

EigenGP: KL-expansion based Gaussian process learning *

Yuan Qi
Purdue University
alanqi@cs.purdue.edu

Bo Dai
Purdue University
dai25@purdue.edu

Yao Zhu
Purdue University
zhu36@purdue.edu

September 13, 2022

Abstract

Gaussian processes (GPs) provide a nonparametric representation of functions. Given N training points, the exact GP inference incurs high computational cost. In this paper, we propose a sparse Gaussian process model, EigenGP, based on Karhunen-Loève (KL) expansions of a GP prior. We use the Nyström approximation to obtain eigenfunctions of the covariance function and use an empirical Bayesian approach to select these eigenfunctions. To handle nonlinear likelihoods, we develop an efficient expectation propagation inference algorithm, and couple it with expectation maximization for evidence maximization. By selecting eigenfunctions of Gaussian kernels that are associated with data clusters, EigenGP is also suitable for semi-supervised learning. Our experimental results demonstrate improved predictive performance of EigenGP over alternative state-of-the-art sparse GP and semi-supervised learning methods for regression, supervised and semi-supervised classification.

keywords: Gaussian processes, eigenfunctions, continuous basis functions, evidence maximization, regression, semi-supervised and supervised classification, and kernel learning

1 Introduction

Gaussian processes (GPs) are powerful nonparametric Bayesian tools for modeling unknown functions. However, GP inference is computationally challenging. Even for the regression case where the GP prediction formula is analytic, training the exact GP model with N points demands an $O(N^3)$ computational cost for inverting the covariance matrix. To reduce the computational cost, a variety of approximate sparse GP inference approaches have been developed (Williams & Seeger, 2001; Csató & Oppner, 2002; Snelson & Ghahramani, 2006; Lázaro-Gredilla et al., 2010; Williams & Barber, 1998; Qi et al., 2010)—for example, using the Nyström method to approximate covariance matrices (Williams & Seeger, 2001) or grounding the GP on a small set of (blurred) basis points (Snelson & Ghahramani, 2006; Qi et al., 2010). An elegant unifying view for various approximate sparse GP regression models are given by Quiñero-Candela & Rasmussen (2005). Note that all these sparse GP approaches gain computational efficiency with certain approximations—possibly degenerating prediction accuracy.

In this paper, we propose a new sparse GP learning approach based on Karhunen-Loève (KL) expansions. KL expansions have been used in a variety of scientific and engineering domains. For example, based on KL expansions, principle component analysis projects a random vector onto eigenvectors associated with largest eigenvalues for signal compression and noise reduction. Moving our focus from random vectors (e.g., noisy data input) to stochastic processes (e.g., function values), we use Karhunen-Loève (KL) expansions to obtain eigenfunctions of a GP prior. Truncated KL expansions of GPs have been used widely in the scientific computing community, to solve stochastic partial differential equations and quantify uncertainty of large finite element systems (e.g., (Marzouk & Najm, 2009) and (Schenk & Schuëller, 2005)). For these applications, these KL expansions are often obtained via (generalized) polynomial chaos representations. The complexity of these representations, however, increases quickly with dimensionality of data, making them unsuitable for high dimensional problems. Fortunately we can use the Nyström approximation

*This work was sponsored by the NSF grants IIS-0916443, IIS-1054903, ECCS-0941533, and CCF-0939370. All the authors gratefully acknowledge the support of the grants. Any opinions, findings, and conclusion or recommendation expressed in this material are those of the author(s) and do not necessarily reflect the view of the funding agencies or the U.S. government.

(Williams & Seeger, 2001) to efficiently compute eigenfunctions based on a subset of the data for high dimensional problems.

Three differences distinguish our method from the previous Nyström method. First, we define a valid probabilistic model based on an eigen-decomposition of the GP prior. By contrast, the previous model numerically approximates covariance matrices by low-rank matrices and its predictive distribution is not well-formed in a probabilistic framework. Accordingly, both predictive means and variances of these two methods are different. Second, using an empirical Bayesian framework, we maximize the model marginal likelihood to *select* eigenfunctions and further adjust their weights. Third, because eigenfunctions of a Gaussian covariance function correspond to clusters of data points (both labeled and unlabeled), we can select eigenfunctions to conduct semi-supervised learning.

The main contributions of this paper are summarized as follows:

- We propose a new sparse GP model, EigenGP, based on KL expansions of GP priors for regression, classification and semi-supervised classification.
- We present an expectation propagation algorithm to efficiently estimate the posterior process of the new model for both regression and classification tasks. The computational cost for training is $O(NL^2)$ —instead of $O(N^3)$ —where L is the number of eigenfunctions in the model. We also present an expectation maximization based algorithm to estimate the weights of the basis functions.
- Finally, our experimental results demonstrate that EigenGPs achieve superior predictive performance over alternative state-of-the-art approaches.

2 Background of Gaussian processes

We denote N independent and identically distributed samples as $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}_N$, where $\mathbf{x}_i$ is a d dimensional input and y_i is a scalar output. We assume there is a latent function f that we are modeling and the noisy realization of f at $\mathbf{x}_i$ is y_i .

A Gaussian process places a prior distribution over the latent function f . Its projection $\mathbf{f}_{\mathbf{x}}$ at $\{\mathbf{x}_i\}_{i=1}^N$ defines a joint Gaussian distribution: $p(\mathbf{f}_{\mathbf{x}}) = \mathcal{N}(\mathbf{f}|\mathbf{m}^0, \mathbf{K})$, where $m_i^0 = m^0(\mathbf{x}_i)$ is the mean function and $\mathbf{K}_{ij} = \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)$ is the covariance function, encoding the prior notion of smoothness. Normally the mean function is simply set to be zero and we follow this tradition in this paper. A typical kernel covariance function is

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}' - \mathbf{x}\|^2}{2\eta^2}\right), \quad (1)$$

where η is a hyperparameter. This is known as the Gaussian kernel function.

For regression, we use a Gaussian likelihood function

$$p(y_i|f) = \mathcal{N}(y_i|f(\mathbf{x}_i), v_y), \quad (2)$$

where v_y is the observation noise. For classification, the data likelihood has the form

$$p(y_i|f) = (1 - \epsilon)\sigma(f(\mathbf{x}_i)y_i) + \epsilon\sigma(-f(\mathbf{x}_i)y_i) \quad (3)$$

where ϵ models the labeling error and $\sigma(\cdot)$ is the link function (e.g., a probit function).

Given the Gaussian process prior over f and the data likelihood, the exact posterior process is

$$p(f|\mathcal{D}, \mathbf{y}) \propto GP(f|0, k) \prod_{i=1}^N p(y_i|f) \quad (4)$$

For the regression problem, the posterior process has an analytical form. But to make a prediction on a new sample, we need to invert a N by N matrix. If the training set is big, this matrix inversion will be too costly. For classification or other nonlinear problems, the computational cost is even higher because we do not have an analytical solution to the posterior process and the complexity of the process grows with the number of training samples.

3 EigenGP

Before we present the new model, let us consider the Karhunen-Loève (KL) expansion of a zero-mean Gaussian process $f(\mathbf{x})$:

$$f(\mathbf{x}) \approx f^L(\mathbf{x}) = \sum_{j=1}^L \sqrt{\lambda_j} \phi^j(\mathbf{x}) \xi_j \quad (5)$$

where $\{\xi_j\}$ are centered, normalized, uncorrelated random variables: $\mathbf{E}[\xi_j] = 0$ and $\mathbf{E}[\xi_j \xi_k] = \delta_{ij}$.

Although in general we cannot solve the exact eigenfunctions $\phi^j(\mathbf{x})$ of a GP prior, we can use Galekin projection to approximate them by Hermite polynomials (Marzouk & Najm, 2009). For high dimensional problems, we need a tensor product of univariate Hermite polynomials, dramatically increasing the number of model parameters. To avoid this problem, we use the Nyström method (Williams & Seeger, 2001) that allows us to efficiently obtain an approximation to the eigenfunctions in high dimensional spaces.

Specifically, given some basis points $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_Q]$, we approximate the j -th eigenfunction $\phi^j(\mathbf{x})$ by

$$\phi^j(\mathbf{x}) = \frac{\sqrt{Q}}{\lambda_j^{(Q)}} \mathbf{k}(\mathbf{x}) \tilde{\mathbf{u}}_j = \mathbf{k}(\mathbf{x}) \mathbf{u}_j \quad (6)$$

where $\mathbf{k}(\mathbf{x}) \triangleq [k(\mathbf{x}, \mathbf{b}_1), \dots, k(\mathbf{x}, \mathbf{b}_Q)]$, λ_j^q and $\tilde{\mathbf{u}}_j$ are the j -th eigenvalue and eigenvector of the covariance matrix $\mathbf{K}_{\mathbf{B}}$ evaluated at $\mathbf{B}$, and $\mathbf{u}_j = \frac{\sqrt{(Q)}}{\lambda_j^{(Q)}} \tilde{\mathbf{u}}_j$. We use these eigenfunctions (6) to construct a new covariance function:

$$\tilde{k}(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^L w_j \phi^j(\mathbf{x}) \phi^j(\mathbf{x}') \quad (7)$$

where $L \leq Q$, and w_j is the weight for the j -th basis function. If we set $w_j = \lambda_j^q / Q$, then w_j is an eigenvalue of the original kernel k . We can also sparsify the weight vector $\mathbf{w}$ by maximizing the model marginal likelihood of the GP model as shown in Section 5.

For Gaussian likelihoods (i.e., regression), we can use the matrix inverse lemma to efficiently compute the posterior process and the marginal likelihood. For nonlinear likelihoods (such as probit or Poisson likelihoods), we use expectation propagation (Minka, 2001b) to map the nonlinear likelihood into a linear Gaussian form $\mathcal{N}(g_i | f(\mathbf{x}_i), \tau_i)$. As a result, the (approximate) posterior process of EigenGP is

$$q(f) \propto GP(f | 0, \tilde{k}) \prod_{i=1}^N \mathcal{N}(g_i | f(\mathbf{x}_i), \tau_i). \quad (8)$$

Let $\mathbf{g} = [g_1, \dots, g_N]$, $\boldsymbol{\tau} = [\tau_1, \dots, \tau_N]$, and $\Phi_{\mathcal{D}}$ (a N by L matrix) represent the values of the L basis functions at the N training points. Then combining (7) and (8) with equations (6.66) and (6.67) in (Bishop, 2006), we obtain the following proposition.

Proposition 3.1. *The posterior process $q(f)$ defined in (8) has the mean function $m(\mathbf{x})$ and covariance function $V(\mathbf{x}, \mathbf{x}')$:*

$$m(\mathbf{x}) = \mathbf{k}(\mathbf{x})\mathbf{U}\alpha \quad \text{and} \quad V(\mathbf{x}, \mathbf{x}') = \tilde{\mathbf{k}}(\mathbf{x}, \mathbf{x}') - \mathbf{k}(\mathbf{x})\mathbf{U}\beta\mathbf{U}^T\mathbf{k}(\mathbf{x}') \quad (9)$$

where $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_L]$, $\mathbf{W} = \text{diag}(\mathbf{w})$, $\beta = \mathbf{W}\Phi_{\mathcal{D}}^T(\tilde{\mathbf{K}} + \text{diag}(\boldsymbol{\tau}))^{-1}\Phi_{\mathcal{D}}\mathbf{W}$, $\tilde{\mathbf{K}} = \Phi_{\mathcal{D}}\mathbf{W}\Phi_{\mathcal{D}}^T$ and $\alpha = \beta\mathbf{g}$.

3.1 Predictive distribution

Although the previous Nyström method (Williams & Seeger, 2001) can significantly speed up the GP inference, as pointed by Quiñonero-Candela & Rasmussen (2005), its approximation does not correspond to a well-formed probabilistic model. Specifically, the predictive posterior distribution of the previous Nyström method is

$$p(f(\mathbf{x}_*)|\mathbf{y}) \sim \mathcal{N}(\mathbf{k}(\mathbf{x}_*)^T[\tilde{\mathbf{K}} + \sigma^2\mathbf{I}]^{-1}\mathbf{y}, k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}(\mathbf{x}_*)^T[\tilde{\mathbf{K}} + \sigma^2\mathbf{I}]^{-1}\mathbf{k}(\mathbf{x}_*)) \quad (10)$$

where the scalar $k(\mathbf{x}_*, \mathbf{x}_*)$ is the kernel function value at $\mathbf{x}_*$ and the vector $\mathbf{k}(\mathbf{x}_*)$ are the kernel function values between $\mathbf{x}_*$ and all the training points. Note that the variance of this posterior distribution is not guaranteed to be nonnegative.

By contrast, the variance of the predictive posterior distribution of our new model is $\tilde{\mathbf{k}}(\mathbf{x}_*, \mathbf{x}_*) - \tilde{\mathbf{k}}(\mathbf{x}_*)^T[\tilde{\mathbf{K}} + \sigma^2\mathbf{I}]^{-1}\tilde{\mathbf{k}}(\mathbf{x}_*)$, which is guaranteed to be nonnegative. This is unsurprising since we define a new Gaussian process prior and it leads to a valid Gaussian process posterior.

However, this covariance is constrained by the eigen-subspace. As a result, when the test point is far from the training points, the variance goes to zero. We can see this point by realizing both $\tilde{\mathbf{k}}(\mathbf{x}_*, \mathbf{x}_*)$ and $\tilde{\mathbf{k}}(\mathbf{x}_*)$ become zeros when $\mathbf{x}_*$ is very far from all the training points. To fix this degeneration issue, we replace the low rank kernel function $\tilde{\mathbf{k}}(\mathbf{x}_*, \mathbf{x}_*)$ in the variance by $\mathbf{k}(\mathbf{x}_*, \mathbf{x}_*)$. Then we obtain the predictive posterior distribution of our model:

$$p(f(\mathbf{x}_*)|\mathbf{y}) \sim \mathcal{N}(\tilde{\mathbf{k}}(\mathbf{x}_*)^T[\tilde{\mathbf{K}} + \sigma^2\mathbf{I}]^{-1}\mathbf{y}, k(\mathbf{x}_*, \mathbf{x}_*) - \tilde{\mathbf{k}}(\mathbf{x}_*)^T[\tilde{\mathbf{K}} + \sigma^2\mathbf{I}]^{-1}\tilde{\mathbf{k}}(\mathbf{x}_*)), \quad (11)$$

whose variance, when $\mathbf{x}_*$ is far away, nicely reduces to the prior variance—just like a GP model. Also, note that the predictive mean is also different that of the previous Nyström method.

The new model combines the low-rank approximation with the Deterministic Training Conditional (DTC) Approximation scheme described by Quiñonero-Candela & Rasmussen (2005). Unlike the previous DTC approximation (Seeger & Williams, 2003), we do not use projected latent variables to approximate the exact GP model; instead, we replace exact conditional models on *sparse* latent variables with low-rank models on all the training data points (Please see the Appendix for details). Although this modified model does not correspond to a GP, it is still based on a valid probabilistic model.

4 Expectation propagation inference

Now we describe the expectation propagation inference to estimate (α, β) (equivalently, $(\mathbf{g}, \boldsymbol{\tau})$), so that the sparse posterior process $q(f)$ accurately approximates the exact posterior (4). This approximation is possible because the approximating family in (8) is an exponential family. EP approximates each likelihood factor $p(y_i|f)$ by a message $\tilde{t}_i(f)$ in this family. To this end, EP repeats three steps: message deletion, projection, and message update. In the message deletion step, we compute the partial belief $q^{\setminus i}(f; \alpha^{\setminus i}, \beta^{\setminus i})$ by removing a message $\tilde{t}_i$ from the approximate posterior $q(f; \alpha, \beta)$: $q^{\setminus i}(f; \alpha^{\setminus i}, \beta^{\setminus i}) \propto q(f; \alpha, \beta)/\tilde{t}_i$. In the data projection step, we minimize the KL divergence between $\tilde{p}(f) \propto p(y_i|f)q(f; \alpha^{\setminus i}, \beta^{\setminus i})$ and the new approximate posterior $q(f; \alpha, \beta)$, such that the information from each data point is incorporated into the model. Finally, the message $\tilde{t}_i$ is updated based on the new and old posteriors: $\tilde{t}_i \propto q(f; \alpha, \beta)/q^{\setminus i}(f; \alpha^{\setminus i}, \beta^{\setminus i})$.

We start with the projection step, because it is the most crucial step of EP. From (8), we see that EigenGP defines an exponential family with features $(\mathbf{f}_\mathbf{x}, \mathbf{f}_\mathbf{x} \mathbf{f}_\mathbf{x}^\top)$. Therefore, the minimization of the KL divergence between $\tilde{p}(f)$ and its posterior process $q(f)$ is achieved by moment matching on the mean $\mathbf{m}_\mathbf{x}$ and the covariance $\mathbf{V}_\mathbf{x}$ of $\mathbf{f}_\mathbf{x}$. Similar to the derivation of Bayes Point Machine in Chapter 5 of (Minka, 2001a), the moment matching equations for GPs are

$$\mathbf{m}_\mathbf{x} = \mathbf{m}_\mathbf{x}^{\setminus i} + \mathbf{V}^{\setminus i}(\mathbf{X}, \mathbf{x}_i) \frac{d \log Z}{dm^{\setminus i}(\mathbf{x}_i)} \text{ and } \mathbf{V}_\mathbf{x} = \mathbf{V}_\mathbf{x}^{\setminus i} + \mathbf{V}^{\setminus i}(\mathbf{X}, \mathbf{x}_i) \frac{d \log^2 Z}{dm^{\setminus i}(\mathbf{x}_i)^2} \mathbf{V}^{\setminus i}(\mathbf{x}_i, \mathbf{X}), \quad (12)$$

where $Z = \int q^{\setminus i}(f) p(y_i | f) df$, and $m^{\setminus i}(\mathbf{x}_i)$ is the mean of $q^{\setminus i}(f(\mathbf{x}_i))$ and $\mathbf{V}_\mathbf{x}^{\setminus i}(\mathbf{x}_i, \mathbf{X})$ is covariance matrix of the EigenGP at $\mathbf{x}_i$ and $\mathbf{X}$.

Given $\mathbf{m}_\mathbf{x}$ and $\mathbf{V}_\mathbf{x}$, we want to compute α and β . To do so, first note that, from Proposition 1, it follows that

$$\mathbf{m}_\mathbf{x} = \mathbf{K}_{\mathbf{x}, \mathbf{B}} \mathbf{U} \alpha \quad \mathbf{V}_\mathbf{x} = \tilde{\mathbf{K}} - \mathbf{K}_{\mathbf{x}, \mathbf{B}} \mathbf{U} \beta \mathbf{U}^\top \mathbf{K}_{\mathbf{x}, \mathbf{B}}^\top \quad (13)$$

where the (i, j) element of the matrix $\mathbf{K}_{\mathbf{x}, \mathbf{B}}$ is $k(\mathbf{x}_i, \mathbf{b}_j)$, and $\mathbf{x}_i$ and $\mathbf{b}_j$ are the i -th labeled data point and the j -th data point used for the eigenvector estimation.

Then combining (12) and (13), we obtain

$$\alpha = \alpha^{\setminus i} + \mathbf{h} \frac{d \log Z}{dm^{\setminus i}(\mathbf{x}_i)} \quad \beta = \beta^{\setminus i} - \mathbf{h} \frac{d^2 \log Z}{dm^{\setminus i}(\mathbf{x}_i)^2} \mathbf{h}^\top \quad (14)$$

where $\mathbf{h} \triangleq \mathbf{K}^{-1} \mathbf{V}^{\setminus i}(\mathbf{X}, \mathbf{x}_i) = \mathbf{p}_i - \beta^{\setminus i} \phi_i$, $\mathbf{p}_i \triangleq \text{diag}(\mathbf{w}) \phi_i$ and $\phi_i \triangleq \mathbf{k}(\mathbf{x}_i) \mathbf{U}$ is precomputed before the EP inference. The above equations define the projection step to obtain the new $q(f)$. Note that in general $KL(\tilde{p}(f) | q(f)) \neq 0$ unless the likelihood function is a Gaussian. Please see the Appendix for the specific updates for regression and classification.

Given the new $q(f)$, we will update the message $\tilde{t}_i(f)$ according to the ratio $q(f)/q^{\setminus i}(f)$:

$$\tilde{t}_i(f) = \frac{|\mathbf{V}_\mathbf{x}|^{-\frac{1}{2}} \exp(-\frac{1}{2}(\mathbf{f}_\mathbf{x} - \mathbf{m}_\mathbf{x}) \mathbf{V}_\mathbf{x}^{-1} (\mathbf{f}_\mathbf{x} - \mathbf{m}_\mathbf{x}))}{|\mathbf{V}_\mathbf{x}^{\setminus i}|^{-\frac{1}{2}} \exp(-\frac{1}{2}(\mathbf{f}_\mathbf{x} - \mathbf{m}_\mathbf{x}^{\setminus i}) (\mathbf{V}_\mathbf{x}^{\setminus i})^{-1} (\mathbf{f}_\mathbf{x} - \mathbf{m}_\mathbf{x}^{\setminus i}))} = \mathcal{N}(f(\mathbf{x}_i) | g_i, \tau_i) \quad (15)$$

where

$$\tau_i \triangleq -\left(\frac{d^2 \log Z}{(dm^{\setminus i}(\mathbf{x}_i))^2}\right)^{-1} - v^{\setminus i}(\mathbf{x}_i) \quad g_i \triangleq m^{\setminus i}(\mathbf{x}_i) - \left(\frac{d^2 \log Z}{(dm^{\setminus i}(\mathbf{x}_i))^2}\right)^{-1} \frac{d \log Z}{dm^{\setminus i}(\mathbf{x}_i)} \quad (16)$$

where $v^{\setminus i}(\mathbf{x}_i) = \phi_i^\top \mathbf{h}$ and $m^{\setminus i}(\mathbf{x}_i) = \phi_i^\top \alpha^{\setminus i}$. The function $\tilde{t}_i(f)$ can be viewed as a message from the i -th data point to $q(f)$.

If we want to approximate the marginal likelihood of the model, we need to scale $\mathcal{N}(f(\mathbf{x}_i) | g_i, \tau_i)$ so that the message $\tilde{t}_i(f) = s_i \exp(f(\mathbf{x}_i) | g_i, \tau_i) = Z q(f) / q^{\setminus i}(f)$ preserves the local ‘evidence’—in other words, $\int \tilde{t}_i(f) q^{\setminus i}(f) df = \int t_i(f) q^{\setminus i}(f) df$. It is easy to show

$$\log s = \log Z - \frac{1}{2} \log(-(\log Z)'' \tau_i) - \frac{((\log Z)')^2}{2(\log Z)''}$$

where $(\log Z)'$ and $(\log Z)''$ are the first- and the second- order derivatives of $\log Z$ over $m^{\setminus i}(\mathbf{x}_i)$.

To delete a message $\tilde{t}_i(f)$, we need to compute $q^{\setminus i}(f) \propto q(f) / \tilde{t}_i(f)$. Instead of computing this ratio directly, we can equivalently multiply its reciprocal with the current $q(f)$. Then we can solve the multiplication by minimizing $KL(q(f) | q^{\setminus i}(f) \tilde{t}_i(f))$ over $q^{\setminus i}(f)$. Since $q(f)$, $q^{\setminus i}(f)$, and $\tilde{t}_i(f)$ all have the form of the exponential family, this minimal value of this KL is 0, so that $q^{\setminus i}(f) \propto q(f) / \tilde{t}_i(f)$. This KL minimization can be easily done by the moment matching equations very similar to (14):

$$\alpha^{\setminus i} = \alpha + \mathbf{h}^{\setminus i} \frac{d \log \tilde{Z}_d}{dm(\mathbf{x}_i)} \quad \beta^{\setminus i} = \beta - \mathbf{h}^{\setminus i} \frac{d^2 \log \tilde{Z}_d}{(dm(\mathbf{x}_i))^2} (\mathbf{h}^{\setminus i})^T \quad (17)$$

where

$$\mathbf{h}^{\setminus i} \triangleq \mathbf{p}_i - \beta \phi_i \quad \tilde{Z}_d = \int \frac{1}{\tilde{t}_i(f)} q^{\setminus i}(f) df \propto \mathcal{N}(u_i | \mathbf{p}_i^T \mathbf{m}_x^{\setminus i}, -\tau_i + v(\mathbf{x}_i)) \quad (18)$$

$$\frac{d^2 \log \tilde{Z}_d}{dm(\mathbf{x}_i)^2} = \tau_i - v(\mathbf{x}_i) \quad \frac{d \log \tilde{Z}_d}{dm(\mathbf{x}_i)} = \frac{d^2 \log Z}{dm(\mathbf{x}_i)^2} (m(\mathbf{x}_i) - g_i) \quad (19)$$

And $m(\mathbf{x}_i) = \phi_i^T \alpha$ and $v(\mathbf{x}_i) = \phi_i^T (\text{diag}(\mathbf{w}) - \beta) \phi_i = \phi_i^T \mathbf{h}^{\setminus i}$.

Computational complexity. Since $\frac{d^2 \log Z}{(dm(\mathbf{x}_i))^2}$ is a scalar and $\mathbf{h}$ is a L by 1 vector, it takes $O(L^2)$ to update β via (14). Similarly, it takes $O(L^2)$ to update $\beta^{\setminus i}$ via (17). Therefore, given N training points, the computational cost is $O(L^2 N)$ per EP iteration over all the data points. Since in practice the number of EP iterations is small (e.g., 10), the overall cost is $O(L^2 N)$. As shown in the next section, the computation cost will be further significantly reduced to $O(r^2 N)$ where r is the actual eigenfunctions used in the EP iterations and $r < L$. In addition, we have the cost of computing the top L eigenvectors $\mathbf{U}$ of the Q by Q gram matrix $\mathbf{K}$; using iterative algorithms such as the Lanczos method, it takes $O(Q^2 L)$. In summary, the complexity of the EP inference is $O(Q^2 L + NL^2)$ where L is the truncation level in (5). We give the summary of the EP algorithm and the marginal likelihood formula in the Appendix.

5 Selection of eigenfunctions and semi-supervised learning

In the previous section we assume the weights $\mathbf{w}$ are known. We can improve the model by either selecting only the eigenfunctions associated with the largest L eigenvalues (out of Q of them), or sparsify the values of $\mathbf{w}$, based on the training data. For regression problems, the truncation with the top eigenfunctions might be sufficient already—because given a fixed number of any orthogonal basis functions, the truncated KL expansion minimizes the mean squared approximation error. (Of course, we only obtain approximate eigenfunctions in practice, thus not minimizing the error exactly.) We can determine the truncation level L by incrementally increasing it until the model evidence decreases. This procedure can be done efficiently by sequentially applying the matrix inverse lemma to compute the evidence.

However, for classifications tasks, in particular, semi-supervised classification tasks, sparsifying the eigenfunctions can improve the model significantly. The reason is that we can explore an interesting property of Gaussian kernel: its eigenfunctions of correspond to clusters of data points. Thus we can first use both labeled and *unlabeled* data to learn clusters in the whole dataset. Assuming that most data points in the same cluster share the same label, we can then make predictions for unlabeled data points in the same cluster. However, because some of the eigenfunctions may seriously violate this assumption, using all of them may hurt the model’s predictive performance. Instead of using a predefined threshold on the number of sign changes by Sinha & Belkin (2010), we can achieve this task in a principled way since our model is well grounded in a probabilistic framework. Specifically, we select the eigenfunctions by maximizing the model marginal likelihood.

To select eigenfunctions—equivalently sparsify $\mathbf{w}$ —we maximize the marginal likelihood of the EigenGP over $\mathbf{w}$ via automatic relevance determination (ARD) (MacKay, 1992). To do so, we use an EP-EM approach. In the E-step, we perform the EP inference to obtain $q(\mathbf{f})$. For the M-step, we maximize the expected likelihood $E_q(\mathbf{f})[\log p(y|f)]$ over $\mathbf{w}$, we can obtain the following update:

$$w_j^{new} = w_j - \beta_{j,j} + \alpha_j^2. \quad (20)$$

Besides the EM updates, we can use an active-set method (Faul & Tipping, 2001; Qi et al., 2004) to obtain efficient updates of $\mathbf{w}$. Please see the Appendix for more details about the EM and the active-set

method. We can certainly explore other sparsification approaches, l_1 penalty (i.e, a Laplace prior), an elastic net penalty, or spike and slab priors to sparsify $\mathbf{w}$. But a thorough comparison on various sparsity penalty functions or priors is out of the scope of this paper.

6 Related work

Our approach is significantly different from many previous low-rank sparse GP approaches. The differences between ours and the previous Nyström approach (Williams & Seeger, 2001) have been discussed in Section 1 and 3.1 already. Essentially, the previous approaches (Nyström or the incomplete Cholesky factorization approximations (Yan & Qi, 2010)) aim at a low-rank approximation to the finite covariance/kernel matrix used in training—purely from a numerical approximation perspective. By contrast, our approach approximates the GP prior based on KL expansions and defines a valid probabilistic model.

With finite number of basis functions, our model bears some similarity to relevance vector machines (RVMs) (Tipping, 2000) and the model proposed by Lázaro-Gredilla et al. (2010). But our basis functions are eigenfunctions, very different from these models. And our models can handle semi-supervised learning while they cannot. And our model handles the uncertainty better than RVMs: RVMs give zero variance for predictions of data points far from the training dataset, while ours does not (as shown in (11)). As an alternative to (11), we can also add a delta function to the covariance, as suggested by Quiñero-Candela et al. (2007), to improve the modeling of the uncertainty.

7 Experiments

7.1 Regression

For regression, we compared EigenGPs with the full GP and two sparse approximate GP algorithms: the Nyström method (Williams & Seeger, 2001), the pseudo-input approach (also, known as Fully Independent Training Conditional (FITC) approximations) (Snelson & Ghahramani, 2006). We optimize the kernel width for full GP and use them for the Nyström method and EigenGPs for a simple comparison. We downloaded the code from this site¹ and used it to optimize both the kernel width for each dimension and the basis points via evidence maximization. Because the optimization is sensitive to local optima (as observed in (Snelson & Ghahramani, 2006; Qi et al., 2010)), we tried many initial values/schemes to improve the results of FITC and presented here the best results we obtained (which is also better than the results of FITC using the full GP kernel parameters).

We compared the prediction accuracy of these methods under the same computational cost. Let Q_1 and L be the number of basis points and the total number of eigenfunctions of EigenGPs and the Nyström method, and Q_2 be the number of basis points of FITC. The computational complexity of EigenGPs is $O(Q_1^2 L + NL^2)$, while FITC is $O(NQ_2^2)$. To let these algorithms have the same computational cost, we required $O(Q_1^2 L + NL^2) = O(NQ_2^2)$. To achieve this, we can set $L = Q_2$ and Q_1 is an integer smaller than $\sqrt{NQ_2}$. Here we simply set $Q_1 = Q_2$ as well in our experiments (This can allow the competing methods to use more time since $N > Q_2$).

The results on two benchmark datasets, Boston Housing and Pumadyn-8nm, are reported in 1. For EigenGPs, we denote the version that selects the top L eigenfunctions as EigenGP* and the version that sparsifies weights $\mathbf{w}$ as EigenGP. These two datasets contain 506 and 8192 data points. From these two datasets, we randomly selected 400 and 2000 points for training, respectively. And we used the rest for testing. We repeated these experiments 10 times. Apparently while the two versions of EigenGPs perform very similarly, both of them are very competitive compared to the competing methods. The average root mean square errors (RMSE) of the Nyström method on Boston housing are 312.5, 41.68, 15.17 and 6.480

¹http://www.gatsby.ucl.ac.uk/snelson/SPGP_dist.tgz

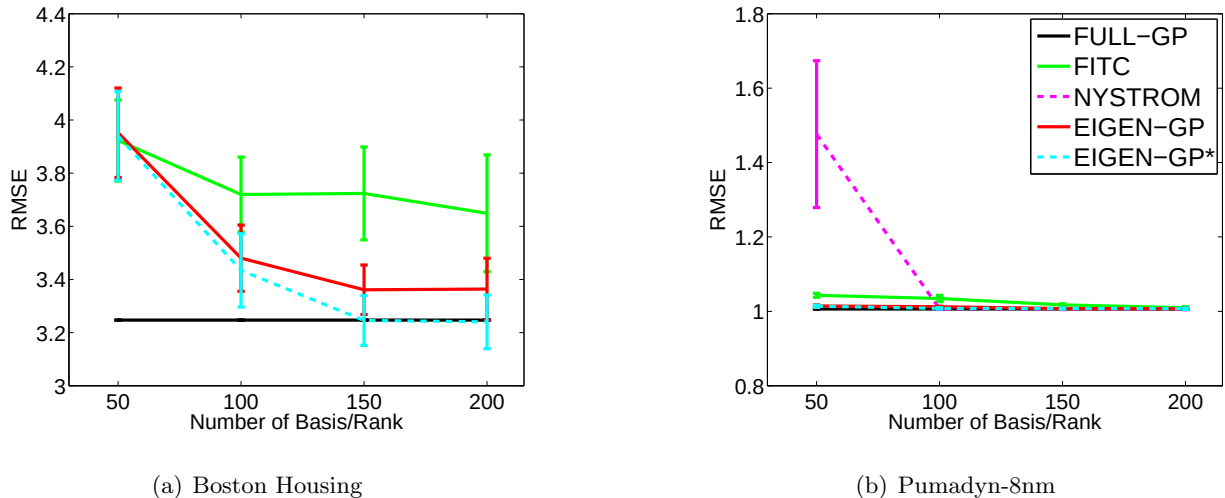


Figure 1: Regression results on Boston Housing and Pumadyn-8nm. The results are averaged over 10 runs, and the error bars represent the standard errors. The results of the Nyström method on Boston Housing are not included in the figure, since their errors are much larger than the others.

with 50, 100, 150 and 200 basis points, respectively. The corresponding standard errors are 70.45, 11.23, 6.194 and 1.502. Despite the similarity between EigenGPs and the Nyström method and the fact they used the identical setting in our experiments, EigenGPs significantly outperform the Nyström method on Boston Housing consistently and on Pumadyn-8nm when the number of basis points is small.

7.2 Supervised classification

For supervised classification, we compared EigenGPs with the full GP and three sparse approximate GP algorithms: the Nyström method (Williams & Seeger, 2001), Sparse Online Gaussian Processes (SOGP) (Csató & Opper, 2002), and Fully Independent Training Conditional (FITC) models (Snelson & Ghahramani, 2006; Naish-Guzman & Holden, 2008). We used the Gaussian kernel function for all the GP classification algorithms and tuned their kernel widths by extensive cross-validation. For the Nyström method, we used Laplace’s method to approximate the posterior distribution as described in (Williams & Seeger, 2001). A better comparison with the Nyström method would be using EP, instead of the less effective Laplace’s method. However, there is no previous work that combines EP with the Nyström method and the development of this algorithm is out of the scope of this paper. For the FITC model, we used EP for the posterior approximation as proposed by Naish-Guzman & Holden (2008). Although we did not maximize the model evidence to learn the kernel parameters as in (Naish-Guzman & Holden, 2008) (thus, we may not obtain their best results), we found in practice that our extensive cross-validation of kernel parameters often lead to prediction accuracies at least comparable to those of FITC based on evidence maximization.

We tested these algorithms on two datasets: Spambase and USPS. Each digit in USPS is treated as a 256 dimensional vector. In Spambase, we randomly selected 2300 data points for training and the rest 2300 points as the test set. For the USPS dataset, we conducted three digit classification tasks: 8 vs 9, 3 vs 8, and 5 vs 8. We randomly selected 1200 training and 1000 test points. For each split in Spambase, we chose the centers from K-means clustering of a randomly selected subset of the whole dataset as basis points for all the sparse GP methods. In USPS, we used the same method to selecting basis points when the number of basis points is small and, while the basis number is above 500, we selected the basis points randomly. We repeated these experiments 10 times.

As shown in Figure 2.(a), by sparsifying $\mathbf{w}$, EigenGP consistently outperforms EigenGP* for classification; this improvement is unsurprising as discussed in Section 5. Thus, we only included EigenGP

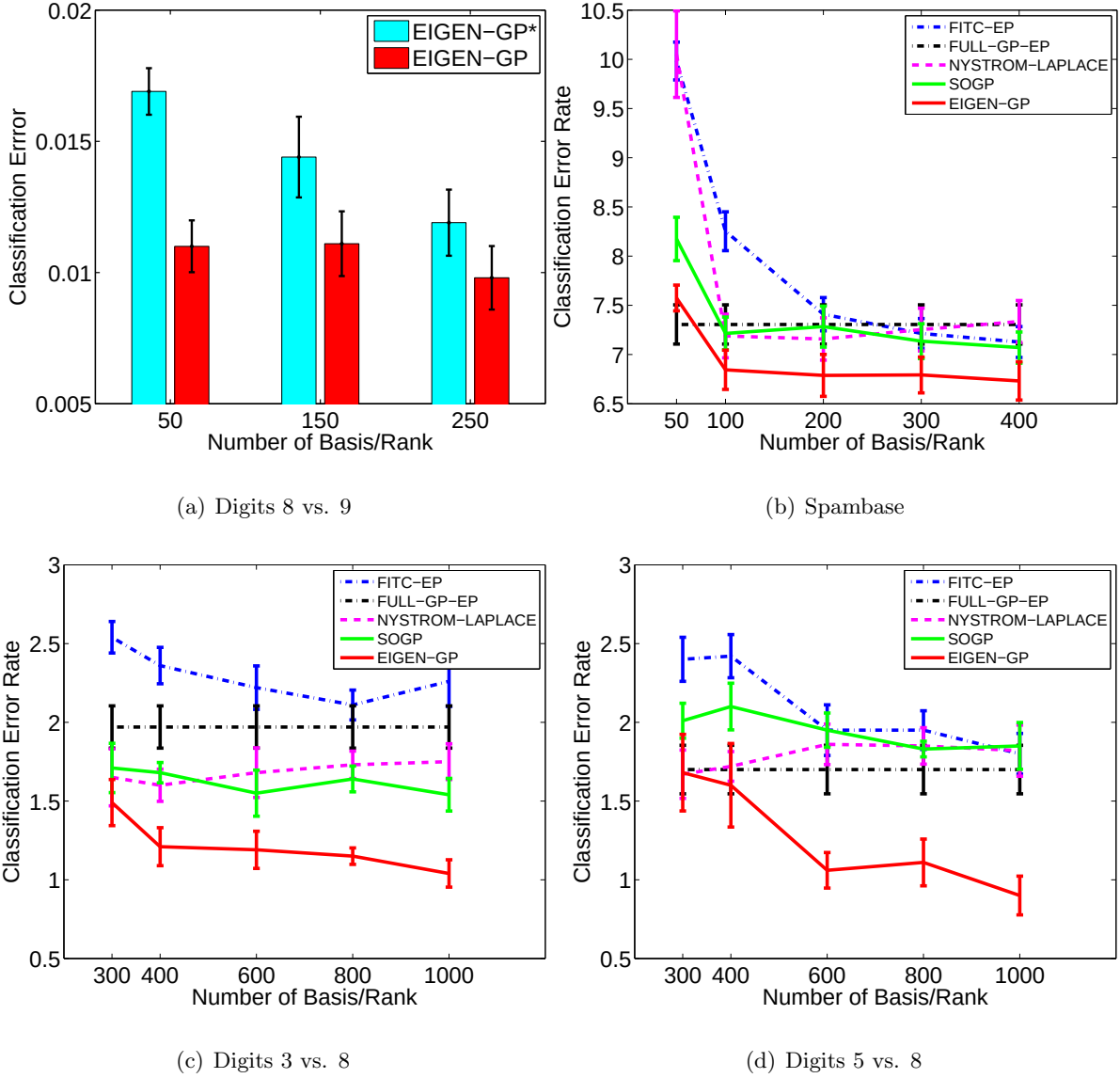


Figure 2: Classification results on Spambase and USPS. The results are averaged over 10 runs and the error bars represent the standard errors. Figure (a) shows that EigenGP with sparsification of $\mathbf{w}$ outperforms EigenGP* significantly on classification of digits 8 vs 9. Figures (b) to (d) demonstrate that EigenGP achieves significantly higher prediction accuracy than the alternative sparse GP methods with the same or less computational complexity.

in the comparison with alternative sparse GP algorithms. Figures 2.(b)-(d) demonstrate that, with the same or less computational cost, EigenGP outperforms the other sparse GP algorithms. And interestingly, EigenGP even outperforms the full GP. Two possible reasons are i) that by eigenfunctions selection, we can effectively remove labeling noise, just like PCA for noise reduction; and ii) that the clustering property of eigenfunctions helps improve classification accuracy.

7.3 Semi-supervised classification

For semi-supervised learning, we used all the labeled and unlabeled data points as basis points for EigenGP. We compared EigenGP with three well-known semi-supervised learning algorithms: the graph regularization (GR) approach (Zhou et al., 2004), the Laplacian support vector machine (LAPSVN) (Belkin et al., 2006), and the sparse-eigenfunction-bases approach (SEB) (Sinha & Belkin, 2010). We did not include

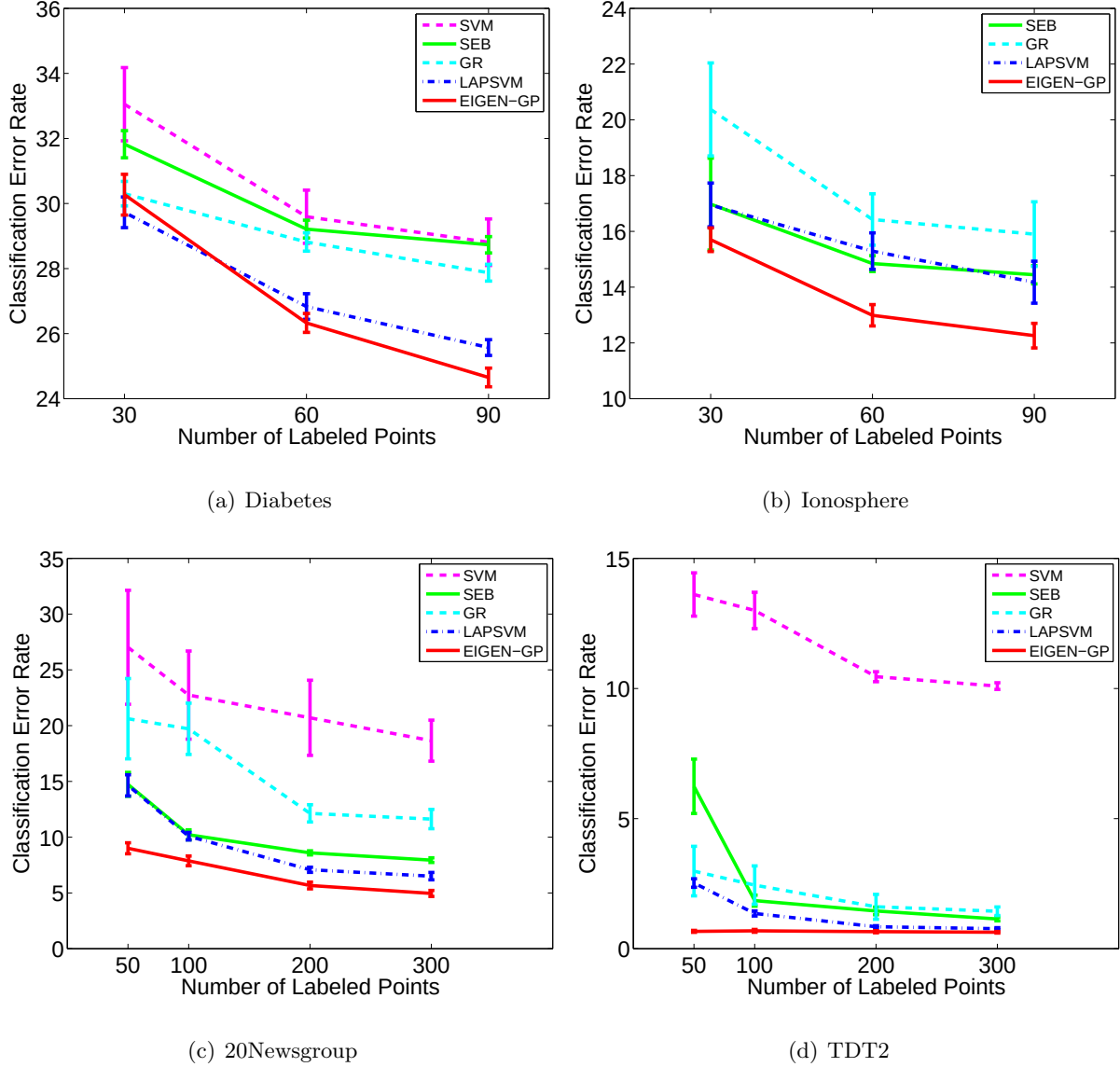


Figure 3: Semi-supervised classification results on Diabetes, Ionosphere, 20Newsgroups and TDT2. In our training, we varied l , the number of labeled data points, to examine the performance. The results are averaged over 10 runs and the error bars represent the standard errors. EigenGP outperforms the alternative methods for almost all the cases. The results of SVM on Ionosphere are not showed since their errors are much larger than others.

EigenGP* here since it is not designed for semi-supervised learning. We also include supervised SVM learning as a baseline. All these algorithms use the Gaussian kernel function with the same kernel width except GR.

We tested all the algorithms on two UCI dataset (Diabetes and Ionosphere, 768 and 351 data points respectively), and two text datasets (20Newsgroups and TDT2, 1976 and 3672 documents respectively). Each document is represented by first 1000 TF-IDF terms. We selected two categories—comp.sys.ibm.pc.hardware and rec.sport.baseball—from the 20Newsgroup dataset and two categories containing most instances from the TDT2 dataset. We randomly split the data 10 times and reported the averaged prediction error rates for each algorithm as well as the standard errors. Figure 3 shows that by selecting suitable eigenfunctions, EigenGP consistently outperformed the alternative approaches.

8 Conclusions

In this paper, we presented a new GP model, EigenGP, which has sparse eigenfunctions efficiently learned from data. Our experiments demonstrated EigenGP’s superior performance for regression, classification and semi-supervised classification on several benchmark datasets.

References

- Belkin, M., Niyogi, P., and Sindhwani, V. Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.
- Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006.
- Csató, L. and Opper, M. Sparse online Gaussian processes. *Neural Computation*, 14:641–668, March 2002.
- Faul, A. C. and Tipping, M. E. Analysis of sparse bayesian learning. In *Advances in Neural Information Processing Systems 14*, pp. 383–389. MIT Press, 2001.
- Lázaro-Gredilla, M., Quinonero-Candela, J., Rasmussen, C.E., and Figueiras-Vidal, A.R. Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research*, 11:1865–1881, 2010.
- MacKay, D. J. C. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.
- Marzouk, Y. M. and Najm, H. N. Dimensionality reduction and polynomial chaos acceleration of bayesian inference in inverse problems. *Journal of Computational Physics*, 228(6):1862 – 1902, 2009. ISSN 0021-9991.
- Minka, T. P. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001a.
- Minka, T. P. Expectation propagation for approximate Bayesian inference. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pp. 362–369, 2001b.
- Naish-Guzman, A. and Holden, S. The generalized fitc approximation. In Platt, J.C., Koller, D., Singer, Y., and Roweis, S. (eds.), *Advances in Neural Information Processing Systems 20*, pp. 1057–1064. MIT Press, Cambridge, MA, 2008.
- Qi, Y., Minka, T. P., and Picard, R. W. Predictive automatic relevance determination by expectation propagation. In *Proceedings of 21st International Conference on Machine Learning*, pp. 671–678, 2004.
- Qi, Y., Abdel-Gawad, A., and Minka, T. Sparse-posterior Gaussian processes for general likelihoods. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, 2010.
- Quiñonero-Candela, J. and Rasmussen, C. E. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1935–1959, 12 2005.
- Quiñonero-Candela, J., Snelson, E., and Williams, O. Sensible priors for sparse Bayesian learning. Technical Report MSR-TR-2007-121, Microsoft Research, 2007.
- Schenk, C. A. and Schuëller, G. I. *Uncertainty Assessment of Large Finite Element Systems*, volume 24. Springer, Aug 2005.
- Seeger, M. and Williams, C. K. I. Fast forward selection to speed up sparse Gaussian process regression. In *Workshop on Artificial Intelligence and Statistics 9*, 2003.

- Sinha, K. and Belkin, M. Semi-supervised learning using sparse eigenfunction bases. In *Advances in Neural Information Processing Systems 22*, pp. 1687–1695, 2010.
- Snelson, E. and Ghahramani, Z. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pp. 1257–1264. MIT press, 2006.
- Tipping, M. E. The relevance vector machine. In *Advances in Neural Information Processing Systems 12*, pp. 652–658. The MIT Press, 2000.
- Williams, C. K. I. and Barber, D. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12), 1998.
- Williams, C. K. I. and Seeger, M. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, volume 13, pp. 682–688, 2001.
- Yan, F. and Qi, Y. Sparse Gaussian process regression via ℓ_1 penalization. In *Proceedings of 27th International Conference on Machine Learning*, pp. 1183–1190, 2010.
- Zhou, D., Bousquet, O., T.N. Lal, J. Weston, and Schölkopf, B. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, 2004.

A Projection updates for regression and classification

A.1 Regression

Given the regression likelihood (2), we have

$$Z(m^{\setminus i}(x_i), v^{\setminus i}(x_i, x_i)) = \mathcal{N}(u_i | m^{\setminus i}(x_i), v_y + v^{\setminus i}(x_i, x_i)).$$

And the quantities in the projection step (14)

$$\frac{d \log Z}{dm^{\setminus i}(\mathbf{x}_i)} = \frac{y_i - m^{\setminus i}(\mathbf{x}_i)}{v_y + v^{\setminus i}(\mathbf{x}_i, \mathbf{x}_i)} \quad \frac{d^2 \log Z}{dm^{\setminus i}(\mathbf{x}_i)^2} = \frac{-1}{v_y + v^{\setminus i}(\mathbf{x}_i, \mathbf{x}_i)} \quad (\text{A.1})$$

This KL minimization does not introduce approximation error here. For regression, the EP algorithm are basically iterative updates to solve equations in Proposition 1 (avoiding an explicit matrix inversion in β).

A.2 Classification

Given the classification likelihood (3) where $\sigma(\cdot)$ is the step function, the quantities in the projection step are

$$z = \frac{m^{\setminus i}(\mathbf{x}_i)y_i}{\sqrt{v^{\setminus i}(\mathbf{x}_i, \mathbf{x}_i)}} \quad Z = \epsilon + (1 - 2\epsilon)c(z) \quad (\text{A.2})$$

$$\frac{d \log Z}{dm^{\setminus i}(\mathbf{x}_i)} = \gamma y_i \quad \frac{d^2 \log Z}{(dm^{\setminus i}(\mathbf{x}_i))^2} = -\frac{\gamma(m^{\setminus i}(x_i)y_i + v^{\setminus i}(\mathbf{x}_i)\gamma)}{v^{\setminus i}(\mathbf{x}_i)} \quad (\text{A.3})$$

where $\gamma = \frac{(1-2\epsilon)\mathcal{N}(z|0,1)}{Z\sqrt{v^{\setminus i}(\mathbf{x}_i)}}$ and $c(\cdot)$ is the standard Gaussian cumulative distribution function.

B Marginal likelihood and the algorithm summary

After the algorithm converges, the marginal likelihood is

$$p(\mathbf{y}, \mathbf{X} | \mathbf{w}) = \left(\prod_i s_i \right) \left(\prod_j w_j^{\frac{1}{2}} \right) |\mathbf{W} - \beta|^{\frac{1}{2}} \exp\left(\frac{1}{2}(\alpha^T(\mathbf{W} - \beta)^{-1}\alpha - \sum_i g^2(i)/\tau_i)\right) \quad (\text{B.1})$$

The EP inference algorithm is summarized in Algorithm 1.

Algorithm 1 Expectation Propagation inference

1. Initialize $q(f)$, g_i , and τ_i all to be 0.
 2. Loop until the change over all g_i , and τ_i is smaller than a threshold
 - Loop over all training data point $\mathbf{x}_i$
 - Deletion.** Compute $\alpha^{\setminus i}$ and $\beta^{\setminus i}$ for $q^{\setminus i}(f)$ via **Eqn** (17)
 - Projection.** Compute α and β for the posterior $\mathbf{q}(f)$ via **Eqn** (14)
 - Inclusion.** Update g_i , and τ_i for the message $\tilde{t}_i$ via **Eqn** (16)
-

C Details about low-rank Deterministic Training Conditional (DTC) Approximation

As mentioned in the main text, our model can be interpreted from the Deterministic Training Conditional (DTC) Approximation scheme. It replaces exact conditional models on *sparse* latent variables in DTC approximation with low-rank models on all training data points.

As shown by Quiñonero-Candela & Rasmussen (2005), the original GP model is equivalent to the following model:

$$p(\mathbf{u}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}}) \quad (\text{C.1})$$

$$p(\mathbf{f}_{\mathbf{x}}|\mathbf{u}) = \mathcal{N}(\mathbf{K}_{\mathbf{x},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}, \mathbf{K}_{\mathbf{x},\mathbf{x}} - \mathbf{K}_{\mathbf{x},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{x}}) \quad (\text{C.2})$$

$$p(\mathbf{f}_{\star}|\mathbf{u}) = \mathcal{N}(\mathbf{K}_{\star,\mathbf{u}}\mathbf{K}_{\mathbf{x},\mathbf{x}}^{-1}\mathbf{u}, \mathbf{K}_{\star,\star} - \mathbf{K}_{\star,\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\star}) \quad (\text{C.3})$$

The equivalence means that, after marginalizing out $\mathbf{u}$, the above model gives the same distribution on $\mathbf{f}_{\mathbf{x}}$ and $\mathbf{f}_{\star}$ as the exact GP model.

The DTC approximation basically approximates $p(\mathbf{f}_{\mathbf{x}}|\mathbf{u})$ by a deterministic function:

$$q_{DTC}(\mathbf{f}_{\mathbf{x}}|\mathbf{u}) = \mathcal{N}(\mathbf{K}_{\mathbf{x},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}, \mathbf{0}) \quad (\text{C.4})$$

Choosing a small set of the latent variables $\mathbf{u}$, DTC can speed up the GP inference.

By contrast, our model approximates $p(\mathbf{u})$, $p(\mathbf{f}_{\mathbf{x}}|\mathbf{u})$, and $q(\mathbf{f}_{\star}|\mathbf{u})$ by

$$q(\mathbf{u}) = \mathcal{N}(\mathbf{0}, \tilde{\mathbf{K}}_{\mathbf{u},\mathbf{u}}) \quad (\text{C.5})$$

$$q(\mathbf{f}_{\mathbf{x}}|\mathbf{u}) = \mathcal{N}(\tilde{\mathbf{K}}_{\mathbf{x},\mathbf{u}}\tilde{\mathbf{K}}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}, \mathbf{0}) \quad (\text{C.6})$$

$$q(\mathbf{f}_{\star}|\mathbf{u}) = \mathcal{N}(\tilde{\mathbf{K}}_{\star,\mathbf{u}}\tilde{\mathbf{K}}_{\mathbf{x},\mathbf{x}}^{-1}\mathbf{u}, \mathbf{K}_{\star,\star} - \tilde{\mathbf{K}}_{\star,\mathbf{u}}\tilde{\mathbf{K}}_{\mathbf{u},\mathbf{u}}^{-1}\tilde{\mathbf{K}}_{\mathbf{u},\star}) = \mathcal{N}(\tilde{\mathbf{K}}_{\star,\mathbf{u}}\tilde{\mathbf{K}}_{\mathbf{x},\mathbf{x}}^{-1}\mathbf{u}, \mathbf{K}_{\star,\star} - \tilde{\mathbf{K}}_{\star,\star}). \quad (\text{C.7})$$

Note that $\tilde{\mathbf{K}}$ and $\tilde{\mathbf{K}}_{\mathbf{x},\mathbf{u}}$ are based on the new low-rank covariance function of our model (see (7)). This model gives the same posterior of $\mathbf{f}_{\mathbf{x}}$ as (9) and the same predictive distribution of $\mathbf{f}_{\star}$ as (11).

Note that for our model, we do *not* select a subset of the training points as $\mathbf{u}$ – as the DTC approximation (Seeger & Williams, 2003) – to save computation. Instead, we simply let $\mathbf{u}$ be all the training points, but use low-rank approximations to speed up the inference.

D Details about model marginal likelihood maximization

We maximize the model marginal likelihood (B.1) via automatic relevance determination (ARD). In particular, we use an EP-EM approach. In the M step, we compute $\beta_{j,j}$ and α_j via EP based on the old $\mathbf{w}$; in the E step, we update $\mathbf{w}$:

$$w_j^{new} = w_j - \beta_{j,j} + \alpha_j^2. \quad (\text{D.1})$$

First, for classification, use EP to map y_i into g_i via (17). For regression, we have $y_i = g_i$.

Then, we define $\tilde{f} = \Phi_{\mathbf{x}}^+ \mathbf{f}_{\mathbf{x}}$, where $\Phi_{\mathbf{x}}^+$ is the pseudo-inverse of $\Phi_{\mathbf{x}}$ such that $\Phi \Phi_D^+ = \mathbf{I}$. Then we obtain the following model

$$p(\mathbf{g}, \tilde{f}) = N(\tilde{f}|\mathbf{0}, \mathbf{w}) \prod_i p(g_i|\Phi_{\mathbf{x}}\tilde{f}_i).$$

Once we marginalize out $\tilde{f}$, we obtain the same marginal likelihood as from the original model (equation (8)).

Given (9), we obtain

$$E(\tilde{f}_i) = \Phi_{\mathbf{x}}\Phi_{\mathbf{x}}^+\alpha = \alpha \quad (\text{D.2})$$

$$E(\tilde{f}_i^2) = w_j - \beta_{j,j} + \alpha_j^2 \quad (\text{D.3})$$

Setting the gradient of $E(\ln p(\mathbf{g}, \tilde{f}))$ gives us

$$w_j^{new} = w_j - \beta_{j,j} + \alpha_j^2. \quad (\text{D.4})$$

We can also use the active-set method proposed by Faul & Tipping (2001) to conduct the ARD. This method was proposed to estimate sparse features for the regression task. Its central idea is to recursively re-estimate elements of $\mathbf{w}$ in the current active set while holding the other elements unchanged. Since EP essentially maps nonlinear data likelihood into linear Gaussian factors, we can adopt this active-set method for our EP inference. Started from a sparse initial set, we update $\mathbf{w}$ recursively. Plugging the sufficient statistics for $\mathbf{w}$, we obtain the updates almost identical to those in (Qi et al., 2004). During each iteration, EP updates α and β based on a small number of nonzero elements in $\mathbf{w}$; this number ($\|\mathbf{w}\|_0$) is typically much smaller than L . Because we only use a small number of iterations, the overall computational cost is $O(r^2N)$ where r is the biggest $\|\mathbf{w}\|_0$ we encounter along the iterations.

Finally, we use the predictive ARD (Qi et al., 2004) to avoid overfitting of marginal likelihood maximization. Specifically, we explore a leave-one-out estimate conveniently obtained from the EP inference without any extra cost: For the classification task, we have $e_{loo} = \frac{1}{N} \sum_i \text{Step}(\mathbf{k}(\mathbf{x}_i)\mathbf{U}\alpha^{\setminus i}y(i))$, where $\text{Step}(a) = 1$ if $a > 0$ and $\text{Step}(a) = 0$ otherwise. For the regression task, we have $e_{loo} = \frac{1}{N} \sum_i (\mathbf{k}(\mathbf{x}_i)\mathbf{U}\alpha^{\setminus i} - y(i))^2$. Along the evidence maximization iterations, we choose the hyperparameter $\mathbf{w}$ that gives the smallest leave-one-out error estimate.