y}^{(n)}, \Theta), \tag{3.3}$$

meaning that the relevant variables will have greater marginal posterior probability $p_h^{(n)}$. (See Figure 1 for a simplified illustration.) When the latent variables $s_1^{(n)}, \ldots, s_H^{(n)}$ 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 $p(s_1, \ldots s_h | \mathbf{y}, \Theta)$ in the case when

a subset of latents has values clamped to zero: $s_h = 0$ for all $h \notin I$ (compare equation 3.2). 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:

$$\sum_{\mathbf{s} \text{ with } s_h=0 \text{ for all } h \notin I} p(s_1, \dots s_H \mid \mathbf{y}, \Theta) = 1 - \prod_{h \notin I} p(s_h = 1 \mid \mathbf{y}, \Theta). \qquad (3.4)$$

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 $p(s_h = 1 \mid \mathbf{y}, \Theta)$, then discarding those $(H - H')$ marginal with the lowest values is equivalent to discarding the space with the least posterior mass (compared to discarding with regard to 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 $\hat{p}_{h=1}^{(n)}, \dots, \hat{p}_H^{(n)}$ from the affinity function $\mathbf{f}(\mathbf{y}^{(n)}) = (f_1(\mathbf{y}^{(n)}), \dots, f_H(\mathbf{y}^{(n)}))$, we define $\gamma(\hat{\mathbf{p}}^{(n)})$ 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. $\gamma(\hat{\mathbf{p}}^{(n)})$ thus returns the H' selected variable indices I chosen by the affinity to be relevant to the nth data point. To ensure that there is a nonzero probability of selecting each variable per EM iteration, 10% of the H' indices are uniformly chosen from H at random. This prevents the possible propagation of errors from $q_{(n)}$ continuously assigning small probabilities to a variable s_h in early EM iterations. The rationale for this is that the optimization of $q_{(n)}$ in early iterations of EM starts from randomly initialized s_h. If the affinity function itself is based on the posterior approximation (as it will be in the algorithm described in section 4), it has a tendency to not select indices that were previously not selected. Thus, in order for selection-based EM to not "get stuck," it is important to select a few extra hidden indices randomly to give the algorithm an opportunity to evaluate possibly unused variables that might be relevant for $\mathbf{y}^{(n)}$.

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 nonrelevant variable states s_h for all variables $h \notin I$ are effectively set to 0 in equation 3.2 by not being present in the state set $\mathcal{K}_n$. For example, say that there are five s_h, where $h \in \{1, \dots, 5\}$. 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 $\mathcal{S} : \mathrm{R}^D \mapsto 2^{\{1,\ldots,H\}}$ 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 $p(\mathbf{s} \mid \mathbf{y})$ 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

$$\mathcal{S}(\mathbf{y}^{(n)}) = \mathcal{I}\left[\gamma\left[\mathbf{f}(\mathbf{y}^{(n)})\right]\right] = \mathcal{K}_n. \tag{3.5}$$

To summarize, the main task is to formulate a general data-driven function to identify relevant latent variables and select the corresponding set of states $\mathcal{K}_n$. This is performed using GP regression in order to compute the truncated posterior equation 3.2 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 3.5 to perform this selection.

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 equation 3.5 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 equation 3.2. The truncated posterior can be computed using any standard inference method, such as exact inference or, for example, 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 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., 2012; Dai & Lücke, 2012a, 2012b; Bornschein et al., 2013; Sheikh et al., 2014; Shelton, Sheikh, Bornschein, Sterne, & Lücke, 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 blackbox and model-free way based on the data. We learn $\mathcal{S}$ using gaussian process (GP) regression (Rasmussen & Williams, 2005), a flexible nonparametric model that scales cubicly[2] 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.

---

[2]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 5.3 for details).

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 $\langle \mathbf{S} \rangle$ from 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 $\mathcal{D} = \{(\mathbf{y}^{(n)}, \langle \mathbf{s}^{(n)} \rangle_{q_n(\mathbf{s}^{(n)})} | n = 1, \ldots, N\}$, where we recall that $q_n(\mathbf{s}^{(n)})$ is the approximate posterior distribution for $\mathbf{s}^{(n)}$ in equation 3.2. 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 want only 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 with regard to 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 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 leave-one-out (LOO) prediction. It can be shown that this can be implemented efficiently (see section 5.4.2 in Rasmussen & Williams, 2005), and we use this result to update the predicted affinity as follows:

$$\hat{p}_h^{(n)} \leftarrow \langle s_h^{(n)} \rangle_{q_n} - \frac{[K^{-1} \langle \mathbf{s}_h \rangle_{q_n}]_{nn}}{[K^{-1}]_{nn}}. \tag{4.1}$$

Equation 4.1 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 4.1 for $\mathbf{f}$ in the affinity-based selection function, equation 3.5,

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

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 $\hat{\mathbf{p}}_t^{(n)}$: equation 4.1
         compute subset of relevant states $\mathcal{S}$: equation 3.5
         compute truncated posterior $q_{n,t}(\mathbf{s}^{(n)})$ in E-step: equation 3.2
         update model parameters in M-step, e.g. as in section 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

---

5 Experiments

We apply our GP-select inference approach to five probabilistic generative models. First, we consider three sparse coding models (binary sparse coding, spike-and-slab, and 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 to provide a functional intuition of approach and 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, Exarchakis, & Lücke, 2013; Dai & Lücke, 2012a, 2014).

5.1 Sparse Coding Models. Many types of natural data are composed of potentially many component types, but any data point often contains only a very small number of this potentially large set of components. For the example of toys on the nursery floor, for instance, there are many different toys that can potentially be in a given image, but typically only a relatively small number of toys actually appear 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, only relatively few keys (typically much fewer than 10) actually generated the sound. Sparse coding algorithms model such data properties by providing a large number of hidden variables (potential data components) but assigning nonzero (or significantly different from zero) values to only a small subset of components (those actually appearing). Sparse coding algorithms are typically used for tasks such as denoising (Elad & Aharon, 2006; Mairal, Bach, Ponce, Sapiro et al., 2009), inpainting (Mairal, Bach, Ponce, & Sapiro, 2009; Mairal, Bach, Ponce, Sapiro, & Zisserman, 2009; Titsias & Lázaro-Gredilla, 2011), classification (LeCun, n.d.; Titsias & Lázaro-Gredilla, 2011; Raina et al., 2007, e.g., MNIST data set, http://yann.lecun.com/exdb/mnist/, and the flowers data set, http://www.robots.ox.ac.uk/~vgg/data/flowers/), transfer learning (Raina et al., 2007), and

collaborative filtering (Titsias & Lázaro-Gredilla, 2011) and are important models for neurosensory processing (Olshausen & Field, 1997; Zylberberg, Murphy, & Deweese, 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, Puertas, Bornschein, Eggert, & Lücke, 2010); Bornschein et al., 2013; Sheikh et al., 2014) or selection combined with Gibbs sampling (Shelton et al., 2011, 2012, 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 handcrafted 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.

The models and their parameters are:

- *Binary sparse coding*

$$\text{latents:} \quad \mathbf{s} \sim \text{Bern}(\mathbf{s}|\pi) = \prod_{h=1}^{H} \pi^{s_h} \left(1 - \pi\right)^{1-s_h}$$

$$\text{observations:} \quad \mathbf{y} \sim \mathcal{N}(\mathbf{y}; W\mathbf{s}, \sigma^2 I)$$

$$\text{parameters:} \quad W = \left(\sum_{n=1}^{N} \mathbf{y}^{(n)} \langle \mathbf{s} \rangle_{q_n}^{T}\right) \left(\sum_{n=1}^{N} \langle \mathbf{s}\,\mathbf{s}^{T} \rangle_{q_n}\right)^{-1}$$

$$\sigma^2 = \frac{1}{ND} \sum_{n} \left\langle \left\|\mathbf{y}^{(n)} - W\,\mathbf{s}\right\|^2 \right\rangle_{q_n}$$

$$\pi = \frac{1}{N} \sum_{n} |\langle \mathbf{s} \rangle_{q_n}|, \quad \text{where } |\mathbf{x}| = \frac{1}{H} \sum_{h} x_h,$$

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

- *Spike-and-slab sparse coding*

$$\text{latents:} \quad \mathbf{s} = \mathbf{b} \odot \mathbf{z} \quad \text{where}$$

$$\mathbf{b} \sim \text{Bern}(\mathbf{b}|\pi) \text{ and } \mathbf{z} \sim \mathcal{N}(\mathbf{z};\, \mu, \Sigma_h)$$

$$\text{observations:} \quad \mathbf{y} \sim \mathcal{N}(\mathbf{y}; W\mathbf{s}, \sigma^2 I)$$

$$\text{parameters:} \quad W = \frac{\sum_{n=1}^{N} \mathbf{y}^{(n)} \langle \mathbf{s} \odot \mathbf{z} \rangle_{n}^{T}}{\sum_{n=1}^{N} \langle (\mathbf{s} \odot \mathbf{z})(\mathbf{s} \odot \mathbf{z})^{T} \rangle_{n}}$$

$$\pi = \frac{1}{N} \sum_{n=1}^{N} \langle \mathbf{s} \rangle_n$$

$$\sigma^2 = \sum_{n=1}^{N} \left[\langle (\mathbf{s} \odot \mathbf{z})(\mathbf{s} \odot \mathbf{z})^{\mathrm{T}} \rangle_n - \langle \mathbf{s}\,\mathbf{s}^{\mathrm{T}} \rangle_n \odot \mu\mu^{\mathrm{T}} \right]$$

$$\odot \left(\sum_{n=1}^{N} \left[\langle \mathbf{s}\,\mathbf{s}^{\mathrm{T}} \rangle_n \right] \right)^{-1}$$

$$\mu_{pr} = \frac{\sum_{n=1}^{N} \langle \mathbf{s} \odot \mathbf{z} \rangle_n}{\sum_{n=1}^{N} \langle \mathbf{s} \rangle_n}$$

$$\sigma_{pr}^2 = \frac{1}{N} \sum_{n=1}^{N} \left[\mathbf{y}^{(n)} (\mathbf{y}^{(n)})^{\mathrm{T}} \right.$$

$$\left. - W \left[\langle (\mathbf{s} \odot \mathbf{z}) \rangle_n \langle (\mathbf{s} \odot \mathbf{z}) \rangle_n^{\mathrm{T}} \right] W^{\mathrm{T}} \right]$$

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

- *Nonlinear spike-and-slab sparse coding*

$$\text{latents:} \quad \mathbf{s} = \mathbf{b} \odot \mathbf{z} \quad \text{where} \quad \mathbf{b} \sim \text{Bern}(\mathbf{b}|\pi)$$

$$\text{and } \mathbf{z} \sim \mathcal{N}(\mathbf{z};\ \mu_{\mathbf{pr}}, \sigma^2)$$

$$\text{observations:} \quad \mathbf{y} \sim \mathcal{N}(\mathbf{y};\ \max_h \{s_h \mathbf{W}_h\}, \sigma^2 I)$$

$$\text{parameters:} \quad \hat{W}_{hd} = \frac{\langle s_h y_d \rangle^*}{\langle s_d^2 \rangle^*} \qquad\qquad \hat{\pi} = \langle \mathbb{I}(s) \rangle$$

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

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

Where expectations $\langle\,.\,\rangle^*$ mean:

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

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 & Sahani, 2008; Shelton et al., 2012; Bornschein et al., 2013), 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 3.2 using handcrafted 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 3.5, 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 3.5, was the cosine similarity between the weights $\mathbf{W}_h$ (e.g., dictionary elements, components) 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 nonzero. The goal of these experiments is to demonstrate the performance of GP-select and the effects or 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 $N = 2000$ data points consisting of $D = 5 \times 5 = 25$ observed dimensions and $H = 10$ latent components according to each of the models **A to 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 two bars, that is, 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 handcrafted selection function, (2) GP-select using a linear covariance kernel, (3) GP-select using an 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 term the *composition kernel*. As hyperparameters of kernels are learned, the composition kernel experiment 4, can adapt itself to the data and only use the kernel components required. (See Rasmussen & Williams, 2005 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, equaiton 3.2, but as exact inference is analytically intractable in model **C**, inference was performed by drawing Gibbs samples from the
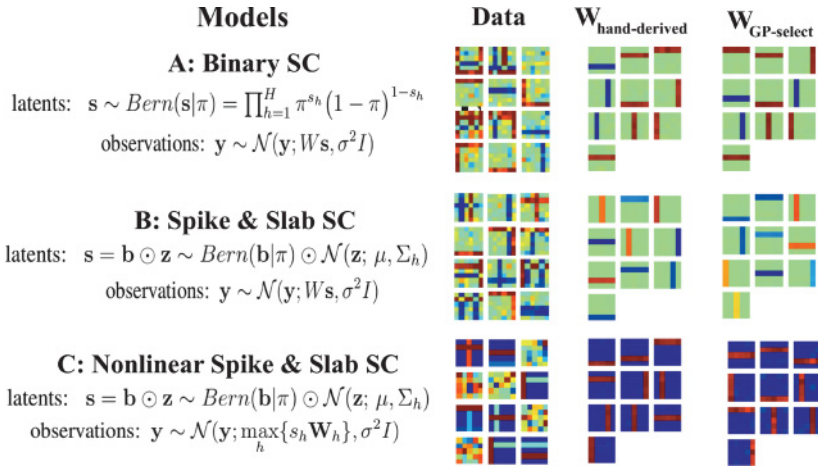
Models   **Data** $W_{\text{hand-derived}}$ $W_{\text{GP-select}}$

A: Binary SC

latents: $\mathbf{s} \sim Bern(\mathbf{s}|\pi) = \prod_{h=1}^{H} \pi^{s_h}(1-\pi)^{1-s_h}$

observations: $\mathbf{y} \sim \mathcal{N}(\mathbf{y}; W\mathbf{s}, \sigma^2 I)$

B: Spike & Slab SC

latents: $\mathbf{s} = \mathbf{b} \odot \mathbf{z} \sim Bern(\mathbf{b}|\pi) \odot \mathcal{N}(\mathbf{z}; \mu, \Sigma_h)$

observations: $\mathbf{y} \sim \mathcal{N}(\mathbf{y}; W\mathbf{s}, \sigma^2 I)$

C: Nonlinear Spike & Slab SC

latents: $\mathbf{s} = \mathbf{b} \odot \mathbf{z} \sim Bern(\mathbf{b}|\pi) \odot \mathcal{N}(\mathbf{z}; \mu, \Sigma_h)$

observations: $\mathbf{y} \sim \mathcal{N}(\mathbf{y}; \max_h\{s_h \mathbf{W}_h\}, \sigma^2 I)$

Figure 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 handcrafted 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 handcrafted selection function). In all cases, the model using the GP-select function converged to the ground-truth solution, just as the handcrafted selection functions did.

truncated space (Shelton et al., 2011, 2012, 2015). We run all models until convergence.

Results are shown in Figure 2. In all experiments, the GP-select approach was able to infer ground-truth parameters as well as the handcrafted 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 handcrafted function, its affinity function using GP regression, equation 4.1 learned a similar enough function to quickly yield identical results. Furthermore, in the model with a less straightforward handcrafted 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 of 10 repetitions, whereas the handcrafted 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 handcrafted 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.

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 GMM's flexibility allow 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

$$p(\mathbf{y}^{(n)}|\mu_{\mathbf{c}}, \sigma_{\mathbf{c}}, \pi) = \sum_{c=1}^{C} \mathcal{N}(\mathbf{y}^{(n)}; \mu_{\mathbf{c}}, \sigma_{\mathbf{c}}) \pi_c, \tag{5.1}$$

where C is the number of mixture components; the task is to assign each data point to its latent cluster.

For training data for GP regression, we used $\mathcal{D} = \{(\mathbf{y}^{(n)}, \langle s_h^{(n)} \rangle_{q_n})|n = 1, \ldots, N\}$, 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 function according to equation 3.5 with affinity f defined by GP regression equation 4.1, 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 5.1: 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 two-dimensional observed data ($\mathbf{y}^{(n)} \in \mathbb{R}^{D=2}$) from $C = 3$ clusters—first with randomly assigned cluster means and then 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 nonmixture 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 3.
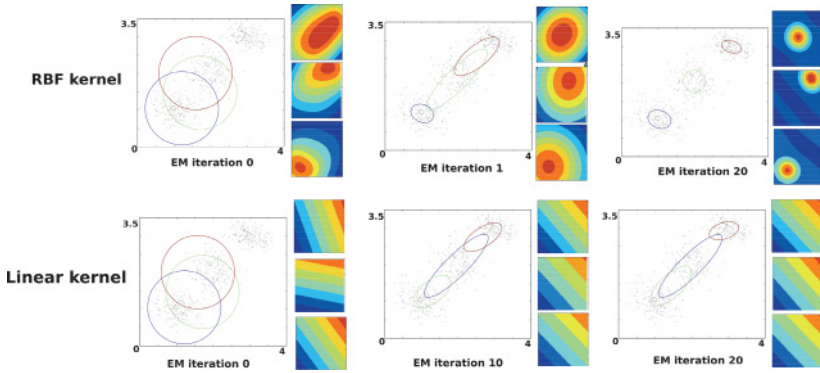
Figure 3: Gaussian mixture model results using GP-select (selection of $C' = 2$ in a $C = 3$ class scenario) for inference. Progress of the inference is shown using (top row) an RBF covariance kernel in the regression and (bottom row) a linear covariance kernel. For each iteration shown, we see (1) the observed data and their inferred cluster assignments and (2) the C corresponding GP regression functions learned or 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.

On these data, the linear GP regression prediction cannot correctly assign the data to their clusters (as seen in Figure 3B), but the nonlinear approach successfully and easily finds the ground-truth clusters (Figure 3A). 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 3B). The RBF kernel cluster assignments remained at the optimal solution even with the 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.

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.

5.3.1 Model. Translation-invariant models address the problem that, for example, visual objects can appear in any location of an image. Probabilistic

models for translation invariance are particularly appealing as they allow separately inferring 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 & Lücke, 2012b, 2014; Dai et al., 2013). The selection function used so far for reducing 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 is 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 3.2.

This selection system is very costly: the selection function has parameters that 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 the GP-selection function, 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 components appearing at all possible locations. Therefore, the input to the GP-selection function is the image to be inferred, and the output is a score for each possible location of each component in the model. For example, when learning 10 components in a $D = 30 \times 30$ pixel image patch, the output dimensionality of GP-select is 9000. 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 additional computations are necessary for an automatic selection function like GP-select, for instance, due to the adjustment of its parameters, there are many options to reduce computational costs. First, we may approximate the full $N \times N$ gram matrix by an incomplete Cholesky approximation (Fine & Scheinberg, 2001), resulting in a cost of $O(N \times Q)$, where $Q << N$ 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

Figure 4: COIL data set (Nene, Nayar, & Murase, 1996): A handful of data points used in experiments with the translation-invariant occlusive (InvECA) model, showing the occluding objects to be learned.

plus infrequent updates will have the following benefits: a factor of five speed-up for infrequent updates and a factor of $(N - Q)^2$ speed-up from incomplete Cholesky, where Q is the rank of the Cholesky approximation and N is the number of original data points.

5.3.2 COIL Data Set. We apply our GP-selection function to the invariant exclusive component analysis (InvECA) model (Dai & Lücke, 2012b; Dai et al., 2013). For our experiments, we consider an image data set used in previous work. Data were generated using objects from the COIL-100 image data set (Nene et al., 1996), taking 16 different objects, 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. The appearing objects were 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, $N = 2000$ images were generated in the data set (examples shown in Figure 4). The task of the InvECA model is to discover the visual components (i.e., 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 handcrafted 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 $H = 20$, 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).
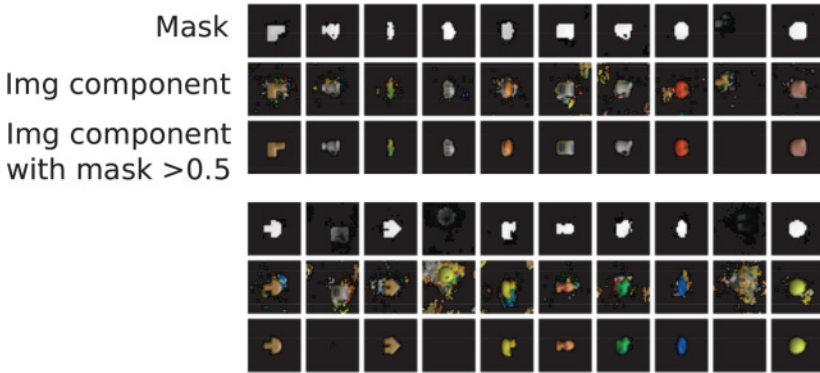
Figure 5: Image components and their masks learned by GP-select with the translation-invariant model. GP-select learned all objects in the data set. The first row shows the mask of each component, the second row shows the learned image components, and the third row shows only the area of the learned components that had a mask > 0.5. For the second three-row block of images, the same titles of the first three-row block hold. Enlarge and rotate (+90°) for best visibility.

5.3.3 Results. All four versions of the InvECA model using each of the selection functions considered successfully recover each individual object in our modified COIL image set. The learned object representations with GP-select are shown in Figure 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, and 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 locations to the location of the top candidate locations from each selection function. (See Figure 6 for a histogram of these distances and the corresponding 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 7. Figures 6 and 7 show that all four selection functions can accurately predict the locations of all the objects in the data set: the GP-select selection functions yield no loss in inference performance in
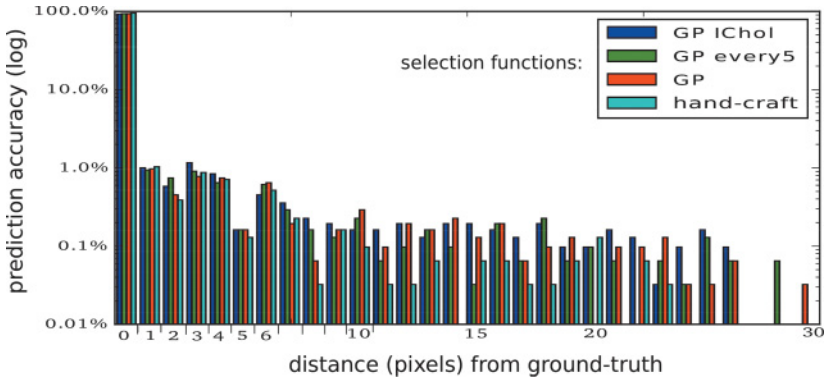
Figure 6: Prediction accuracy of the four selection functions in the InvECA model. Functions depicted in the figures: GP-select with no modifications (GP, red), the incomplete Cholesky decomposition (GP IChol, blue), with updated kernel hyperparameters every five EM iterations (GP every5, green), and with handcrafted selection (handcrafted, 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.
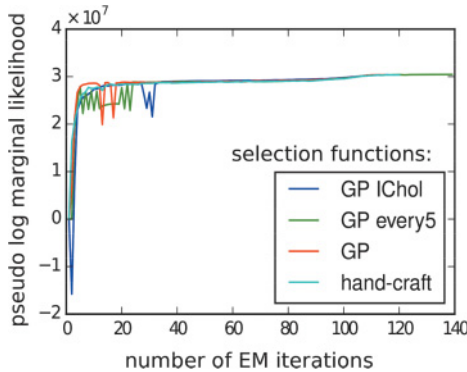

Figure 7: Baseline comparison of the four selection functions in the InvECA model. Functions depicted in the figures are identical to those in Figure 6. Shown: the convergence of the pseudo-log marginal likelihood (of the model parameters learned at each EM iteration) for the four selection functions 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 handcrafted function, and GP IChol represents a factor of 100 speedup versus GP, and GP every5 represents a factor of 5 speed-up.

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 five EM iterations, GP every5) have indistinguishable prediction accuracy on the task.

An analysis of the benefits indicates 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 handcrafted 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 handcrafted 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 hyperparameters for GP-select, it is important to note that the hyperparameters need not perfectly fit 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, computing time was reduced to 194 seconds (corresponding to the $(N - Q)^2$ speed-up, where Q is the rank of the Cholesky approximation), with minimal loss in accuracy. Furthermore, when updating the GP-select hyperparameters only every five iterations, average computing time was reduced by another one-fifth, again without loss in accuracy.

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 selection 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 in terms of both 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 handcrafted function for the complex hierarchical model of Dai et al. (2013).

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, Duchi, & Wainwright, 2014), or the problem may be solved in the primal via random Fourier features (Le, Sarlos, & Smola, 2013). Furthermore, there are many recent developments regarding the scaling up of GP inference to large-scale problems— for example, sparse GP approximation (Lawrence, Seeger, & Herbrich, 2002), stochastic variational inference (Hensman, Fusi, & Lawrence, 2013; Hensman, Rattray, & Lawrence, 2012), using parallelization techniques and GPU acceleration (Dai, Damianou, Hensman, & Lawrence, 2014), or in combination with stochastic gradient descent (Bottou & 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.

Acknowledgments

References

Bornschein, J., Henniges, M., & Lücke, J. (2013). Are V1 simple cells optimized for visual occlusions? A comparative study. *PLoS Computational Biology*, *9*(6), e1003062.

Bottou, L., & Bousquet, O. (2008). The tradeoffs of large scale learning. In J. C. Platt, D. Koller, Y. Singer, & S. T. Roweis (Eds.), *Advances in neural information processing systems*, *20* (pp. 161–168). Cambridge, MA: MIT Press.

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/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 C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Weinberger (Eds.), *Advances in neural information processing systems*, *26* (pp. 243–251). Red Hook, NY: Curran.

Dai, Z., & Lücke, J. (2012a). Autonomous cleaning of corrupted scanned documents: A generative modeling approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3338–3345). Piscataway, NJ: IEEE.

Dai, Z., & Lücke, J. (2012b). Unsupervised learning of translation invariant occlusive components. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2400–2407). Piscataway, NJ: IEEE.

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., Hinton, G. E., Neal, R. M., & Zemel, R. S. (1995). The Helmholtz machine. *Neural Computation*, *7*, 889–904.

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.

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

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

Fine, S., & Scheinberg, K. (2001). Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, *2*, 243–264.

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* (Technical Report). Helsinki: University of Helsinki. http://arxiv.org/abs/1501.03291.

Henniges, M., Puertas, G., Bornschein, J., Eggert, J., & Lücke, J. (2010). Binary Sparse Coding. In *Proceedings LVA/ICA*, LNCS 6365 (pp. 450–457). New York: Springer.

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 *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*. Bright Doc.

Hensman, J., Rattray, M., & Lawrence, N. D. (2012). Fast variational inference in the conjugate exponential family. In P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 25* (pp. 2897–2905). Red Hook, NY: Curran.

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

Kingma, D. P., & Welling, M. (2014). Efficient gradient-based inference through transformations between Bayes nets and neural nets. In *Proceedings of the 31th International Conference on Machine Learning* (pp. 1782–1790).

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

Lawrence, N., Seeger, M., & Herbrich, R. (2002). Fast sparse gaussian process methods: The informative vector machine. In S. Becker, S. Thrun, & K. Obermayer

(Eds.), *Advances in neural information processing systems, 15* (pp. 609–616). Cambridge, MA: MIT.

Le, Q., Sarlos, T., & Smola, A. J. (2013). Fastfood: Computing Hilbert space expansions in loglinear time. In *Proceedings of the 30th International Conference on Machine Learning* (pp. 244–252).

LeCun, Y. *MNIST database of handwritten digits*. http://yann.lecun.com/exdb/mnist/.

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*, 1227–1267.

Mairal, J., Bach, F., Ponce, J., & Sapiro, G. (2009). Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning* (pp. 689–696).

Mairal, J., Bach, F., Ponce, J., Sapiro, G., & Zisserman, A. (2009). Non-local sparse models for image restoration. In *IEEE 12th International Conference on Computer Vision* (pp. 2272–2279). Piscataway, NJ: IEEE.

Meeds, E., & Welling, M. (2014). GPS-ABC: Gaussian process surrogate approximate Bayesian computation. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence (UAI)* (pp. 593–602).

Mnih, A., & Gregor, K. (2014). Neural variational inference and learning in belief networks. In *Proceedings of the 31th International Conference on Machine Learning* (pp. 1791–1799).

Neal, R., & Hinton, G. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan (Ed.), *Learning in graphical models*. New York: Kluwer.

Nene, S. A., Nayar, S. K., & Murase, H. (1996). *Columbia Object Image Library (COIL-100)* (Technical Report, CUCS-006-96). New York: Department of Computer Science, Columbia University.

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

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

Rasmussen, C. E., & Williams, C. K. I. (2005). *Gaussian processes for machine learning*. Cambridge, MA: MIT Press.

Rupp, M., Tkatchenko, A., Müller, K.-R., & von Lilienfeld, O. A. (2012). Fast and accurate modeling of molecular atomization energies with machine learning. *Phys. Rev. Lett.*, *108*, 058301.

Sacks, J., Welch, W. J., Mitchell, T. J., & Wynn, H. P. (1989). Design and analysis of computer experiments. *Statist. Sci.*, *4*(4), 433–435.

Schwaighofer, A., Tresp, V., & Yu, K. (2004). Learning gaussian process kernels via hierarchical bayes. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems 17* (pp. 1209–1216). Cambridge, MA: MIT Press.

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., Bornschein, J., Sheikh, A.-S., Berkes, P., & Lücke, J. (2011). Select and sample: A model of efficient neural inference and learning. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 24* (pp. 2618–2626). Red Hook, NY: Curran.

Shelton, J., Sterne, P., Bornschein, J., Sheikh, A.-S., & Lücke, J. (2012). Why MCA? Nonlinear sparse coding with spike-and-slab prior for neurally plausible image encoding. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, 25 (pp. 2285–2293). Red Hook, NY: Curran.

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.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Narayanan Sundaram, M., … Adams, R. (2015). *Scalable Bayesian optimization using deep neural networks* (Technical Report). Cambridge, MA: Harvard University. http://arxiv.org/abs/1502 .05700.

Titsias, M., & Lázaro-Gredilla, M. (2011). Spike and slab variational inference for multi-task and multiple kernel learning. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *24* (pp. 2510–2518).

Wilkinson, R. D. (2014). *Accelerating ABC methods using gaussian processes*. (Technical Report). Sheffield: University of Sheffield. http://arxiv.org/abs/1401.1436.

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* (Technical Report). Berkeley University of California, Berkeley. http://arxiv.org/abs/1305.5029.

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.