GP-select: Accelerating EM using adaptive subspace preselection

Jacquelyn A. Shelton
TU Berlin
shelton@tu-berlin.edu

Jan Gasthaus
UCL Gatsby
j.gasthaus@ucl.ac.uk

Zhenwen Dai
University of Sheffield
z.dai@sheffield.ac.uk

Joerg Luecke
University of Oldenburg
joerg.luecke@uni-oldenburg.de

Arthur Gretton
UCL Gatsby
arthur.gretton@gmail.com

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 use 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 unfeasible. 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 selections were hand-designed for each problem setting. We show our approach to perform 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 a large number of hidden variables, each of which may take on one of several state values. The EM algorithm is widely applied for inference when hidden variables are present, however inference can quickly become intractable as the dimensionality increases.

Expectation truncation (ET) (Lücke & Eggert, 2010) is a meta algorithm for accelerating EM learning in graphical models, which restricts 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*. In previous work, functions to select states of high posterior mass were hand-crafted to model the properties of the graphical model of interest, using intuition from noiseless limits or upper-bound results specific to the model (Lücke & Eggert, 2010; Shelton *et al.* , 2012; Bornschein *et al.* , 2013; Sheikh *et al.* , 2014). The crucial underlying assumption remains that the posterior mass is finally concentrated in small volumes of the latent state space This property is observed to hold in many visual data and general pattern recognition settings.

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, dating back at least to the Helmholtz machine 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 & Welling, 2014). A two-stage inference has furthermore been discussed in the context of computer vision (Yuille & Kersten, 2006) and neural inference (Körner *et al.* , 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.

For more complex graphical models, notably the hierarchical model of (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 the major contribution

in several of the foregoing papers).

In the present work, we propose a generalization of the ET approach, where we avoid completely the challenge of problem-specific selection function design. Instead, we learn this function 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 states is preserved. We use Gaussian process regression (Rasmussen & Williams, 2005) to learn the selection function, though other regression techniques could also be applied. The main advantage of GPs is that they have an analytic one-shot learning procedure, and that learning different functions based on the same inputs is computationally cheap, which makes adaptive learning of a changing target function efficient. We term this part of our approach *GP-selection*. 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.

We have applied GP-selection in a number of experimental benchmarks. 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-selection can produce results with equal performance to the respective hand-picked 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-selection on a simple Gaussian mixture model, where we explicitly show the form of the learned regression function: this matches well with our expectations for the model, and again reveals that it may be essential to learn a nonlinear mapping. Finally, we present results for a recently published hierarcial model for translation invariant occlusion components analysis (Dai & Lücke, 2014). The performance of our inference algorithm matched that of the complex hand-engineered selection function of the previous work, while being straightforward to implement and having a far lower computational cost.

[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.

2 Variable selection for accelerated inference/learning

Expectation truncation (ET) is an extension of the expectation maximization (EM) algorithm (e.g. (Dempster *et al.* , 1977; Neal & Hinton, 1998)) 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.

Let $\mathbf{Y} = (\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(N)})$ denote the observed data vectors, where $\mathbf{y}^{(n)} = (y_1^{(n)}, \ldots, y_D^{(n)})^{\mathrm{T}}$ is the nth observation. Similarly we define corresponding binary latent variablesh[2] by $\mathbf{S} = (\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(N)})$ with $\mathbf{s}^{(n)} = (s_1^{(n)} \ldots, s_H^{(n)})^{\mathrm{T}} \in \{0,1\}^H$.

The main idea underlying ET is that for a given observation $\mathbf{y}^{(n)}$ only a subset of the H latent variables $s_h^{(n)}$ will be relevant, meaning that the marginal posterior probability $p_h^{(n)} \equiv p(s_h^{(n)} = 1 | \mathbf{y}^{(n)}, \Theta)$ exceeds some threshold. The selection function is used to identify this subset of variables, which in turn is used to define a subset $\mathcal{K}_n$ of the space set where all non-relevant variables are fixed to 0. This truncated posterior distribution, computed on reduced support, can be expressed:

$$p(\mathbf{s}^{(n)} | \mathbf{y}^{(n)}, \Theta) \approx q_n(\mathbf{s}^{(n)}; \Theta) = \frac{p(\mathbf{s}^{(n)}, \mathbf{y}^{(n)} | \, \Theta) \, \delta(\mathbf{s}^{(n)} \in \mathcal{K}_n)}{\displaystyle\int_{\mathbf{s}'^{(n)} \in \mathcal{K}_n} p(\mathbf{s}'^{(n)}, \mathbf{y}^{(n)} | \, \Theta) \, d\mathbf{s}'^{(n)}} \quad (1)$$

where $\mathcal{K}_n$ denotes the reduced set of relevant latent states for data point $\mathbf{y}^{(n)}$ and $\delta(\mathbf{s} \in \mathcal{K}_n) = 1$ if $\mathbf{s} \in \mathcal{K}_n$ and zero otherwise.

One way of constructing a selection function $\mathcal{S}^{(n)}$ is by ranking the latent variables according to an *affinity function* $f_h(\mathbf{y}^{(n)})$ which directly reflects the relevance of latent variable s_h. A natural choice for such a function is one that approximates the marginal posterior probability, e.g. we try to learn f such that $f_h(\mathbf{y}^{(n)}) \approx p_h^{(n)}$. When the H latent variables in the marginal posterior probability $p_{h=1}^{(n)}, \ldots, p_H^{(n)}$ are conditionally independent given a data point $\mathbf{y}^{(n)}$, this

[2]We restrict ourselves binary latent variables here, though discrete variables with higher cardinality can easily be used by converting them into binary variables.

affinity function correctly isolates the most relevant variables in the posterior and yet still can highlight relevant variables even when dependencies exist.

Using these concepts, we define a general selection function to isolate the relevant regions $\mathcal{K}_n$ as follows. First, we define $f_h(\mathbf{y}^{(n)})$ as a function that takes the input of a single data point $\mathbf{y}^{(n)}$ and predicts the probability that the latent variable s_h is equal to 1 for this particular data point, such that each data point $\mathbf{y}^{(n)}$ is associated with an H-dimensional vector of predictions $\mathbf{f}(\mathbf{y}^{(n)}) = (f_1(\mathbf{y}^{(n)}), \ldots, f_H(\mathbf{y}^{(n)}))$. Next, define $\gamma(\mathbf{s}^{(n)})$ as a function of each $\mathbf{s}^{(n)}$ that returns a sorted and reduced set of indices I corresponding to the highest probability states that $\mathbf{s}^{(n)} = 1$ (output from $\mathbf{f}(\mathbf{y}^{(n)})$). In other words, $\gamma(\mathbf{s}^{(n)})$ sorts all $h, ..., H$ of the $s_h^{(n)}$ variables in descending order of estimated relevance, and gives the sorted incides of the $H' < H$ 'most relevant' states. Finally, we define $\mathcal{I}(I)$ as a function of the sorted indices I that sets all 'non-relevant' variables $h \notin I$ for a given data point $\mathbf{y}^{(n)}$ to 0, and returns the set of H' selected relevant latent states' indices $\mathcal{K}_n$. Using f_h, $\mathcal{I}_n$ and γ, we can define a *selection function* $\mathcal{S}_h(\mathbf{y}^{(n)})$ to select subsets $\mathcal{K}_n$ per data point $\mathbf{y}^{(n)}$, with the goal of containing most of the probability mass $p(\mathbf{s} \,|\, \mathbf{y})$ while also being significantly smaller than the entire latent space. So far, the form of $\mathcal{S}_h(\mathbf{y}^{(n)})$ has been a hand-engineered deterministic function derived based on the structure of the considered model (see e.g. (Shelton *et al.* , 2011, 2012; Dai & Lücke, 2012a,b; Bornschein *et al.* , 2013; Sheikh *et al.* , 2014)). The *general selection function* can be expressed as

$$\mathcal{S}(\mathbf{y}^{(n)}) \;=\; \mathcal{I}\left[\gamma\left[\mathbf{f}(\mathbf{y}^{(n)})\right]\right] \;=\; \mathcal{K}_n \tag{2}$$

When the set $\mathcal{K}_n$ fulfills the above property of containing the indices of the states with concentrated posterior mass, Equation (1) is a good approximation of the full posterior.

To summarize, in one iteration of this approximate EM algorithm, the following occurs: prior to the E-step, the algorithm is given a latent variable subset $\mathcal{K}_n$ computed by the selection function $\mathcal{S}_h$ to use in the E-step. Based on this, the E-step computes and returns the approximate posterior distribution $q(\mathbf{s})$. This result is then used to calculate the model parameter update equations in the M-step.

3 Gaussian process selection

We wish to generalize this approximate EM algorithm for inference in graphical models: instead of predefining the form of $\mathcal{S}_h$ for variable selection, we want to learn it in a black-box and model-free way based on the data. The case we consider is to learn $\mathcal{S}_h$ by using Gaussian process (GP) regression (e.g. (Rasmussen & Williams, 2005)), which is flexible, can make exact predictions, and scales with the number of data points N. Thus, with the previously defined concepts, we can express the Gaussian process model for the posterior marginal probability that $s_h^{(n)} = 1$:

$$f_h(\mathbf{y}^{(n)}) \sim \mathrm{GP}\left(0,\, k(\cdot,\cdot)\right), \;\; p_h^{(n)} \sim f(\mathbf{y}^{(n)}) + \epsilon^{(n)}, \tag{3}$$

$$\text{and} \quad \epsilon^{(n)} \sim \mathcal{N}(0, \sigma^2_{noise}) \tag{4}$$

where p_h represents the posterior marginal probability that $s_h^{(n)} = 1$, $k(\cdot,\cdot)$ is the covariance kernel, which can flexibly govern the representation of the relationship between variables, and σ^2_{noise} is the degree of Gaussian noise on the function f_h.

Thus, we use a GP to regress $\mathbf{S}$ onto $\mathbf{Y}$. In order to train the GP, we define our training data as the output of the previous iteration of the EM parameter inference algorithm, $q(\mathbf{S})$, or rather, the expectations of the latent variables, and the set of observed variables $\mathbf{Y}$, namely $\mathcal{D} = \{(\mathbf{y}^{(n)}, \mathbf{s}^{(n)}_{\text{prev-it}}) | n = 1, \ldots, N\}$. This means that in the very first EM iteration contains no actual information from the posterior distribution, and in this iteration the GP receives training data that is simply the initialization of the latent variables and the observed data.

Leave-one-out (LOO) prediction for GP regression can be shown to be:

$$\mu^{(n)} \;=\; \hat{\mathbf{s}}^{(n)} \;=\; \frac{\mathbf{s}^{(n)} - [K^{-1}\mathbf{s}^{(n)}]}{diag[K^{-1}]} \tag{5}$$

which computes the predictions of the GPs for the entire training set by only computing one kernel matrix inversion (see e.g. Chapter 5 (Rasmussen & Williams, 2005)). Namely, it computes a prediction for each $s_h^{(n)}$ given all variables $\mathbf{Y}$, where each predicted $\hat{s}_h$ represents the GP's characterization of the relationship between the two. Using LOO prediction leads to the following expression of the *GP selection function* for the nth datapoint:

$$\mathcal{S}(\mathbf{y}^{(n)}) \;=\; \mathcal{I}\left[\gamma\left[\hat{\mathbf{s}}^{(n)}\right]\right], \tag{6}$$

which specializes Equation (2).

Optimization of the model parameters in EM continues as described earlier: the M-step computes the model parameters based on these subsets of $\mathcal{K}_n$ to approximate the posterior distribution shown in Equation (1).

4 Experiments

We now apply our GP-selection inference approach to five different probabilistic models.

4.1 Sparse coding models

Using hand-crafted functions to preselect latent variables, a variety of sparse coding models have been successfully scaled to high-dimensional data with use of the selection (Henniges *et al.* , 2010; Bornschein *et al.* , 2013; Sheikh *et al.* , 2014) and select-and-sample (Shelton *et al.* , 2011, 2012) inference approaches. In this set of experiments, we consider three sparse generative models and do inference with our GP-selection approach instead of a hand-crafted selection function:

1. *Binary sparse coding*:

$$\text{latents:} \quad \mathbf{s} \sim Bern(\mathbf{s}|\pi) = \textstyle\prod_{h=1}^{H} \pi^{s_h} \left(1-\pi\right)^{1-s_h},$$
$$\text{observations:} \quad \mathbf{y} \sim \mathcal{N}(\mathbf{y}; W\mathbf{s}, \sigma^2 I),$$

   where $W \in \mathrm{R}^{D\times H}$ denotes the dictionary elements and π parameterizes the sparsity (See e.g. (Henniges *et al.* , 2010; Shelton *et al.* , 2011)).

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

$$\text{latents:} \quad \mathbf{s} = \mathbf{b} \odot \mathbf{z} \sim Bern(\mathbf{b}|\pi) \odot \mathcal{N}(\mathbf{z};\, \mu, \Sigma_h)$$
$$\text{observations:} \quad \mathbf{y} \sim \mathcal{N}(\mathbf{y}; W\mathbf{s}, \sigma^2 I)$$

   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. (Titsias & Lázaro-Gredilla, 2011; Goodfellow *et al.* , 2013; Sheikh *et al.* , 2014)).

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

$$\text{latents:} \quad \mathbf{s} = \mathbf{b} \odot \mathbf{z} \sim Bern(\mathbf{b}|\pi) \odot \mathcal{N}(\mathbf{z};\, \mu, \Sigma_h),$$
$$\text{observations:} \quad \mathbf{y} \sim \mathcal{N}(\mathbf{y}; \max_h\{s_h W_{dh}\}, \sigma^2 I)$$

   for which the mean of the Gaussian for each $\mathbf{y}^{(n)}$ is centered at $\max_h\{s_h W_{dh}\}$, where $\max_h$ is a nonlinearity that considers all H latent components and takes the h yielding the maximum value for $s_h W_{dh}$ ((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, iference with the truncated posterior in Equation (1) and selection functions designed by hand has yielded results as good as or more robust than exact inference. For models (1) and (3), the hand-crafted selection function used successfully was the cosine similarity between the weights (e.g. dictionary elements, components, etc.) $\mathbf{W}_h$ associated with each latent variable s_h and each data point $\mathbf{y}^{(n)}$: $\mathcal{S}_h(\mathbf{y}^{(n)}) = (\mathbf{W}_h^{\mathrm{T}} / ||\mathbf{W}_h||)\, \mathbf{y}^{(n)}$, with $||\mathbf{W}_h|| = \sqrt{\sum_{d=1}^{D} (W_{dh})^2}$. For model (2), the hand-crafted function was the data likelihood given a singleton state $\mathbf{s}_h$, not taking into account the probability of the state itself (i.e., $p(\mathbf{s}_h \,|\, \Theta)$). We will train GP regression for our flexible selection approach, where, for all models, the targets of the training data is the expected values of the latents.

We generate $N = 2,000$ datapoints consisting of $D = 5 \times 5 = 25$ observed dimensions and $H = 10$ latent components according to each of the models (1-3): N images of randomly selected overlapping 'bars' with varying intensities for models (2) and (3) and additive Gaussian noise parameterized by ground-truth $\sigma^2 = 2$. On average, each data point contains 2 bars.

For each of these models, we run 10 repetitions the following set of experiments: (1) selection using the respective hand-crafted selection function, (2) GP-selection using a linear covariance kernel, (3) GP-selection using an RBF covariance kernel, (4) GP-selection using a composition kernel of RBF, linear, bias and white noise kernels. The latter has been shown to be particularly adaptive to data, i.e. generally when there is no linear trend in the data, the variance of that kernel would go to zero. See the book by Rasmussen (Rasmussen & Williams, 2005) for a discussion on kernel selection. Kernel parameters were model-selected via maximum marginal likelihood every 10 EM iterations. As inference with the hand-crafted selection functions has already been shown to extract ground-truth parameters as robustly or more so than exact inference, we do not explicitly compare with this case. Number of selected latent variables was set to $H' = 5$. We run these experiments for 30 EM iterations for models (1) and (3) and for 60 iterations for model (2).

Results are shown in Figure 1. In all experiments, the GP-selection approach could infer ground-truth parameters as well as the hand-crafted function. For models where the cosine similarity was used (in (1) and (3)), 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-selection function explicit weights W as necessary in the hand-crafted function, GP regression 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 (2)), only GP regression with an RBF kernel was able to recover ground-truth parameters. In this case, (model (2)) 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. The other models, algorithm using GP-select converged to the ground-truth parameters with the same average frequency as the hand-crafted functions.

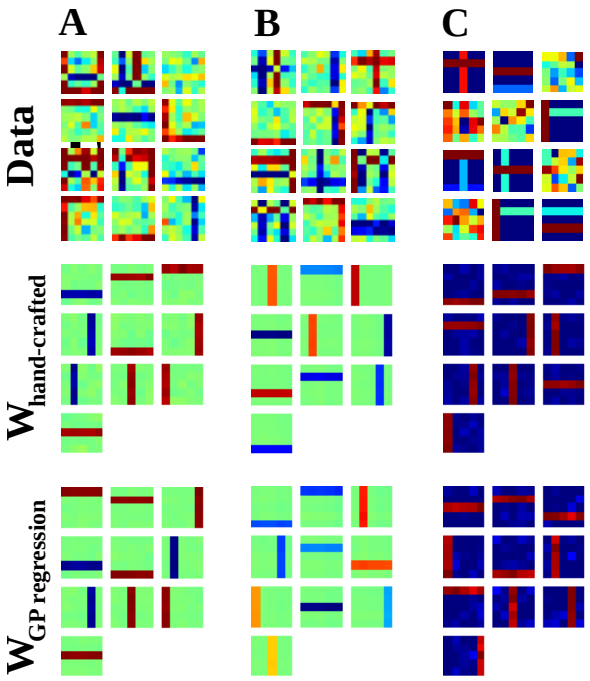

Figure 1: Sparse coding models results comparing GP-select with a successful hand-crafted selection function. Results are shown on artificial ground-truth data with $H = 10$ latent variables and $H' = 5$ preselected variables for: **A** Binary sparse coding, **B** Spike-and-slab sparse coding, and **C** Nonlinear spike-and-slab sparse coding. Top row: Example data points $\mathbf{y}^{(n)}$ generated by each of the models. Middle row: Converged dictionary elements W learned by the hand-crafted selection functions. Bottom row: Converged dictionary elements W learned by the GP-select with $H' = 5$ using the kernel with best performance (matching that of inference with hand-crafted selection function).

Furthermore, as the composition kernel is flexible enough to adapt to both linear and nonlinear relationships, this supports it as a reasonable general choice.

4.2 Gaussian mixture model

Next, we apply GP-selection to a simple example, a Gaussian mixture model, where the flexibility of the approach can be easily and intuitively visualized. The model of the observations is:

$$p(\mathbf{y}^{(n)}|\mu_c, \sigma_c, \pi) = \sum_c^C \mathcal{N}(\mu_c, \sigma_c)\, \pi_c \tag{7}$$

where C is the number of mixture components, the task is assign each data point to a cluster.

The targets of the training data used for GP regression is cluster responsibilities for all data points, for $\mathcal{D} = \{(y^{(n)}, \mathbf{r}^{(n)})|n = 1, \ldots, N\}$. With this, we apply our GP-selection approach to this model, computing the selection function according to Equation (6) and following the approximate EM approach as in the previous experiments. In these experiments we consider three scenarios for EM learning of the data in Equation (7): GP-selection with an RBF covariance kernel and a linear covariance kernel.

For visualization purposes, we generate 2-dimensional observed data ($\mathbf{y}^{(n)} \in \mathrm{R}^{D=2}$) from $C = 3$ clusters – first with randomly assigmed cluster means, and second such that the means of the clusters lie roughly on a line. In the GP-selection experiments, we select $C' = 2$ clusters to select from the full set (where C' clusters is analagous to H' latent variables in the non-mixture model setting), and run 40 EM iterations of the three experiments. We randomly initialize the variance of the clusters σ_c and initalize the cluster means μ_c at randomly selected data points. Results are shown in Figure 2.

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 2**B**), but the nonlinear approach successfully and easily finds the ground-truth clusters (Figure 2**A**). Furthermore, even when both were initialized in the optimal solution, the cluster assignments from GP-selection with a linear kernel quickly wandered away from the optimal solution and was identical to random initialization, converging to the same result shown in iteration 20 of Figure 2**B**) The RBF kernel cluster assignments remained in the optimal solution even with number of preselected clusters set to $C' = 1$.

These experiments demonstrate that, not only is it difficult to know the relationship of the inputs and the output variables a priori, but the structure of the data strongly plays a role in the success of using selection in inference. Given one likely cannot know a priori the structure of the (potentially high-dimensional) data, these results suggest using a flexible kernel that can handle diverse data.

4.3 Translation Invariant Occlusive models

Now that we have verified GP-select can be applied to diverse graphical models and converge to ground-truth parameters, we consider a more challenging model that addresses a problem in the computer vision domain.

Here we consider models that cope with translations of objects in a scene. Such models are particularly difficult to optimize because they must consider multiple objects and locations in a scene – the complexity of latent space becomes the number of locations exponentiated by the number of objects. Inference in

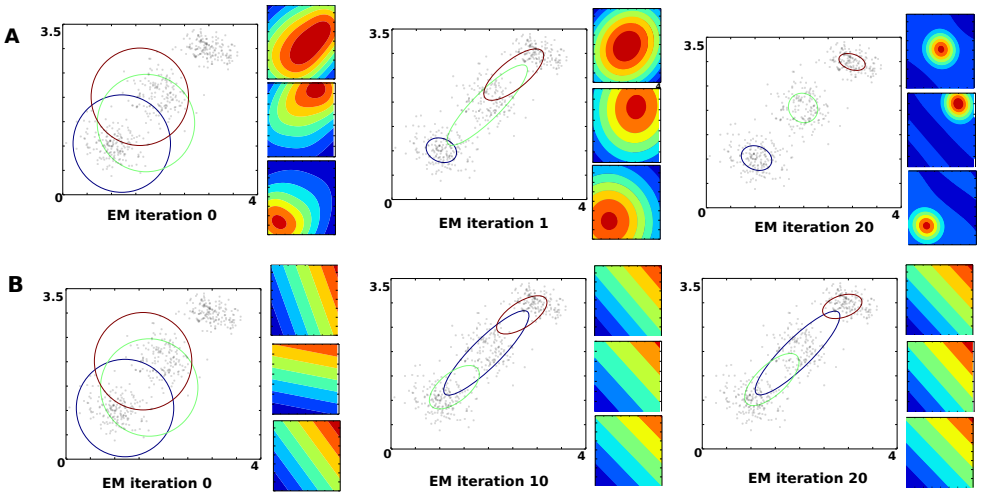

Figure 2: Gaussian mixture model results using GP-selection ($C' = 2$) for inference. Shown is the development of inference using **A** RBF covariance kernel in the regression, and **B** a linear covariance kernel. For each iteration shown, we see (1) the observed data and their inferred cluster assignments and (2) the C corresponding GP regression functions learned/used for GP-selection in that iteration. Different iterations are pictured due to different convergance rates. As shown, inference with GP-selection using a linear kernel is unable to assign the data points to the appropriate clusters, whereas GP-selection with an RBF kernel succeeds.

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 pre-selection functions that consider translational invariance have been successfully applied to this type of model (Dai & Lücke, 2012b, 2014; Dai *et al.* , 2013). In such a model, the construction of a subset of latent space mainly consists of two steps: first, the candidate locations of all the objects in the model are predicted, and then a subset of candidate objects that might appear in the image are selected according to the predicted locations. Next, the reduced subspace $\mathcal{K}_n$ is constructed according to the combinatorics of the different locations of all the candidate objects. The posterior distribution is then computed following Equation (1).

One limitation of such a hand-engineered system is that the selection function itself has parameters which need to be hand-tuned, e.g., the number of representative features.Second, this selection function needs to scan through the entire image at all possible locations, which is very costly for large-scale experiments. In translation invariant models, instead of predicting the existence of a component, the selection function has to predict all possible locations a component could be. To maximally exploit the capabilies of 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. Therefore, the input to 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, for learning 10 components in a $D = 30 \times 30$ pixel image patche, the output dimensionality of GP-select is 9000. Although this initially seems a challenging computation for GP-selection, this is not the case given GP models scale linearly with output dimensionality. As we will show in the following experiment, inference with the GP-selection function performs significantly faster than the original selection function used in previous works, because it does not need to explicitly scan through image.

Intuitively, using GP-select for this problem makes sense, as objects are typically not uniformly distributed in a scene and with limited observations we often can give a good rough prediction of object locations. Approaches such as data-driven location are based entirely on this idea, going further to convert a detection problem into an information retreival problem, and recently achieving the state-of-art performance on object detection (Rodríguez-Serrano & Larlus, 2013).

COIL Data. We apply our GP-selection function to the Invariant Exclusive Component Analysis (InvECA) model (Dai *et al.* , 2013). We consider an image dataset that is heavily used in the computer vision literature: data were generated using objects from the COIL-100 image dataset (Nene *et al.* , 1996), taking 16 different objects, downscaled to $D = 10 \times 10$ pixels and segmented out from the black background.

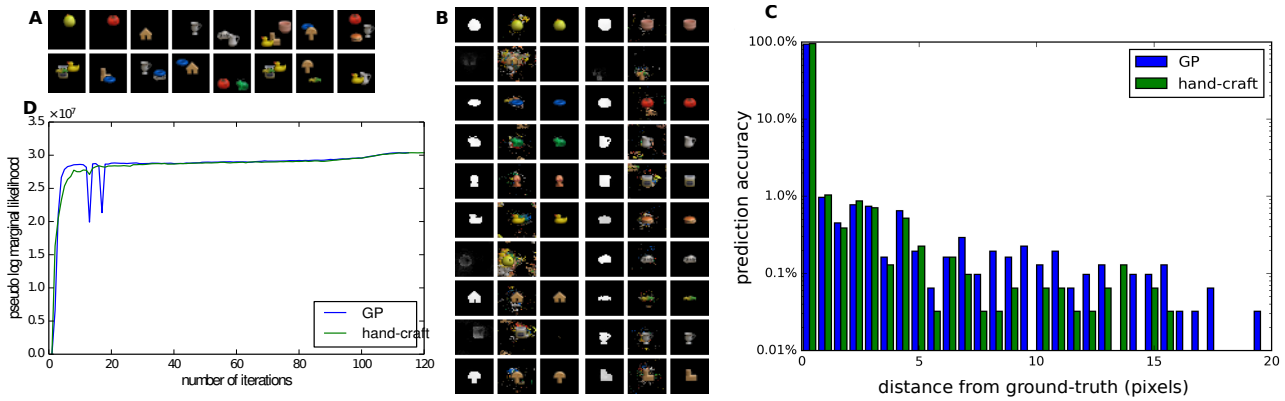

Figure 3: Results of GP-select on the Translation Invariant model. **A** shows some example from the dataset. **B** shows the learned components and their masks. The first column shows the mask of each component, the second column shows the learned feature representations, and the third column only shows the area of the learned components with mask higher than 0.5. **C** shows the log-scale histogram of the prediction accuracy for both selection functions in terms of the distance from the predicted location to the ground truth location: the x-axis shows the distance from the ground-truth location in terms of pixels, and the y-axis shows the percentages. **D** shows the convergence of the pseudo log-likelihood [of the model parameters learned at each EM iteration] for the both selection functions over all EM iterations. Figures **C** and **D** demonstrate the matching inference performance of both selection functions.

A given image was generated by randomly selecting a subset of the 16 objects, where each object has a probability of 0.2 to appear. 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 dataset (examples shown in Figure 3**A**).

For this experiment, the GP model parameters are optimized at the end of each EM iteration, using a maximum of 20 gradient updates with L-BFGS. All images in the training set are used and the marginal truncated posterior probabilties are denoted $p(l_h|\mathbf{y}^{(n)})$, where l_h is the location of the h component. The number of objects sought to be learned is $H = 20$ and preselected objects was $H' = 5$.

Results are shown in Figure 3. GP-select successfully recovers all objects in the dataset, all of which are shown with their masks in Figure 3**B**. Four additional components developed into representations, all of which have very low mask values, which can easily tell from other true components. In order to quantify the accuracy of our GP-selection function, we collected the GP-selection after learning, and applied the GP selection function to the whole dataset again, then compared with the ground-truth location of all the objects in the datasets. Then, the accuracy of the predicted locations are estimated by computing the distance from the location of the top candidate and the ground-truth location. These distances are plotted as a histogram in Figure 3**C**. As a comparison, we also estimated the prediction accuracy of the orginal hand-craft selection function. Both selection functions can very accurately predict the locations of all the objects in the dataset. Furthermore, the pseudo-loglikelihood for both selection functions are shown in Figure 3**D**.

Because GP selection avoids explicitly scanning through the whole image, the time of inference is significantly faster than the original selection function. For the GP-selection function, which was applied without any approximations or tricks to enhance speed, inferring the whole dataset (2000 images) required 22.1 seconds on a single CPU core, while the inference with the original selection function required 1830.9 seconds on a single CPU core. In the orginal work, the hand-crafted selection function was implemented with GPU acceleration and parallelization.

5 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 previ-

ous regression functions. In experiments on simple and easily interpretable models (spike-and-slab, Gaussian mixtures), the learned selection functions behaved in accordance with our expectations for the posterior distribution over the latents. For the complex hierarchical model of (Dai & Lücke, 2014), we obtain equivalent performance in inference at a much lower computational cost.

A major area where further performance gains might be expected is in improving computational performance, since we expect the greatest advantages of GP selection 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 development regarding the scaling up GP inference to large-scale problems, e.g., sparse GP approximation (Lawrence *et al.* , 2003), stochastic variational inference (Hensman *et al.* , 2013, 2012), using parallelization techniques and GPU acceleration (Dai *et al.* , 2014).

Acknowledgements

We acknowledge funding by the German Research Foundation (DFG) under grant LU 1196/4-2 (JS), by the Cluster of Excellence EXC 1077/1 "Hearing4all" (JL), and by the RADIANT and WYSIWYD (EU FP7-ICT) projects (ZD).

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.

Dai, Z., & Lücke, J. 2012a. Autonomous Cleaning of Corrupted Scanned Documents – A Generative Modeling Approach. *Pages 3338–3345 of: IEEE Conference on Computer Vision and Pattern Recognition*.

Dai, Z., & Lücke, J. 2012b. Unsupervised Learning of Translation Invariant Occlusive Components. *Pages 2400–2407 of: IEEE Conference on Computer Vision and Pattern Recognition*.

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.

Dai, Z., Exarchakis, G., & Lücke, J. 2013. What Are the Invariant Occlusive Components of Image Patches? A Probabilistic Generative Approach. *Pages 243–251 of: Advances in Neural Information Processing Systems*.

Dai, Zhenwen, Damianou, Andreas, Hensman, James, & Lawrence, Neil. 2014. *Gaussian Process Models with Parallelization and GPU acceleration*.

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.

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.

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

Hensman, James, Rattray, Magnus, & Lawrence, Neil D. 2012. Fast Variational Inference in the Conjugate Exponential Family. *In: Advances in Neural Information Processing System*.

Hensman, James, Fusi, N, & Lawrence, ND. 2013. Gaussian processes for big data. *In: Uncertainty in Artificial Intelligence*.

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, Diederik P., & Welling, Max. 2014. Efficient Gradient-Based Inference through Transformations between Bayes Nets and Neural Nets. *In: International Conference on Machine Learning*.

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, Neil, Seeger, Matthias, & Herbrich, Ralf. 2003. Fast Sparse Gaussian Process Methods: The Informative Vector Machine. *Pages 625–632 of: Advances in Neural Information Processing Systems 15*. MIT Press.

Le, Q.V., Sarlos, T., & Smola, A. J. 2013. Fastfood — Computing Hilbert Space Expansions in Loglinear Time. *In: International Conference on Machine Learning*.

Lücke, J., & Eggert, J. 2010. Expectation Truncation And the Benefits of Preselection in Training Generative Models. *Journal of Machine Learning Research*, **11**, 2855–900.

Lücke, J., & Sahani, M. 2008. Maximal Causes for Non-linear Component Extraction. *Journal of Machine Learning Research*, **9**, 1227–67.

Mnih, Andriy, & Gregor, Karol. 2014. Neural Variational Inference and Learning in Belief Networks. *In: Proceedings of the 31st International Conference on Machine Learning*.

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.

Nene, S. A., Nayar, S. K., & Murase, H. 1996. *Columbia Object Image Library (COIL-100)*. Tech. rept. CUCS-006-96.

Rasmussen, Carl Edward, & Williams, Christopher K. I. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Rodríguez-Serrano, José A., & Larlus, Diane. 2013. Predicting an Object Location Using a Global Image Representation. *Pages 1729–1736 of: International Conference on Computer Vision*.

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. *Advances in Neural Information Processing Systems*, **24**.

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. *Advances in Neural Information Processing Systems*, **25**.

Titsias, M., & Lázaro-Gredilla, M. 2011. Spike and Slab Variational Inference for Multi-Task and Multiple Kernel Learning. *In: Advances in Neural Information Processing Systems*.

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

Zhang, Yuchen, Duchi, John C., & Wainwright, Martin J. 2014. *Divide and Conquer Kernel Ridge Regression: A Distributed Algorithm with Minimax Optimal Rates*. Tech. rept. University of California, Berkeley. (`http://arxiv.org/abs/1305.5029`).