# Semiparametric Latent Factor Models

**Yee Whye Teh**
Computer Science Div.
University of California
Berkeley, CA 94720-1776
*ywteh@eecs.berkeley.edu*

**Matthias Seeger**
Computer Science Div.
University of California
Berkeley, CA 94720-1776
*mseeger@eecs.berkeley.edu*

**Michael I. Jordan**
Computer Science Div. and Dept. of Statistics
University of California
Berkeley, CA 94720-1776
*jordan@eecs.berkeley.edu*

## Abstract

We propose a semiparametric model for regression problems involving multiple response variables. The model makes use of a set of Gaussian processes that are linearly mixed to capture dependencies that may exist among the response variables. We propose an efficient approximate inference scheme for this semiparametric model whose complexity is linear in the number of training data points. We present experimental results in the domain of multi-joint robot arm dynamics.

## 1 Introduction

We are interested in supervised problems involving multiple responses that we would like to model as conditionally dependent. In statistical terminology, we would like to "share statistical strength" between multiple response variables; in machine learning parlance this is often referred to as "transfer of learning." As we demonstrate empirically, such sharing can be especially powerful if the data for the responses is partially missing.

In this paper we focus on multivariate regression problems.[1] Models related to the one proposed here are used in geostatistics and spatial prediction under the name of *co-kriging* [3], and an example from this domain helps to give an idea of what we want to achieve with our technique. After an accidental uranium spill, a spatial map of uranium concentration is sought covering a limited area. We can take soil samples at locations of choice and measure their uranium content, and then use Gaussian process regression or another spatial prediction technique to infer a map. However, it is known that these carbon concentration and uranium concentration are often significantly correlated, and carbon concentration is easier to measure, allowing for more dense measurements. Thus in co-kriging the aim is to set up a joint spatial model for several responses with the aim of improving the prediction of one of them. The model to be described in the current paper goes beyond simple co-kriging methods in several ways. First, rather than combining responses in a posthoc manner, our model uses latent random processes to represent *conditional* dependencies between responses directly. The latent processes are fitted using the data from all responses and can be used to model chacteristics of the dependencies beyond those based solely on marginal relationships. Second, the nature of the dependencies does not have to be known in advance but is learned from training data using empirical Bayesian techniques.

Another example of a motivating application arises in computer vision, where it is of interest to estimate the pose of a human figure from image data. In this case the response variables are the joint angles of the human body [1]. It is well known that human poses are highly constrained, and it would be useful for a pose estimation algorithm to take into account these strong dependencies among the joint angles.

Historically, the problem of capturing commonalities among multiple responses was one of the motivations behind multi-layer neural networks—the "hidden units" of a neural network were envisaged not only as nonlinear transformations, but also as adaptive basis functions to be "shared" in predicting multivariate responses. As neural networks gave way to kernel machines for classification and regression, with comcomitant improvements in flexibility, analytical tractability and performance, this core ability of neural networks was largely lost.

To elaborate on this point, note that there have been two main paths from neural networks to kernel machines. The first path, due to [10], involved the observation that in a particular limit the probability associated with (a Bayesian interpretation of) a neural network approaches a Gaussian process. For some purposes, it is arguably advantageous to work directly with the Gaussian process via its covariance function. However, in this limit it also turns out that the components of the response (the output

[1] In Section 5 we indicate how our technique can be extended to other settings such as multi-label classification.

vector) are independent—the ability to model couplings among these components is lost. The second path to kernel machines, via the optimization of margins [14], simplified the problem of fitting one-dimensional responses, but largely neglected the problem of fitting dependent multivariate responses. This problem has returned to the research agenda via architectures such as the conditional random field which links response variables using the graphical model formalism [5, 13].

Our approach to modeling dependencies among response variables heads in a direction that is more nonparametric than the CRF. In the spirit of factor analysis, we view the relationships among $C$ components of a response vector $\boldsymbol{y}$ as reflecting a linear (or generalized linear) mixing of $P$ underlying latent variables. These latent variables are indexed by a covariate vector $\boldsymbol{x}$, and thus we have a set of indexed collections of variables; that is, a set of stochastic processes. Specifically, we assume that each of the $P$ variables is conditionally independently distributed according to a Gaussian process, with $\boldsymbol{x}$ as the (common) index set. The mean of the response $\boldsymbol{y}$ is then a (possibly nonlinear) function of a linear combination of these conditionally independent Gaussian processes.

This model is a semiparametric model, as it combines a nonparametric component (several Gaussian processes) with a parametric component (the linear mixing). We refer to the model as a *semiparametric latent factor model* (SLFM). Note that factor analysis is a special case of the SLFM, arising when $\boldsymbol{x}$ is a constant. Note also that Neal's limiting Gaussian process is a special case, arising when $P = 1$. Finally, as we discuss in Section 2, when $C = 1$ and $P > 1$ the SLFM can be viewed as a Gaussian process version of the multiple kernel learning architecture proposed in [6].

As in the case of simpler Gaussian process models, a significant part of the challenge of working with the SLFM is computational. This challenge can be largely met by exploiting recent developments in the literature on fitting large-scale Gaussian process regression and classification models. In particular, we make use of the informative vector machine (IVM) framework for Gaussian processes [7, 11]. In this framework, only a subset of "informative" likelihood terms are included in the computation of the posterior, yielding an training algorithm which scales linearly in the number of training data points. Moreover, the Bayesian underpinnings of the IVM yields general methods for setting free parameters (hyperparameters), an important capability which is not always easily achieved within the context of other kernel-based methodologies.

## 2 Semiparametric Latent Factor Models

In this section we give a description of our model for nonparametric regression with multiple responses. We begin with a short overview of Gaussian process (GP) regression in the simpler setting of single responses.

A GP can be viewed as a prior over random real-valued functions $u : \mathcal{X} \mapsto \mathbb{R}$, and is parametrized by a mean function $\mu(\cdot)$ and a covariance kernel $k(\cdot, \cdot)$. A random function $u(\cdot)$ is said to be distributed according to a GP if for any finite subset $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\} \subset \mathcal{X}$ of covariate vectors the random vector $u(X) = (u(\boldsymbol{x}_1), \ldots, u(\boldsymbol{x}_m))^T \in \mathbb{R}^m$ is distributed according to a Gaussian with mean $(\mu(\boldsymbol{x}_1), \ldots, \mu(\boldsymbol{x}_m))^T$ and covariance $\boldsymbol{K} \in \mathbb{R}^{m,m}$ where the $i, j^{\text{th}}$ entry is $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. In our work we use GPs with zero mean: $\mu(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in \mathcal{X}$. The covariance kernel $k(\cdot, \cdot)$ has to satisfy a symmetric positive-definiteness (SPD) property; that is, $\boldsymbol{K}$ should be SPD for every finite subset $X$. Thus, a GP is simply a consistent way of assigning multivariate Gaussian distributions to $u(X)$ for any finite $X$.

GPs have traditionally been used for Bayesian classification and regression with a single response, where the problem is treated as that of estimating a random univariate function from the covariate space to the response space. Rather than assuming a parametric form for the random function, the nonparametric Bayesian approach places a GP prior over the space of all functions, and infers the posterior over functions given the training data.

Returning to our multiple response setting, let $\mathcal{X}$ be the covariate (input) space and let $\mathcal{Y} = \mathbb{R}^C$ be the response (output) space. We are interested in predicting $\boldsymbol{y} = (y_1, \ldots, y_C)^T \in \mathcal{Y}$ from $\boldsymbol{x} \in \mathcal{X}$; i.e., in estimating $P(\boldsymbol{y}|\boldsymbol{x})$. We model the conditional distribution using latent variables $\boldsymbol{v} \in \mathbb{R}^C$ such that the components $y_c$ are mutually independent and independent of $\boldsymbol{x}$ given $\boldsymbol{v}$:

$$P(\boldsymbol{y}|\boldsymbol{v}, \boldsymbol{x}) = \prod_{c=1}^{C} P(y_c|v_c).$$

The conditional distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$ is then:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \int P(\boldsymbol{v}|\boldsymbol{x}) \prod_{c=1}^{C} P(y_c|v_c)\, d\boldsymbol{v}.$$

In this paper we focus on regression with Gaussian errors; i.e., $P(y_c|v_c) = N(y_c|v_c, \sigma_c^2)$. We treat $P(\boldsymbol{v}|\boldsymbol{x})$ nonparametrically, in particular using GPs. We do this by introducing a further set of latent variables $\boldsymbol{u} \in \mathbb{R}^P$ and letting $\boldsymbol{v}$ be linearly related to $\boldsymbol{u}$:

$$\boldsymbol{v} = \boldsymbol{\Phi}\boldsymbol{u}, \tag{1}$$

with $\boldsymbol{\Phi} \in \mathbb{R}^{C,P}$. Finally we assume that the coordinates of $\boldsymbol{u}$ have independent GP priors conditional on $\boldsymbol{x}$; i.e., there are random functions $u_p : \mathcal{X} \mapsto \mathbb{R}$ such that $u_p = u_p(\boldsymbol{x})$, and $u_p(\cdot)$ are distributed according to a GP with zero mean and covariance kernel $k_p(\cdot, \cdot)$. This setup allows

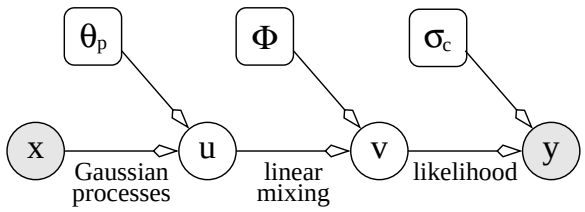


Figure 1: A semiparametric latent factor model.

for conditional dependencies among the coordinates of $\boldsymbol{v}$ to be expressed via $\boldsymbol{\Phi}$ and the latent $\boldsymbol{u}$ variables.

The form assumed in Eq. 1 is in direct analogy with factor analysis models. Note that the information learned from each single response variable $y_c$ is reflected in the posterior for the latent GPs $\boldsymbol{u}(\cdot)$. Thus, there is sharing of statistical strength across response variables.

We call the model a *semiparametric latent factor model (SLFM)*; the graphical model is shown in Figure 1. The parameters are the components of $\boldsymbol{\Phi}$ and the hyperparameters (aka, the nuisance parameters) are the kernel parameters $\boldsymbol{\theta}_p$ and the variance parameters $\sigma_c^2$ associated with the Gaussian likelihoods $P(y_c|v_c)$.

Note that each coordinate $v_c(\cdot)$ of $\boldsymbol{v}(\cdot) = \boldsymbol{\Phi}\boldsymbol{u}(\cdot)$ is a priori a GP with covariance kernel given by $\sum_{p=1}^{P} \phi_{c,p}^2 k_p(\cdot,\cdot)$, where $\phi_{c,p}$ is the $(c,p)^{\text{th}}$ element of the matrix $\boldsymbol{\Phi}$. Hence one interpretation of our model is that each response variable is modeled as a Gaussian process with a kernel that is an adaptive, conic combination of base kernels. In fact, if we have $C = 1$ and $P > 1$, then this model can be viewed as a Gaussian process version of the kernel learning proposed in [6]. However our model does not just fit the kernel, it actually makes use of the same latent Gaussian processes for every response variable, allowing more expressive sharing of information across the response variates. Also, in the case $P < C$ which is our focus in the current paper, it is more efficient to represent $\boldsymbol{u}$ explicitly than to integrate it out.

## 3 Inference

Given a model and training samples $D = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$ drawn i.i.d. from the model[2], we perform Bayesian inference for the latent variables and estimate the parameters and the hyperparameters within an empirical Bayes framework. We begin by introducing the relevant latent variables.

Let $u_{i,p} = u_p(\boldsymbol{x}_i)$ and $v_{i,c} = v_c(\boldsymbol{x}_i)$. We collect these into vectors $\boldsymbol{u} = (u_{i,p})_{i,p}$ and $\boldsymbol{v} = (v_{i,c})_{i,c}$ where the double indices are flattened, with index $i$ running over $1, \ldots, n$ first, e.g., $\boldsymbol{u} = (u_{1,1}, u_{2,1}, \ldots, u_{n,1}, \ldots, u_{n,P})^T$. The vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are again linearly related: $\boldsymbol{v} = (\boldsymbol{\Phi} \otimes \boldsymbol{I})\boldsymbol{u}$, where $\otimes$ is the Kronecker product. In the following we will assume that $P \leq C$ and $\boldsymbol{\Phi}$ has full rank so the pseudo-inverse $\boldsymbol{\Phi}^\dagger$ such that $\boldsymbol{\Phi}^\dagger\boldsymbol{\Phi} = \boldsymbol{I}$ exists and $\boldsymbol{u} = (\boldsymbol{\Phi}^\dagger \otimes \boldsymbol{I})\boldsymbol{v}$. The case $P > C$ requires a different treatment and will be presented in future work. The variable $\boldsymbol{u}$ is distributed a priori according to a Gaussian with zero mean and block diagonal covariance matrix $\boldsymbol{K} = \operatorname{diag}(\boldsymbol{K}^{(p)})_p$, where the $p^{\text{th}}$ block has $i, j^{\text{th}}$ entry given by $\boldsymbol{K}^{(p)}_{i,j} = k_p(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The covariance of $\boldsymbol{v}$ is thus $\tilde{\boldsymbol{K}} = (\boldsymbol{\Phi} \otimes \boldsymbol{I})\boldsymbol{K}(\boldsymbol{\Phi}^T \otimes \boldsymbol{I})$.

The posterior processes $\boldsymbol{u}(\cdot) \mid D$ are Gaussian in the case of Gaussian likelihoods $P(y_c|v_c)$, and in principle we could compute their mean and covariance functions explicitly. However, this is prohibitively expensive for all but fairly small values of $n$ and $C$ (the procedure scales as $O(n^3\, C^3)$ in general). We thus make use of the *informative vector machine (IVM)* framework [7] which computes a sparse approximation to the full Gaussian posterior $P(\boldsymbol{v}|D)$ by means of greedy forward selection of an *active subset* of the training sample using information-theoretic criteria. The difference here is that $P > 1$ processes have to be represented along with their dependencies, and the empirical Bayes maximization has to encompass a large number of non-kernel parameters, namely the elements of $\boldsymbol{\Phi}$.

### 3.1 Forward selection

The active set $I$ of size $d$ consists of tuples $(i,c) \in \{1, \ldots, n\} \times \{1, \ldots, C\}$. The goal is to select $I$ such that the approximate posterior

$$Q(\boldsymbol{v}) \propto P(\boldsymbol{v}) \prod_{(i,c)\in I} P(y_{i,c}|v_{i,c}) \tag{2}$$

is close to $P(\boldsymbol{v}|D)$. For given $I$, the posterior approximation $Q$ is given by simply ignoring all observations not in $I$. However, our method of selecting $I$ depends on the complete sample $D$, as is discussed in this section. The idea is to greedily select the candidate $(i,c)$ which changes the posterior most if we were to include it (i.e., incorporate its likelihood term into $Q$). A good way of measuring this change is the *information gain* studied in the setting of active learning (or sequential design). If $Q_{k-1}$ denotes the posterior approximation after $k-1$ inclusions, the criterion is

$$\begin{aligned}\Delta_{i,c}^{info} &= \mathrm{D}\left[Q_{i,c}(\boldsymbol{v}) \,\|\, Q_{k-1}(\boldsymbol{v})\right] \\ &= \mathrm{D}\left[Q_{i,c}(v_{i,c}) \,\|\, Q_{k-1}(v_{i,c})\right],\end{aligned}$$

where $Q_{i,c}$ is the approximate posterior we obtain if the term $(i,c)$ is included at iteration $k$. At each iteration we pick the $(i,c)$ that maximizes $\Delta_{i,c}^{info}$. Since $Q_{i,c}(v_{i,c}) \propto P(y_{i,c}|v_{i,c})Q_{k-1}(v_{i,c})$, we can compute $\Delta_{i,c}^{info}$ in $O(1)$ if

[2] We allow *incomplete* observations of $\boldsymbol{y}_i$. That is, entries of $\boldsymbol{y}_i$ are allowed to be unobserved—this simply involves dropping likelihood terms corresponding to the unobserved entries.

the current marginal $Q_{k-1}(v_{i,c})$ is known. The representation of $Q$ described in Section 3.2 makes it possible to maintain all these marginals at all times so that we can score all $(i,c) \notin I$ prior to each inclusion. After $d$ iterations we have an active set $(i_1,c_1),\ldots,(i_k,c_k)$ which determines the approximate posterior in Eq. 2.

### 3.2 Representing the approximate posterior

The representation for the approximate posterior in Eq. 2 has to satisfy the following properties in order for it to be useful: it should allow all marginals $Q(v_{i,c})$ to be maintained explicitly at all times, which allows for forward selection (see Section 3.1); it should have a small memory footprint; and it can be efficiently updated when a new likelihood term is included. In this section we describe how these properties are achieved.

Let $\boldsymbol{\Pi} = \mathrm{diag}(\pi_k)_k$ be a diagonal matrix and $\boldsymbol{b} = (b_k)_k$ be a vector which collects the parameters of the approximate posterior, in the sense that

$$Q(\boldsymbol{v}) \propto P(\boldsymbol{v}) \exp\left(-\frac{1}{2}\boldsymbol{v}_I^T \boldsymbol{\Pi} \boldsymbol{v}_I + \boldsymbol{b}^T \boldsymbol{v}_I\right),$$

where $\boldsymbol{v}_I = (v_{i_1,c_1},\ldots,v_{i_d,c_d})^T$. In the case of regression we have $\pi_k = \sigma_{c_k}^{-2}$ and $b_k = \sigma_{c_k}^{-2} y_{i_k,c_k}$. Note that $\boldsymbol{\Pi}$ and $\boldsymbol{b}$ are ordered according to the order in which likelihood terms are included. To convert from this ordering to the natural ordering of $\boldsymbol{u}$ and $\boldsymbol{v}$, define a selection matrix $\boldsymbol{P} \in \mathbb{R}^{d,nC}$ where $\boldsymbol{P}_{k,(i_k,c_k)} = 1$ for $k = 1,\ldots,d$, and zeros elsewhere (note that the natural ordering in which we flatten $i,c$ indices runs over $i$ first).

Given $\boldsymbol{\Pi}$ and $\boldsymbol{P}$, the covariance of $Q_k(\boldsymbol{v})$ is

$$\tilde{\boldsymbol{A}} = (\tilde{\boldsymbol{K}}^{\dagger} + \boldsymbol{P}^T \boldsymbol{\Pi} \boldsymbol{P})^{\dagger}.$$

Given our assumption $P \leq C$, we can represent the approximate posterior in terms of $\boldsymbol{u}$ to minimize memory and time requirements. As $\boldsymbol{u} = (\boldsymbol{\Phi}^{\dagger} \otimes \boldsymbol{I})\boldsymbol{v}$, the covariance of $Q_k(\boldsymbol{u})$ is

$$\boldsymbol{A} = (\boldsymbol{\Phi}^{\dagger} \otimes \boldsymbol{I})(\tilde{\boldsymbol{K}}^{\dagger} + \boldsymbol{P}^T \boldsymbol{\Pi} \boldsymbol{P})^{\dagger} (\boldsymbol{\Phi}^{\dagger T} \otimes \boldsymbol{I}).$$

Using the Sherman-Morrison-Woodbury formula, we obtain

$$\boldsymbol{A} = \boldsymbol{K} - \boldsymbol{M}\boldsymbol{M}^T, \quad \boldsymbol{M} = \boldsymbol{K}(\boldsymbol{\Phi}^T \otimes \boldsymbol{I})\boldsymbol{P}^T \boldsymbol{\Pi}^{1/2} \boldsymbol{L}^{-T}, \tag{3}$$

where $\boldsymbol{L}$ is the lower triangular Cholesky factor of

$$\boldsymbol{B} = \boldsymbol{I} + \boldsymbol{\Pi}^{1/2} \boldsymbol{P} \tilde{\boldsymbol{K}} \boldsymbol{P}^T \boldsymbol{\Pi}^{1/2}.$$

The mean of $Q_k(\boldsymbol{u})$ is obtained as

$$\boldsymbol{h} = \mathrm{E}_{Q_k}[\boldsymbol{u}] = \boldsymbol{M}\boldsymbol{\beta}, \quad \boldsymbol{\beta} = \boldsymbol{L}^{-1} \boldsymbol{\Pi}^{-1/2} \boldsymbol{b} \tag{4}$$

while the mean and variance of $v_{i,c}$ under $Q_k$ are

$$\tilde{h}_{i,c} = \boldsymbol{\phi}^{(c)} \boldsymbol{h}_i, \quad \tilde{a}_{i,c} = \boldsymbol{\phi}^{(c)} \boldsymbol{A}_i \boldsymbol{\phi}^{(c)T},$$

where $\boldsymbol{h}_i$ and $\boldsymbol{A}_i$ are the mean and covariance of $\boldsymbol{u}_i$, extracted from entries of Eq. 4 and Eq. 3 respectively, and $\boldsymbol{\phi}^{(c)}$ is the $c^{\text{th}}$ row of $\boldsymbol{\Phi}$. The forward selection can be carried out once $\tilde{h}_{i,c}$ and $\tilde{a}_{i,c}$ are computed for every $(i,c)$ not already included in the active set.

Finally, the representation of $Q(\boldsymbol{u})$ is given by $\boldsymbol{L} \in \mathbb{R}^{d,d}$, $\boldsymbol{\beta} \in \mathbb{R}^d$, $\boldsymbol{M} \in \mathbb{R}^{nP,d}$, and the mean $\boldsymbol{h}_i \in \mathbb{R}^P$ and covariance $\boldsymbol{A}_i \in \mathbb{R}^{P,P}$ of $\boldsymbol{u}_i$, for $i = 1,\ldots,n$. Since the rows of $\boldsymbol{L}$, $\boldsymbol{\beta}$, and $\boldsymbol{M}$ are already ordered by inclusion iteration, they can be updated efficiently and stably by appending new rows or columns. If at iteration $k$ we included likelihood term $(i,c)$, we compute

$$l = \sqrt{1 + \pi_k \tilde{a}_{i,c}}, \qquad \boldsymbol{l} = \pi_k^{1/2} \sum_p \phi_{c,p} \boldsymbol{M}^{(i,p)\ T}$$

$$\boldsymbol{\mu}_p = \frac{\phi_{c,p} \pi_k^{1/2} \boldsymbol{K}_i^{(p)} - \boldsymbol{M}^{(p)} \boldsymbol{l}}{l} \qquad \gamma = \frac{\pi_k^{-1/2} b_k - \boldsymbol{l}^T \boldsymbol{\beta}}{l},$$

where $\boldsymbol{M}^{(p)}$ is a matrix whose rows are $\boldsymbol{M}^{(i,p)}$. The new row of $\boldsymbol{L}$ is $(\boldsymbol{l}^T\ l)$, the new column of $\boldsymbol{M}^{(p)}$ is $\boldsymbol{\mu}_p$, and the new entry of $\boldsymbol{\beta}$ is $\gamma$. Let $\boldsymbol{\mu}_i = (\mu_{i,p})_p$. Then $\boldsymbol{h}_i$ is updated by adding $\gamma \boldsymbol{\mu}_i$ while $\boldsymbol{\mu}_i \boldsymbol{\mu}_i^T$ is subtracted from $\boldsymbol{A}_i$.

The memory requirement is dominated by $\boldsymbol{M}$ which is $O(nPd)$, while the $P$ matrix-vector multiplications involved in computing $\boldsymbol{\mu}_p$ dominate the update time complexity, which is $O(nPd)$. Computing the information gain scores $\Delta_{i,c}^{info}$ requires $O(nCP^2)$ time which is in general subdominant to $O(nPd)$. Note that all costs are linear in the number of training points $n$ which is the dominant factor in many large applications.

### 3.3 Parameter and hyperparameter estimation

We have described an effective procedure to compute an approximation to the posterior of the latent variables. Here we outline empirical Bayes estimation of the parameters and hyperparameters. Let $\boldsymbol{\alpha}$ denote a parameter or hyperparameter of interest, and let $\boldsymbol{s}$ denote the variational parameters which define the approximate posterior—these are the active set indicators, $\boldsymbol{\Pi}$ and $\boldsymbol{b}$. Since we cannot compute the marginal probability of $\boldsymbol{y}$ given $\boldsymbol{x}$ and $\boldsymbol{\alpha}$ exactly, we optimize a variational lower bound instead:

$$\log P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\alpha}) \geq \mathrm{E}_Q[\log P(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{\alpha})] - \mathrm{D}[Q(\boldsymbol{u}) \| P(\boldsymbol{u} \mid \boldsymbol{\alpha})], \tag{5}$$

where $Q(\boldsymbol{u})$ is the approximate posterior, given by Eq. 2. We use a double loop iterative procedure, where in the inner loop we optimize Eq. 5 with respect to $\boldsymbol{\alpha}$ using a quasi-Newton method while keeping $\boldsymbol{s}$ fixed, and in the outer

loop we reselect a new $\boldsymbol{s}$ greedily as detailed above. Notice that $Q(\cdot)$ is dependent on both $\boldsymbol{s}$ and $\boldsymbol{\alpha}$. For purposes of optimizing $\boldsymbol{\alpha}$ we propagate derivatives with respect to $\boldsymbol{\alpha}$ through $Q(\cdot)$, but keep $\boldsymbol{s}$ fixed. This differs from most other variational methods that keep all of $Q(\cdot)$ fixed when optimizing $\boldsymbol{\alpha}$. Note that the overall optimization is not guaranteed to converge, since the $\boldsymbol{s}$ updates are not guaranteed to increase the lower bound. In practice we find the optimization almost always increases and behaves well. The criterion and gradient computation has the same complexity as the conditional inference, but is much faster in practice because code for large matrix operations can be used. The memory requirements are not increased significantly, because $\boldsymbol{M}$ can be overwritten. The derivation is rather involved; it can be found in [12].

## 4 Experiments

In this section we present experimental results for the regression task of modeling of the dynamics of a 3-D, four-joint robot arm. The dataset is created using realistic simulation code which provides a mapping from twelve covariates (the angles, angular velocities and torques at each of the four joints of the arm) to four responses (the angular accelerations at the four joints).

We preprocessed the raw data by fitting a linear regression to the training set and replacing all responses by the corresponding residuals, then normalizing both covariate and response variables to have mean zero and variance one. This removal of the linear component of the regression helps clarify the relative contributions made by the nonlinear methods that are our focus. Finally the four responses were linearly mixed using randomly sampled unit length vectors to produce six response variables. Thus the dataset is a mapping from twelve covariates to six responses, where it is known that four latent variables are sufficient to capture the mapping.

The dataset sizes are $n = 1000$ for training and $500$ points for testing. We report mean squared error (MSE) and average marginal log probability (LOGP) in the experiments below.[3] To calibrate the numbers, note that linear regression would have an average MSE of 1 on this task.

We compare our model against a baseline method (INDEP) in which each response variable is simply modeled independently, i.e., taking $P = C$ and $\boldsymbol{\Phi} = \boldsymbol{I}$. We use the IVM technique for both models. For our model we use a joint active set $I$ of size $d$, while for the baseline we use individual active sets of size $d'$ per response variate. It is clear that for similar training set size and coverage by active points, training for INDEP is significantly faster than for SLFM, and in this study we do not attempt to equalize training times.

[3] LOGP is $\log Q(y_*|\boldsymbol{x}_*)$ averaged over the test set.

In both models we use the *squared-exponential* covariance function (SQEXP):

$$k_p(\boldsymbol{x}, \boldsymbol{x}') = \nu_p \exp\left(-\sum_l \frac{1}{2\theta_{p,l}^2}|x_l - x_l'|^2\right), \quad (6)$$

where $\theta_{p,l}$ is the length scale for dimension $l$, and $\nu_p > 0$ sets the overall variance. For the SLFM, the same kernel is used for all latent GPs and we set $\nu_p = 1$ since the variance can already be represented by scaling the columns of $\boldsymbol{\Phi}$. We allow different length scales for each input dimension because we find that this has a significant impact on the quality of prediction—if the length scale is constrained to be the same for all covariate dimensions then less relevant input dimensions tend to obscure the more relevant ones. Note that this large set of hyperparameters is adjusted based on the training set within our empirical Bayesian framework; no validation set is needed.

The mean squared errors and log probabilities are shown in Table 1 as a function of $P$ as $P$ is varied from 2 to 6. The model performs best at $P = 4$, although similar accuracy is achieved for $P = 4, 5, 6$. We do expect the performance to degrade for even larger values of $P$ but we have not investigated this. For the rest of this section we chose $P = 4$ since this is the smallest value supported by the data.

| $c \setminus P$ | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| 1 | 0.2930 | 0.2340 | 0.1220 | 0.1190 | 0.1130 |
| 2 | 0.2840 | 0.2950 | 0.1780 | 0.1890 | 0.1880 |
| 3 | 0.3940 | 0.1570 | 0.1080 | 0.1060 | 0.1030 |
| 4 | 0.3630 | 0.2980 | 0.1270 | 0.1490 | 0.1410 |
| 5 | 0.3080 | 0.2830 | 0.1760 | 0.1840 | 0.1810 |
| 6 | 0.5560 | 0.3010 | 0.1180 | 0.1190 | 0.1100 |
| LOGP | -5.770 | -4.571 | -2.342 | -2.516 | -2.466 |

Table 1: The mean squared errors for each response variate on the test set and the average log probability per training point assigned by our model for varying $P$.

Next we compared the SLFM with $P = 4$ to the baseline of independently modeled response variables. We used a training set of size $n = 1000$ and active sets of size $d = 1000$ and $d' = 180$ (if all INDEP active sets are disjoint, their union has size $6 \cdot 180 = 1080$). The results are shown in Table 2. Note that the MS errors for our model are smaller than for the baseline.

We also tested for the effects of varying the active set size $d$; results are given in Table 3.

We see that the active set size $d$ has a significant effect on prediction accuracy. The quadratic scaling in $d$ can be observed from the training times, except for the largest values of $d$, where the $O(d^3)$ component in the gradient computation dominates the $O(n\,P\,d^2)$ component.

Since our method models dependencies among the response variables, for every test point we have a joint predictive distribution over the response variables $\boldsymbol{y}_* \in \mathbb{R}^C$. This

| c | SLFM MSE | INDEP MSE | SLFM LOGP | INDEP LOGP |
|---|---|---|---|---|
| 1 | 0.122 | 0.133 | 0.018 | -0.110 |
| 2 | 0.178 | 0.202 | -0.244 | -0.335 |
| 3 | 0.108 | 0.152 | -0.025 | -0.352 |
| 4 | 0.127 | 0.179 | 0.011 | -0.271 |
| 5 | 0.176 | 0.202 | -0.340 | -0.349 |
| 6 | 0.118 | 0.135 | 0.053 | -0.046 |

Table 2: Comparing our model (SLFM) against the baseline (INDEP) on the robot arm task, with $C = 6, P = 4$. Rows correspond to response variables.

| $c \setminus d$ | 500 | 1000 | 2000 | 3000 |
|---|---|---|---|---|
| 1 | 0.174 | 0.122 | 0.096 | 0.067 |
| 2 | 0.285 | 0.178 | 0.107 | 0.094 |
| 3 | 0.228 | 0.108 | 0.072 | 0.062 |
| 4 | 0.283 | 0.127 | 0.082 | 0.068 |
| 5 | 0.281 | 0.176 | 0.100 | 0.090 |
| 6 | 0.196 | 0.118 | 0.090 | 0.066 |
| time | 382 | 1269 | 5806 | 16746 |

Table 3: MS test errors for each response as the active set size $d$ is varied. *time* gives the complete training time in seconds.

can be used to further improve the prediction of the model for any specific component $y_{*,c}$, if in addition to the covariates $\boldsymbol{x}_*$ we are also given a subset of the other response variables $y_{*,c'}$. In Table 4 we show the mean squared errors attained for response $c = 5$, when we are also given other responses. The errors are reduced significantly, especially for $c' = \{2\}$, and further improve as we observe more responses; in particular when we observe $c' = \{3, 4\}$. Note that for the baseline method each response variable is modeled independently and the predictive distribution over $\boldsymbol{v}_*$ factorizes, hence knowledge of other responses cannot help in predicting $y_{*,c}$.

| $c'$ | MSE | LOGP |
|---|---|---|
| $\{1\}$ | 0.1770 | -0.2640 |
| $\{2\}$ | 0.0380 | 0.2450 |
| $\{3\}$ | 0.1490 | -0.2760 |
| $\{4\}$ | 0.1320 | -0.2940 |
| $\{6\}$ | 0.1740 | -0.3440 |
| $\{3, 4\}$ | 0.112 | -0.221 |

Table 4: Improved predictions of the model on response variable $c = 5$ when the model is given other responses $y_{*,c'}$.

Finally we report an experiment that aimed at improving our understanding of how statistical strength is shared across response variables. We again focus on predicting response variate $c = 5$ in the task with 1000 training points. However, instead of presenting all 1000 covariate/response pairs simultaneously, we start by observing only response variable $c = 5$, for a subset of $l < 1000$ points. Subsequently, we are given all 1000 covariate vectors and the corresponding responses for a subset $c'$ not including $c = 5$. We ask whether this will improve our prediction of response variable $c = 5$. Note that this setup is similar to co-kriging scenarios mentioned in Section 1. Table 5 shows the mean squared errors attained for various values of $l$ and for various subsets $c'$ of additional observed response variables. We see a large improvement of the mean squared error for $c' = \{1, 2\}$. Even for $l = 50$ the errors are already smaller than those in Table 2. This is because of the strong dependencies between response variables $c = 2$ and $c = 5$ (as seen in Table 4). Note also that the case $c' = \{1, 2, 3, 4, 6\}$ performed worse than $c' = \{1, 2\}$, though still yielding a marked improvement over no additional training set ($c' = \emptyset$). This occurs because the functional that our method optimizes is a joint functional over the response variables, and thus depends more strongly on response variables with more training data. Performance as assessed by this functional indeed improved for larger $c'$. If our goal is to obtain good prediction for a particular response variable, we can consider a different functional which focuses on the response variable of interest.

| $l \setminus c'$ | $\emptyset$ | $\{1, 2\}$ | $\{1, 2, 3, 4, 6\}$ |
|---|---|---|---|
| 50 | 1.011 | 0.111 | 0.202 |
| 150 | 0.752 | 0.111 | 0.198 |
| 250 | 0.278 | 0.105 | 0.183 |

Table 5: Mean squared error on test set for varying training set sizes $l$ and additional response variables $c'$.

## 5 Discussion

We have described a model for nonlinear regression in problems involving multiple, linked response variables. In a manner reminiscent of factor analysis in the parametric setting, we model the response vector as (a function of) a linear combination of a set of independent latent Gaussian processes. This rather simple semiparametric approach to sharing statistical strength has a number of virtues—most notably its flexible parametrization in terms of sets of covariance kernels and its computational tractability. We presented an efficient approximate inference strategy based on the IVM. While our primary focus has been prediction, the inferential tools provided by the IVM also allow us to compute posteriors over various components of the model, in particular the latent factors and the parameters. Possible extensions of the model include placing an automatic relevant determination (ARD) prior on the columns of the mixing matrix $\boldsymbol{\Phi}$ and letting the model determine $P$ automatically. It is also of interest to consider ways in which the mixing matrix might be dependent on the covariates as well.

There are other ways of combining multiple Gaussian processes. [9] and [8] present models in which the hyperpa-

rameters of a set of Gaussian processes are endowed with a common prior. This hierarchical model couples the Gaussian processes as in the SLFM, but the amount of sharing that it induces is rather limited, since it involves only the hyperparameters of the Gaussian processes. In our approach the sharing involves entire processes and as such can be much more expressive. Note also that although we considered tasks involving a single regression problem with multiple responses, the SLFM can readily accommodate the setting in which there are multiple related tasks, each with a single response and with a separate training set.

As we have noted, the semiparametric approach presented here is an alternative to the parametric methodology of conditional random fields (CRFs) that has recently been the focus of attention in the machine learning and computer vision communities [5, 4]. When the response variables can plausibly be linked in a simple structure, for example according to a chain or a tree, the CRF approach would seem to be preferable to the SLFM approach. On the other hand, when the graph is not a chain, the potential intractability of the partition function can be a significant drawback of the CRF approach. In vision problems, for example, one would like to use a two-dimensional Markov random field for modeling dependencies, but this runs aground on the problem of the partition function. In our approach, couplings among variables arise by marginalizing over a latent set of linearly mixed Gaussian processes, and this provides an alternative, implicit approach to linking variables. In cases in which graphical models are intractable, this approach may provide the requisite tractability at a cost of modeling flexibility. Finally, note also that the SLFM approach is a kernel-based approach by definition; there is no need to explicitly "kernelize" the SLFM.

Several methods for multiple responses in regression have been proposed which involve posthoc combinations of the outputs of the independent baseline method. An example is the *curds and whey* method [2] for multiple linear regression. It is important to stress that our approach is fundamentally different in that the latent $u$ processes are fitted jointly using all data. These processes can represent conditional dependencies directly, while the processes of the baseline method only ever see marginal data for each response. Posthoc combination schemes should be successful if response dependencies are mainly unconditional but may fail to represent dependencies which change with $\boldsymbol{x}$. An advantage of posthoc methods is that they can be cheap computationally, having essentially the same scaling as the independent baseline (which they use as a subroutine). Whether a more flexible technique such as ours with a computational complexity closer to the baseline method exists is an open question.

### 5.1 Applications to classification

Our model can be extended to classification problems and to other problems involving non-Gaussian likelihoods $P(y_c|v_c)$. The basic idea is to again make use of GP-based techniques such as the IVM that have been extended to GP-based classification in the single response variable case [7]. The non-Gaussian likelihoods are effectively replaced by Gaussians whose parameters are determined by sequential moment matching.

The extension to classification is of particular interest in the multiple response variable setting because it allows us to address multi-label classification problems in which the class labels are not assumed to be mutually exclusive and may exhibit interesting and useful interdependencies.

In a preliminary investigation of this extension we considered the toy example shown in Figure 2. We sampled 500 two-dimensional covariate vectors uniformly at random from $[-1, 1]^2$ and labeled these vectors using eight binary response variables, one for each of the regions shown in the top left panel of Figure 2. There was no label noise, but $10\%$ of the $n = 500$ training responses were missing at random. We fit this data with an SLFM suitably extended to probit likelihoods, using $P = 3$ latent GPs, each with a different SQEXP kernel (Eq. 6) and with a single length scale parameter (i.e., $\theta_{p,l} = \theta_{p,l'}$).

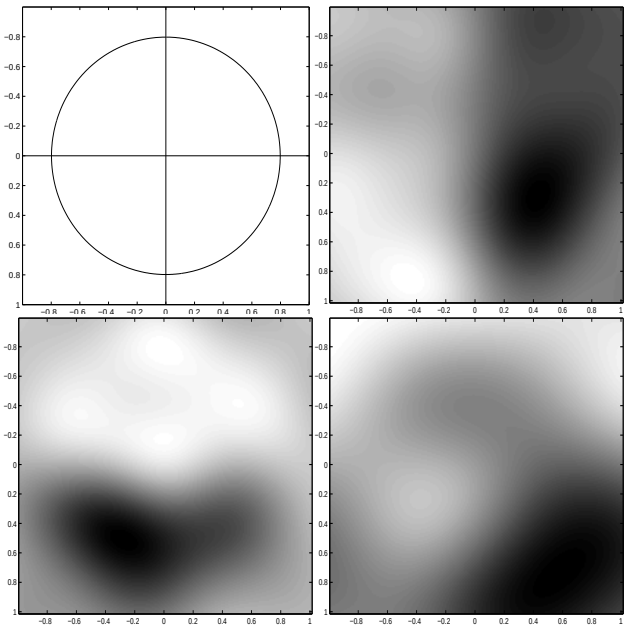

Figure 2: Top left: the eight regions. Rest: posterior mean function of the latent GPs. Light colors correspond to larger values.

After training, the test set errors for the eight response variables were $0.0345$, $0.0261$, $0.0133$, $0.0296$, $0.0602$, $0.0176$, $0.0494$ and $0.0230$. The remaining three panels in Figure 2 show the approximate posterior mean functions

for the three latent GPs. Roughly speaking, one GP is used for vertical discrimination, one for horizontal, and a third for inside-vs-outside separation (although the first two GPs also distinguish inside-vs-outside separation to a lesser extent). Thus we see that the model has formed a combinatorial code in which it is able to classify eight response variables using only three latent GPs.

### 5.2 Other issues

Computational issues remain a serious concern if the SFLM is to be scaled to larger problems. The main avenue open for tackling larger datasets is to refrain from scoring all remaining points for all later inclusions. In particular, after a number of initial inclusions selected from all remaining points we can potentially narrow down the candidate set stepwise by excluding points with the worst current scores. The empirical Bayes learning procedure then approximates the full likelihood by the likelihood restricted to the final candidate set (which always includes the active set). For very large tasks, even more elaborate caching strategies could be envisaged. A natural limit for the active set size $d$ is imposed by the $O(d^3)$ time and memory scaling.

While we have focused on the case $P < C$ in the current paper, it is also of great interest to explore cases in which $P > C$. In particular, in the $P < C$ regime the variable $\boldsymbol{v} \in \mathbb{R}^C$ is constrained to lie in a $P$-dimensional subspace; while the analogy to factor analysis suggests that this may be a useful constraint in some problems, it also may impose an overly narrow bottleneck on the regression mapping in other problems. There are several possible ways to remove this constraint and consider versions of the SFLM that operate in the $P > C$ regime. One interesting variant involves replacing Eq. 1 by $\boldsymbol{v} = \boldsymbol{v}^{(0)} + \boldsymbol{\Phi}\boldsymbol{u}$ where all components of $(\boldsymbol{v}^{(0)T}, \boldsymbol{u}^T)^T$ are conditionally independent and are given GP priors with different kernels. While this setup can be viewed as simply a particular choice of $\boldsymbol{\Phi}$ in a generic SFLM with $P > C$, the additional independences in the model aid in the design of approximate inference methods based on a variant of belief propagation.


### Acknowledgments

This work has been supported by a grant from the Intel Corporation, by a grant from Microsoft Research, and by a grant from DARPA in support of the CALO program.


## References


[1] A. Agarwal and B. Triggs. 3D human pose from silhouettes by relevance vector regression. In *Proceedings of the IEEE International Conference on Computer vision and Pattern Recognition*, 2004.

[2] L. Breiman and J. Friedman. Predicting multivariate responses in multiple linear regression. *J. Roy. Stat. Soc. B*, 59(1):3–54, 1997.

[3] N. Cressie. *Statistics for Spatial Data*. John Wiley & Sons, 2nd edition, 1993.

[4] S. Kumar and M. Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *Proceedings of IEEE International Conference on Computer Vision*, 2003.

[5] J. Lafferty, F. Pereira, and A. McCallum. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML 18*, 2001.

[6] G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M.I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[7] N. D. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In *Advances in NIPS 15*, pages 609–616, 2003.

[8] N.D. Lawrence and J.C. Platt. Learning to learn with the informative vector machine. In *Proceedings of the International Conference in Machine Learning*, 2004.

[9] U. Menzefricke. Hierarchical modeling with Gaussian processes. *Communications in Statistics–Simulation and Computation*, 29(4):1089–1108, 2000.

[10] R.M. Neal. Priors for infinite networks. Technical Report CRG-TR-94-1, Department of Computer Science, University of Toronto, 1994.

[11] M. Seeger. *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. PhD thesis, University of Edinburgh, July 2003. See `www.cs.berkeley.edu/~mseeger`.

[12] M. Seeger, Y.-W. Teh, and M. I. Jordan. Semiparametric latent factor models. Technical report, University of California at Berkeley, 2004. See `www.cs.berkeley.edu/~mseeger`.

[13] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in NIPS 16*, 2004. To appear.

[14] Vladimir N. Vapnik. *Estimation of Dependences based on Empirical Data*. Series in Statistics. Springer, 1st edition, 1982.