Manifold Gaussian Processes for Regression

Roberto Calandra*, Jan Peters*†, Carl Edward Rasmussen‡ and Marc Peter Deisenroth§
*Intelligent Autonomous Systems Lab, Technische Universität Darmstadt, Germany
†Max Planck Institute for Intelligent Systems, Tübingen, Germany
‡Department of Engineering, University of Cambridge, United Kingdom
§Department of Computing, Imperial College London, United Kingdom

***Abstract*—Off-the-shelf Gaussian Process (GP) covariance functions encode smoothness assumptions on the structure of the function to be modeled. To model complex and non-differentiable functions, these smoothness assumptions are often too restrictive. One way to alleviate this limitation is to find a different representation of the data by introducing a feature space. This feature space is often learned in an unsupervised way, which might lead to data representations that are not useful for the overall regression task. In this paper, we propose Manifold Gaussian Processes, a novel supervised method that jointly learns a transformation of the data into a feature space and a GP regression from the feature space to observed space. The Manifold GP is a full GP and allows to learn data representations, which are useful for the overall regression task. As a proof-of-concept, we evaluate our approach on complex non-smooth functions where standard GPs perform poorly, such as step functions and robotics tasks with contacts.**

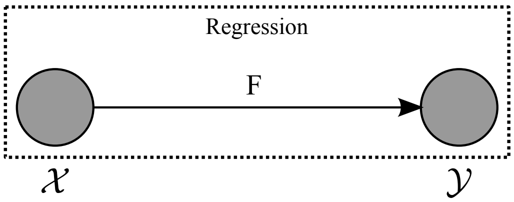
1. Introduction

Gaussian Processes (GPs) are a powerful state-of-the-art nonparametric Bayesian regression method. The covariance function of a GP implicitly encodes high-level assumptions about the underlying function to be modeled, e.g., smoothness or periodicity. Hence, the choice of a suitable covariance function for a specific data set is crucial. A standard choice is the squared exponential (Gaussian) covariance function, which implies assumptions, such as smoothness and stationarity. Although the squared exponential can be applied to a great range of problems, generic covariance functions may also be inadequate to model a variety of functions where the common smoothness assumptions are violated, such as ground contacts in robot locomotion.

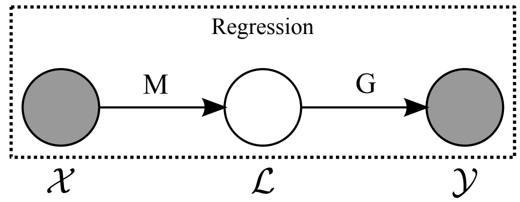
Two common approaches can overcome the limitations of standard covariance functions. The first approach combines multiple standard covariance functions to form a new covariance function (Rasmussen and Williams, 2006; Wilson and Adams, 2013; Duvenaud et al., 2013). This approach allows to automatically design relatively complex covariance functions. However, the resulting covariance function is still limited by the properties of the combined covariance functions. The second approach is based on data transformation (or pre-processing), after which the data can be modeled with standard covariance functions. One way to implement this second approach is to transform the output space as in the Warped GP (Snelson et al., 2004). An alternative is to transform the input space. Transforming the input space and subsequently applying GP regression with a standard covariance function is equivalent to GP regression with a new covariance function that explicitly depends on the transformation (MacKay, 1998). One example is the stationary periodic covariance function (MacKay, 1998; HajiGhassemi and Deisenroth, 2014), which effectively is the squared exponential covariance function applied to a complex representation of the input variables. Common transformations of the inputs include data normalization and dimensionality reduction, e.g., PCA (Pearson, 1901). Generally, these input transformations are good heuristics or optimize an unsupervised objective. However, they may be suboptimal for the overall regression task.

In this paper, we propose the *Manifold Gaussian Process* (mGP), which is based on MacKay's ideas to devise flexible covariance functions for GPs. Our GP model is equivalent to jointly learning a data transformation into a feature space followed by a GP regression with off-the-shelf covariance functions from feature space to observed space. The model profits from standard GP properties, such as a straightforward incorporation of a prior mean function and a faithful representation of model uncertainty.

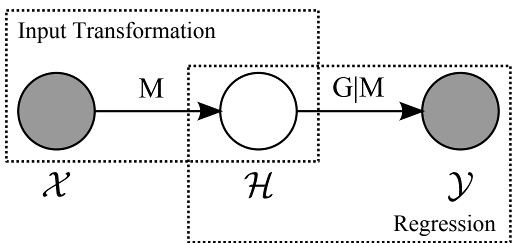
Multiple related approaches in the literature attempt joint supervised learning of features and regression/classification. In Salakhutdinov and Hinton (2007), pre-training of the input transformation makes use of computationally expensive unsupervised learning that requires thousands of data points. Snoek et al. (2012) combined both unsupervised and supervised objectives for the optimization of an input transformation in a classification task. Unlike these approaches, the mGP is motivated by the need of a stronger (i.e., supervised) guidance to discover suitable transformations for regression problems, while remaining within a Bayesian framework. Damianou and Lawrence (2013) proposed the Deep GP, which stacks multiple layers of GP-LVMs, similarly to a neural network. This model exhibits great flexibility in supervised and unsupervised settings, but the resulting model is not a full GP. Snelson and Ghahramani (2006) proposed a supervised dimensionality reduction by jointly learning a liner transformation of the input and a



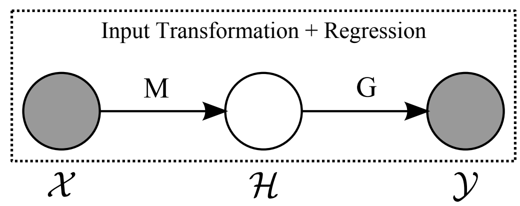
(a) Supervised learning of a regression function.



(b) Supervised learning integrating out a latent space is *intractable*.



(c) Unsupervisedly learned input transformation M followed by a conditional regression $G|M$.



(d) Manifold GP: joint supervised learning of the input transformation M and the regression task G.

Figure 1: Different regression settings to learn the function $F : \mathcal{X} \rightarrow \mathcal{Y}$. (a) Standard supervised regression. (b) Regression with an auxiliary latent space $\mathcal{L}$ that allows to simplify the task. In a full Bayesian framework, $\mathcal{L}$ would be integrated out, which is analytically intractable. (c) Decomposition of the overall regression task F into discovering a feature space $\mathcal{H}$ using the map M and a subsequent (conditional) regression $G|M$. (d) Our mGP learns the mappings G and M *jointly*.

GP. Snoek et al. (2014) transformed the input data using a Beta distribution whose parameters were learned jointly with the GP. However, the purpose of this transformation is to account for skewness in the data, while mGP allows for a more general class of transformations.

2. Manifold Gaussian Processes

In the following, we review methods for regression, which may use latent or feature spaces. Then, we provide a brief introduction to Gaussian Process regression. Finally, we introduce the Manifold Gaussian Processes, our novel approach to jointly learning a regression model and a suitable feature representation of the data.

2.1. Regression with Learned Features

We assume N training inputs $\boldsymbol{x}_n \in \mathcal{X} \subseteq \mathbb{R}^D$ and respective outputs $y_n \in \mathcal{Y} \subseteq \mathbb{R}$, where $y_n = F(\boldsymbol{x}_n) + w$, $w \sim \mathcal{N}\big(0, \sigma_w^2\big)$, $n = 1, \ldots, N$. The training data is denoted by $\boldsymbol{X}$ and $\boldsymbol{Y}$ for the inputs and targets, respectively. We consider the task of learning a regression function $F : \mathcal{X} \rightarrow \mathcal{Y}$. The corresponding setting is given in Figure 1a. Discovering the regression function F is often challenging for nonlinear functions. A typical way to simplify and distribute the complexity of the regression problem is to introduce an auxiliary latent space $\mathcal{L}$. The function F can then be decomposed into $F = G \circ M$, where $M : \mathcal{X} \rightarrow \mathcal{L}$ and $G : \mathcal{L} \rightarrow \mathcal{Y}$, as shown in Figure 1b. In a full Bayesian framework, the latent space $\mathcal{L}$ is integrated out to solve the regression task F, which is often analytically unfeasible (Schmidt and O'Hagan, 2003).

A common approximation to the full Bayesian framework is to introduce a deterministic feature space $\mathcal{H}$, and to find the mappings M and G in two consecutive steps. First, M is determined by means of unsupervised feature learning. Second, the regression G is learned supervisedly as a conditional model $G|M$, see Figure 1c. The use of this feature space can reduce the complexity of the learning problem. For example, for complicated non-linear functions a higher-dimensional (overcomplete) representation $\mathcal{H}$ allows learning a simpler mapping $G : \mathcal{H} \rightarrow \mathcal{Y}$. For high-dimensional inputs, the data often lies on a lower-dimensional manifold $\mathcal{H}$, e.g., due to non-discriminant or strongly correlated covariates. The lower-dimensional feature space $\mathcal{H}$ reduces the effect of the curse of dimensionality. In this paper, we focus on modeling complex functions with a relatively low-dimensional input space, which, nonetheless, cannot be well modeled by off-the-shelf GP covariance functions.

Typically, unsupervised feature learning methods determine the mapping M by optimizing an unsupervised objective, independent from the objective of the overall regression F. Examples of such unsupervised objectives are the minimization of the input reconstruction error (auto-encoders (Vincent et al., 2008)), maximization of the variance (PCA (Pearson, 1901)), maximization of the statistical independence (ICA (Hyvärinen and Oja, 2000)), or the preservation of the distances between data (isomap (Tenenbaum et al., 2000) or LLE (Roweis and Saul, 2000)). In the context of regression, an unsupervised approach for feature learning can be insufficient as the learned data representation $\mathcal{H}$ might not suit the overall regression task F (Wahlström et al., 2015): Unsupervised and supervised learning optimize different objectives, which do not necessarily match, e.g., minimizing the reconstruction error as unsupervised objective and maximizing the marginal likelihood as supervised objective. An approach where feature learning is performed in a supervised manner can instead guide learning the feature mapping M toward representations that are useful for the overall regression $F = G \circ M$. This intuition is the key insight of our Manifold Gaussian Processes, where the feature mapping M and the GP G are learned jointly using the same supervised objective as depicted in Figure 1d.

2.2. Gaussian Process Regression

GPs are a state-of-the-art probabilistic non-parametric regression method (Rasmussen and Williams, 2006). Such a GP is a distribution over functions

$$F \sim \mathcal{GP}(m, k) \tag{1}$$

and fully defined by a mean function m (in our case $m \equiv \mathbf{0}$) and a covariance function k. The GP predictive distribution at a test input $\boldsymbol{x}_*$ is given by

$$p(F(\boldsymbol{x}_*)|\mathbb{D}, \boldsymbol{x}_*) = \mathcal{N}\big(\mu(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*)\big), \tag{2}$$

$$\mu(\boldsymbol{x}_*) = \boldsymbol{k}_*^T(\boldsymbol{K} + \sigma_w^2\boldsymbol{I})^{-1}\boldsymbol{Y}, \tag{3}$$

$$\sigma^2(\boldsymbol{x}_*) = k_{**} - \boldsymbol{k}_*^T(\boldsymbol{K} + \sigma_w^2\boldsymbol{I})^{-1}\boldsymbol{k}_*, \tag{4}$$

where $\mathbb{D} = \{\boldsymbol{X}, \boldsymbol{Y}\}$ is the training data, $\boldsymbol{K}$ is the kernel matrix with $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $k_{**} = k(\boldsymbol{x}_*, \boldsymbol{x}_*)$, $\boldsymbol{k}_* = k(\boldsymbol{X}, \boldsymbol{x}_*)$ and σ_w^2 is the measurement noise variance. In our experiments, we use different covariance functions k. Specifically, we use the squared exponential covariance function with Automatic Relevance Determination (ARD)

$$k_{\text{SE}}(\boldsymbol{x}_p, \boldsymbol{x}_q) = \sigma_f^2 \exp\left(-\tfrac{1}{2}(\boldsymbol{x}_p - \boldsymbol{x}_q)^T\boldsymbol{\Lambda}^{-1}(\boldsymbol{x}_p - \boldsymbol{x}_q)\right), \tag{5}$$

with $\boldsymbol{\Lambda} = \text{diag}([l_1^2, ..., l_D^2])$, where l_i are the characteristic length-scales, and σ_f^2 is the variance of the latent function F. Furthermore, we use the neural network covariance function

$$k_{\text{NN}}(\boldsymbol{x}_p, \boldsymbol{x}_q) = \sigma_f^2 \sin^{-1}\left(\frac{\boldsymbol{x}_p^T\boldsymbol{P}\boldsymbol{x}_q}{\sqrt{(1+\boldsymbol{x}_p^T\boldsymbol{P}\boldsymbol{x}_p)(1+\boldsymbol{x}_q^T\boldsymbol{P}\boldsymbol{x}_q)}}\right), \tag{6}$$

where $\boldsymbol{P}$ is a weight matrix. Each covariance function possesses various hyperparameters $\boldsymbol{\theta}$ to be selected. This selection is performed by minimizing the Negative Log Marginal Likelihood (NLML)

$$\begin{aligned}\text{NLML}(\boldsymbol{\theta}) &= -\log p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta}) \qquad (7)\\ &\doteq \tfrac{1}{2}\boldsymbol{Y}^T(\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma_w^2\boldsymbol{I})^{-1}\boldsymbol{Y} + \tfrac{1}{2}\log|\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma_w^2\boldsymbol{I}|\end{aligned}$$

Using the chain-rule, the corresponding gradient can be computed analytically as

$$\frac{\partial \text{NLML}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial \text{NLML}(\boldsymbol{\theta})}{\partial \boldsymbol{K}_{\boldsymbol{\theta}}}\frac{\partial \boldsymbol{K}_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}}, \tag{8}$$

which allows us to optimize the hyperparameters using Quasi-Newton optimization, e.g., L-BFGS (Liu and Nocedal, 1989).

2.3. Manifold Gaussian Processes

In this section, we describe the mGP model and its parameters $\boldsymbol{\theta}_{\text{mGP}}$ itself, and relate it to standard GP regression. Furthermore, we detail training and prediction with the mGP.

2.3.1. Model. As shown in Figure 1d, the mGP considers the overall regression as a composition of functions

$$F = G \circ M. \tag{9}$$

The two functions M and G are learned jointly to accomplish the overall regression objective function, i.e., the marginal likelihood in Equation (7). In this paper, we assume that M is a deterministic, parametrized function that maps the input space $\mathcal{X}$ into the feature space $\mathcal{H} \subseteq \mathbb{R}^Q$, which serves as the domain for the GP regression $G : \mathcal{H} \to \mathcal{Y}$. Performing this transformation of the input data corresponds to training a GP G having $\boldsymbol{H} = M(\boldsymbol{X})$ as inputs. Therefore, the mGP is equivalent to a GP for a function $F : \mathcal{X} \to \mathcal{Y}$ with a covariance function $\tilde{k}$ defined as

$$\tilde{k}(\boldsymbol{x}_p, \boldsymbol{x}_q) = k\left(M(\boldsymbol{x}_p), M(\boldsymbol{x}_q)\right), \tag{10}$$

i.e., the kernel operates on the Q-dimensional feature space $\mathcal{H} = M(\mathcal{X})$. According to MacKay (1998), a function defined as in Equation (10) is a valid covariance function and, therefore, the mGP is a valid GP.

The predictive distribution for the mGP at a test input $\boldsymbol{x}_*$ can then be derived from the predictive distribution of a standard GP in Equation (2) as

$$\begin{aligned}p(F(\boldsymbol{x}_*)|\mathbb{D}, \boldsymbol{x}_*) &= p((G \circ M)(\boldsymbol{x}_*)|\mathbb{D}, \boldsymbol{x}_*)\\ &= \mathcal{N}\big(\mu(M(\boldsymbol{x}_*)), \sigma^2(M(\boldsymbol{x}_*))\big), \qquad (11)\end{aligned}$$

$$\mu(M(\boldsymbol{x}_*)) = \tilde{\boldsymbol{k}}_*^T(\tilde{\boldsymbol{K}} + \sigma_w^2\boldsymbol{I})^{-1}\boldsymbol{Y}, \tag{12}$$

$$\sigma^2(M(\boldsymbol{x}_*)) = \tilde{k}_{**} - \tilde{\boldsymbol{k}}_*^T(\tilde{\boldsymbol{K}} + \sigma_w^2\boldsymbol{I})^{-1}\tilde{\boldsymbol{k}}_*, \tag{13}$$

where $\tilde{\boldsymbol{K}}$ is the kernel matrix constructed as $\tilde{K}_{ij} = \tilde{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $\tilde{k}_{**} = \tilde{k}(\boldsymbol{x}_*, \boldsymbol{x}_*)$, $\tilde{\boldsymbol{k}}_* = \tilde{k}(\boldsymbol{X}, \boldsymbol{x}_*)$, and $\tilde{k}$ is the covariance function from Equation (10). In our experiments, we used the squared exponential covariance function from Equation (5) for the kernel k in Equation (10).

2.3.2. Training. We train the mGP by jointly optimizing the parameters $\boldsymbol{\theta}_M$ of the transformation M and the GP hyperparameters $\boldsymbol{\theta}_G$. For learning the parameters $\boldsymbol{\theta}_{\text{mGP}} = [\boldsymbol{\theta}_M, \boldsymbol{\theta}_G]$, we minimize the NLML as in the standard GP regression. Considering the composition of the mapping $F = G \circ M$, the NLML becomes

$$\begin{aligned}&\text{NLML}(\boldsymbol{\theta}_{\text{mGP}}) = -\log p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{mGP}})\\ &\doteq \tfrac{1}{2}\boldsymbol{Y}^T(\tilde{\boldsymbol{K}}_{\boldsymbol{\theta}_{\text{mGP}}} + \sigma_w^2\boldsymbol{I})^{-1}\boldsymbol{Y} + \tfrac{1}{2}\log|\tilde{\boldsymbol{K}}_{\boldsymbol{\theta}_{\text{mGP}}} + \sigma_w^2\boldsymbol{I}|.\end{aligned}$$

Note that $\tilde{\boldsymbol{K}}_{\boldsymbol{\theta}_{\text{mGP}}}$ depends on both $\boldsymbol{\theta}_G$ and $\boldsymbol{\theta}_M$, unlike $\boldsymbol{K}_{\boldsymbol{\theta}}$ from Equation (7), which depends only on $\boldsymbol{\theta}_G$. The analytic gradients $\partial\text{NLML}/\partial\boldsymbol{\theta}_G$ of the objective in Equation (14) with respect to the parameters $\boldsymbol{\theta}_G$ are computed as in the standard GP, i.e.,

$$\frac{\partial \text{NLML}(\boldsymbol{\theta}_{\text{mGP}})}{\partial \boldsymbol{\theta}_G} = \frac{\partial \text{NLML}(\boldsymbol{\theta}_{\text{mGP}})}{\partial \boldsymbol{K}_{\boldsymbol{\theta}_{\text{mGP}}}} \frac{\partial \boldsymbol{K}_{\boldsymbol{\theta}_{\text{mGP}}}}{\partial \boldsymbol{\theta}_G} . \tag{14}$$

The gradients of the parameters $\boldsymbol{\theta}_M$ of the feature mapping are computed by applying the chain-rule

$$\frac{\partial \text{NLML}(\boldsymbol{\theta}_{\text{mGP}})}{\partial \boldsymbol{\theta}_M} = \frac{\partial \text{NLML}(\boldsymbol{\theta}_{\text{mGP}})}{\partial \boldsymbol{K}_{\boldsymbol{\theta}_{\text{mGP}}}} \frac{\partial \boldsymbol{K}_{\boldsymbol{\theta}_{\text{mGP}}}}{\partial \boldsymbol{H}} \frac{\partial \boldsymbol{H}}{\partial \boldsymbol{\theta}_M} , \tag{15}$$

where only $\partial \boldsymbol{H}/\partial \boldsymbol{\theta}_M$ depends on the chosen input transformation M, while $\partial \boldsymbol{K}_{\boldsymbol{\theta}_{\text{mGP}}}/\partial \boldsymbol{H}$ is the gradient of the kernel matrix with respect to the Q-dimensional GP training inputs $\boldsymbol{H} = M(\boldsymbol{X})$. Similarly to standard GP, the parameters $\boldsymbol{\theta}_{\text{mGP}}$ in the mGP can be obtained using off-the-shelf optimization methods.

2.3.3. Input Transformation. Our approach can use any deterministic parametric data transformation M. We focus on multi-layer neural networks and define their structure as $[q_1 - \ldots - q_l]$ where l is the number of layers, and q_i is the number of neurons of the i^{th} layer. Each layer $i = 1, \ldots, l$ of the neural network performs the transformation

$$\mathbb{T}_i(\boldsymbol{Z}) = \sigma\left(\boldsymbol{W}_i \boldsymbol{Z} + \boldsymbol{B}_i\right) , \tag{16}$$

where $\boldsymbol{Z}$ is the input of the layer, σ is the transfer function, and $\boldsymbol{W}_i$ and $\boldsymbol{B}_i$ are the weights and the bias of the layer, respectively. Therefore, the input transformation M of Equation (9) is $M(\boldsymbol{X}) = (\mathbb{T}_l \circ \ldots \circ \mathbb{T}_1)(\boldsymbol{X})$. The parameters $\boldsymbol{\theta}_M$ of the neural network M are the weights and biases of the whole network, so that $\boldsymbol{\theta}_M = [\boldsymbol{W}_1, \boldsymbol{B}_1, \ldots, \boldsymbol{W}_l, \boldsymbol{B}_l]$. The gradients $\partial \boldsymbol{H}/\partial \boldsymbol{\theta}_M$ in Equation (15) are computed by repeated application of the chain-rule (backpropagation).

3. Experimental Results

To demonstrate the efficiency of our proposed approach, we apply the mGP to challenging benchmark problems and a real-world regression task. First, we demonstrate that mGPs can be successfully applied to learning discontinuous functions, a daunting undertaking with an off-the-shelf covariance function, due to its underlying smoothness assumptions. Second, we evaluate mGPs on a function with multiple natural length-scales. Third, we assess mGPs on real data from a walking bipedal robot. The locomotion data set is highly challenging due to ground contacts, which cause the regression function to violate standard smoothness assumptions.

To evaluate the goodness of the different models on the training set, we consider the NLML previously introduced in Equation (7) and (14). Additionally, for the test set, we make use of the Negative Log Predictive Probability (NLPP)

$$-\log p(\boldsymbol{y} = \boldsymbol{y}_* | \boldsymbol{X}, \boldsymbol{x}_*, \boldsymbol{Y}, \boldsymbol{\theta}) , \tag{17}$$

where the $\boldsymbol{y}_*$ is the test target for the input $\boldsymbol{x}_*$ as computed for the standard GP in Equation (2) and (11) for the mGP model.

We compare our mGP approach with GPs using the SE-ARD and NN covariance functions, which implement the model in Figure 1a. Moreover, we evaluate two unsupervised feature extraction methods, Random Embeddings and PCA, followed by a GP SE-ARD, which implements the model in Figure 1c.[1] For the model in Figure 1d, we consider two variants of mGP with the log-sigmoid $\sigma(x) = 1/(1 + e^{-x})$ and the identity $\sigma(x) = x$ transfer functions. These two transfer functions lead to a non-linear and a linear transformation M, respectively.

3.1. Step Function

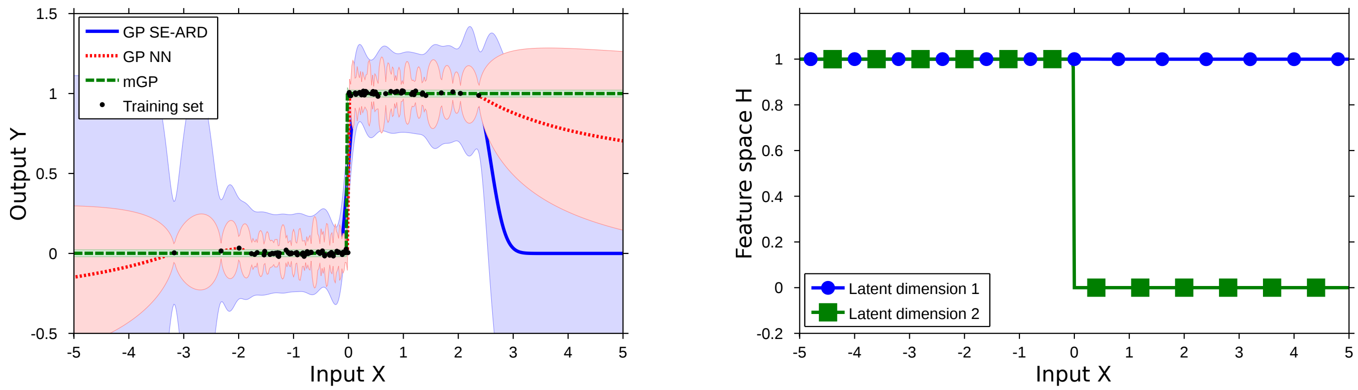
In the following, we consider the step function

$$y = F(x) + w , \quad w \sim \mathcal{N}\big(0, 0.01^2\big) ,$$
$$F(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases} . \tag{18}$$

For training, 100 inputs points are sampled from $\mathcal{N}\big(0, 1\big)$ while the test set is composed of 500 data points uniformly distributed between -5 and $+5$. The mGP uses a multi-layer neural network of [1-6-2] neurons (such that the feature space $\mathcal{H} \subseteq \mathbb{R}^2$) for the mapping M and a standard SE-ARD covariance function for the GP regression G. Values of the NLML per data point for the training and NLPP per data point for the test set are reported in Table 1. In both performance measures, the mGP using a non-linear transformation outperforms the other models. An example of the resulting predictive mean and the 95% confidence bounds for three models is shown in Figure 2a. Due to the implicit assumptions employed by the SE-ARD and NN covariance functions on the mapping F, neither of them appropriately captures the discontinuous nature of the underlying function or its correct noise level. The GP model applied to the random embedding and mGP (identity) perform similar to a standard GP with SE-ARD covariance function as their linear transformations do not substantially change the function. Compared to these models, the mGP (log-sigmoid) captures the discontinuities of the function better, thanks to its non-linear transformation, while the uncertainty remains small over the whole function's domain.

Note that the mGP still assumes smoothness in the regression G, which requires the transformation M to take care of the discontinuity. This effect can be observed in Figure 2b, where an example of the 2D learned feature space $\mathcal{H}$ is shown. The discontinuity is already encoded in the feature space. Hence, it is easier for the GP to learn the mapping G. Learning the discontinuity in the feature space is a direct result from jointly training M and G as feature learning is embedded in the overall regression F.

1. The random embedding is computed as the transformation $\boldsymbol{H} = \boldsymbol{\alpha}\boldsymbol{X}$, where the elements of $\boldsymbol{\alpha}$ are randomly sampled from a normal distribution.



(a) GP prediction.

(b) Learned mapping M using mGP (log-sigmoid).

Figure 2: **Step Function**: (a) Predictive mean and 95% confidence bounds for a GP with SE-ARD covariance function (blue solid), a GP with NN covariance function (red dotted) and a log-sigmoid mGP (green dashed) on the step function of Equation (18). The discontinuity is captured better by an mGP than by a regular GP with either SE-ARD or NN covariance functions. (b) The 2D feature space $\mathcal{H}$ discovered by the non-linear mapping M as a function of the input $\mathcal{X}$. The discontinuity of the modeled function is already captured by the non-linear mapping M. Hence, the mapping from feature space $\mathcal{H}$ to the output $\mathcal{Y}$ is smooth and can be easily managed by the GP.

Table 1: **Step Function**: Negative Log Marginal Likelihood (NLML) and Negative Log Predictive Probability (NLPP) per data point for the step function of Equation (18). The mGP (log-sigmoid) captures the nature of the underlying function better than a standard GP in both the training and test sets.

Method	Training set		Test set	
	NLML	RMSE	NLPP	RMSE
GP SE-ARD	-0.68	1.00×10^{-2}	$+0.50 \times 10^{-3}$	0.58
GP NN	-1.49	$\mathbf{0.57 \times 10^{-2}}$	$+0.02 \times 10^{-3}$	0.14
mGP (log-sigmoid)	$\mathbf{-2.84}$	1.06×10^{-2}	$\mathbf{-6.34 \times 10^{-3}}$	**0.02**
mGP (identity)	-0.68	1.00×10^{-2}	$+0.50 \times 10^{-3}$	0.58
RandEmb + GP SE-ARD	-0.77	5.26×10^{-2}	$+0.51 \times 10^{-3}$	0.52

3.2. Multiple Length-Scales

In the following, we demonstrate that the mGP can be used to model functions that possess multiple intrinsic length-scales. For this purpose, we rotate the function

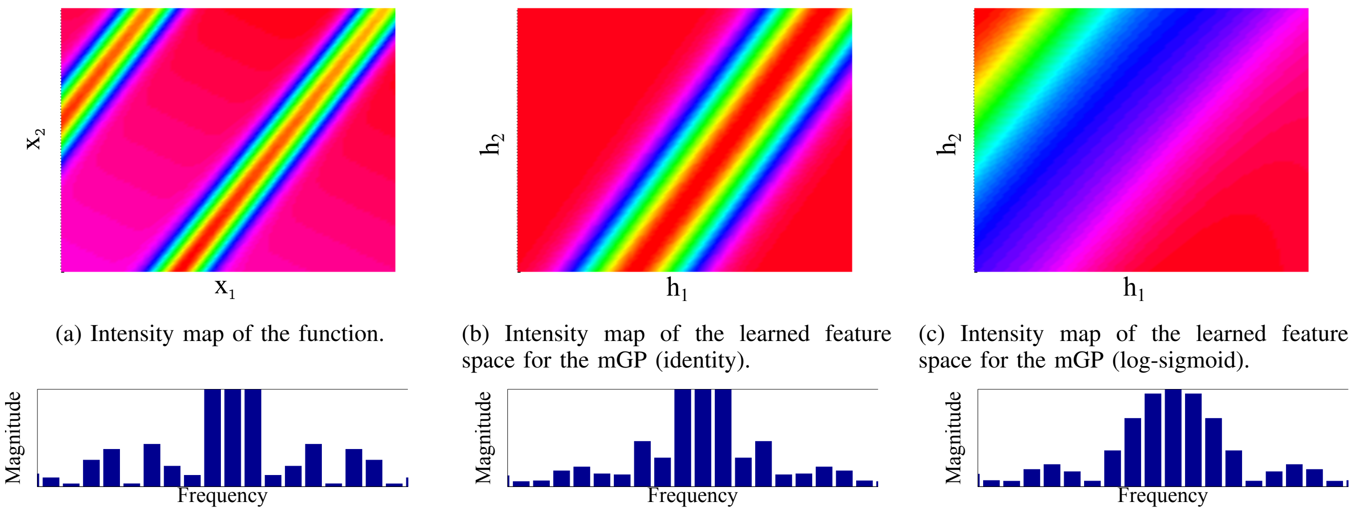
$$y = 1 - \mathcal{N}\left(x_2|3, 0.5^2\right) - \mathcal{N}\left(x_2| - 3, 0.5^2\right) + \tfrac{x_1}{100} \qquad (19)$$

anti-clockwise by $45°$. The intensity map of the resulting function is shown in Figure 3a. By itself (i.e., without rotating the function), Equation (19) is a fairly simple function. However, when rotated, the correlation between the covariates substantially complicates modeling. If we consider a horizontal slice of the rotated function, we can see how different spectral frequencies are present in the function, see Figure 3d. The presence of different frequencies is problematic for covariance functions, such as the SE-ARD, which assume a single frequency. When learning the hyperparameters, the length-scale needs to trade off different frequencies. Typically, the hyperparameter optimization gives a preference to shorter length-scales. However, such a trade-off greatly reduces the generalization capabilities of the model.

We compare the performances of a standard GP using SE-ARD and NN covariance functions and random embeddings followed by a GP using the SE-ARD covariance function, and our proposed mGP. We train these models with 400 data points, randomly sampled from a uniform distribution in the intervals $x_1 = [0, 10]$, $x_2 = [0, 10]$. As a test set we use 2500 data points distributed on a regular grid in the same intervals. For the mGP with both the log-sigmoid and the identify transfer functions, we use a neural network of [2-10-3] neurons. The NLML and the NLPP per data point are shown in Table 2. The mGP outperforms all other methods evaluated. We believe that this is due to the mapping M, which transforms the input space so as to have a single natural frequency. Figure 3b shows the intensity map of the feature space after the mGP transformed the inputs using a neural network with the identify transfer function. Figure 3c shows the intensity map of the feature when the log-sigmoid transfer function is used. Both transformations tend to make the feature space smother compared to the initial input space. This effect is the result of the transformations, which aim to equalize the natural frequencies of the original function in order to capture them more efficiently with a single length-scale. The effects of these transformations are clearly visible in the spectrogram of the mGP (identity) in Figure 3e and of the mGP (log-sigmoid) in Figure 3f. The smaller support of the spectrum, obtained through the non-linear transformations performed by mGP using the log-sigmoid transfer function, translates into superior prediction performance.

3.3. Bipedal Robot Locomotion

Modeling data from real robots can be challenging when the robot has physical interactions with the environment. Especially in bipedal locomotion, we lack good contact



(a) Intensity map of the function.

(b) Intensity map of the learned feature space for the mGP (identity).

(c) Intensity map of the learned feature space for the mGP (log-sigmoid).

(d) Spectrum of the function.

(e) Spectrum of the learned feature space for the mGP (identity).

(f) Spectrum of the learned feature space for the mGP (log-sigmoid).

Figure 3: **Multiple Length-Scales**: Intensity map of (a) the considered function, (b) the learned feature space of the mGP with a linear activation function and (c) with a log-sigmoid activation. (d)–(f) The corresponding Spectrum for (d) the original function and the learned feature space for (e) mGP (identity) and (f) mGP (log-sigmoid). The spectral analysis of the original function shows the presence of multiple frequencies. The transformations learned by both variants of mGP focus the spectrum of the feature space towards a more compact frequencies support.

Table 2: **Multiple Length-Scales**: NLML per data point for the training set and NLPP per data point for the test set. The mGP captures the nature of the underlying function better than a standard GP in both the training and test sets.

Method	Training set		Test set	
	NLML	RMSE	NLPP	RMSE
GP SE-ARD	-2.46	0.40×10^{-3}	-4.34	1.51×10^{-2}
GP NN	-1.57	1.52×10^{-3}	-2.53	6.32×10^{-2}
mGP (log-sigmoid)	$\mathbf{-6.61}$	$\mathbf{0.37 \times 10^{-4}}$	$\mathbf{-7.37}$	$\mathbf{0.58 \times 10^{-4}}$
mGP (identity)	-5.60	0.79×10^{-4}	-6.63	2.36×10^{-3}
RandEmb + GP SE-ARD	-0.47	6.84×10^{-3}	-1.29	1.19×10^{-1}

force and friction models. Thus, we evaluate our mGP approach on modeling data from the bio-inspired bipedal walker *Fox* (Renjewski, 2012) shown in Figure 4. The data set consists of measurements of six covariates recorded at regular intervals of 0.0125 sec. The covariates are the angles of the right and left hip joints, the angles of the right and left knee joints and two binary signals from the ground contact sensors. We consider the regression task where the left knee joint is the prediction target $\mathcal{Y}$ and the remaining five covariates are the inputs $\mathcal{X}$. For training we extract 400 consecutive data points, while we test on the following 500 data points. The mGP uses a network structure [1-30-3].

Table 3 shows that the mGP models the data better than the other models. The standard GPs with SE-ARD or NN covariance function predict the knee angle relatively well.

Figure 5 shows that the mGP has larger variance of the prediction for areas where fast movement occurs due to leg swinging. However, it captures the structure and regularity of the data better, such as the mechanically enforced upper bound at 185 degrees. The uncertainty of about 20 degrees is reasonable for the fast changes in the knee angle during the swinging phase. However, the same uncertainty of noise is unrealistic once the knee is fully extended at 185 degrees. Therefore, for control purposes, using the mGP model would be preferable. This is a positive sign of the potential of mGP to learn representations that are meaningful for the overall regression task. Figure 6 visualizes two dimensions of the learned feature space in which the walking trajectory is smoothly embedded.

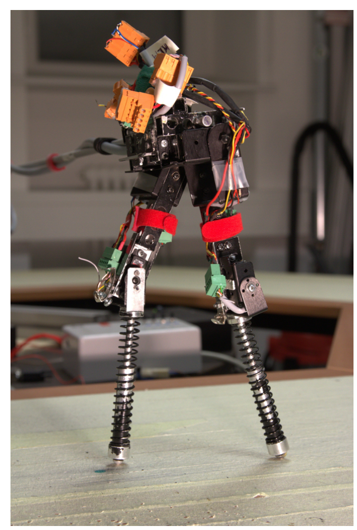
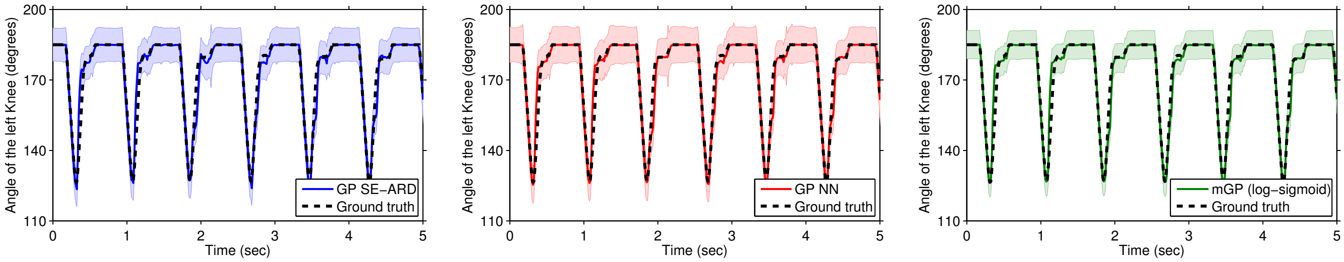


Figure 4: **Bipedal Robot Locomotion**: The bio-inspired bipedal walker *Fox* from which the dataset is generated.

4. Discussion

Unlike neural networks, which have been successfully used to extract complex features, MacKay (1998) argued that GPs are unsuited for feature learning. However, with



(a) GP with SE-ARD (b) GP with NN (c) mGP (log-sigmoid)

Figure 5: **Bipedal Robot Locomotion**: Predictive mean and 95% confidence bounds on the test set on real robot walking data for (a) GP with SE-ARD covariance function, (b) GP with NN covariance function and (c) mGP (log-sigmoid). The mGP (log-sigmoid) captures the structure of the data better compared to GPs with either SE-ARD or NN covariance functions.

Table 3: **Bipedal Robot Locomotion**: NLML per data point for the training set and NLPP per data point for the test set. The mGP captures the nature of the underlying function well in both the training and test sets.

Method	Training set		Test set	
	NLML	RMSE	NLPP	RMSE
GP SE-ARD	-0.01	0.18	-0.13	0.20
GP NN	0.04	**0.17**	-0.13	0.20
mGP (log-sigmoid)	$\mathbf{-0.28}$	**0.17**	$\mathbf{-0.18}$	**0.19**
mGP (identity)	0.97	0.03	0.86	0.66
PCA + GP SE-ARD	0.01	0.18	-0.12	0.20
RandEmb + GP SE-ARD	0.16	0.18	-0.09	0.20

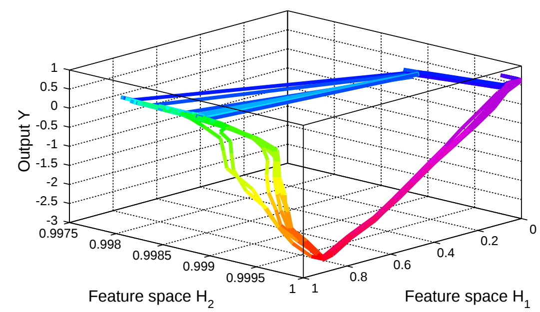


Figure 6: **Bipedal Robot Locomotion**: The mGP learns a smooth feature space representation (the color indicates the phase during a single step).

growing complexity of the regression problem, the discovery of useful data representations (i.e., features) is often necessary. Despite similarities between neural networks and GPs (Neal, 1995), it is still unclear how to exploit the best of both worlds. Inspired by deep neural networks, deep GPs stack multiple layers of GP latent variable models (Damianou and Lawrence, 2013). This method can be used also in a supervised regression framework, which renders it similar to our proposed mGP. The main differences between Deep GPs and the mGP is that (a) Deep GPs integrate out the latent functions connecting the individual layers and do not require to explicitly define a deterministic transformation structure and, (b) unlike the mGP, the Deep GP is not a full GP. Our mGP model extends a standard GP by learning corresponding useful data representations for the regression problem at hand. In particular, it can discover feature representations that comply with the implicit assumptions of the GP covariance function employed.

One of the main challenges of training mGPs using neural networks as mapping M is the unwieldy joint optimization of the parameters $\boldsymbol{\theta}_{\text{mGP}}$. The difficulty resides in the non-convexity of the NLML, leading to multiple local optima. Depending on the number of parameters $\boldsymbol{\theta}_M$ of the feature map M, the problem of local optima can be severe. This problem is well-known for neural networks, and there may be feasible alternatives to L-BFGS, such as the Hessian-free optimization proposed by Martens (2010). Additionally, sparsity and low-rank approximations in $\boldsymbol{W}$ and $\boldsymbol{B}$ can be beneficial to reduce the complexity of the optimization.

The extreme expressiveness of the mGP does not prevent the model from solving "easy" regression tasks. For a proof-of-concept, we applied the mGP to modeling a sinusoidal function, which is very easy to model with a standard GP. The results in Table 4 suggest that even for simple functions the mGP performs as good as a standard GP.

Increasing the number of parameters of the mapping M intuitively leads to an increased flexibility in the learned covariance function. However, when the number of parameters exceeds the size of data set, the model is prone to overfitting. For example, during experimental validation of the step function, we noticed the undesirable effect that the mGP could model discontinuities at locations slightly offset from their actual locations. In these cases, training data was sparse around the locations of discontinuity. This observed effect is due to over-fitting of the deterministic transformation M. Ideally, we replace this deterministic mapping with a probabilistic one, which would describe the uncertainty about the location of the discontinuity. In a fully Bayesian framework, we would average over possible models of the discontinuity. However, the use of such a probabilistic mapping in the context of GP regression is analytically intractable in closed form (Schmidt and O'Hagan, 2003) and would require to train GPs with uncertain inputs. This kind of GP training is also analytically intractable, although approximations exist (Lawrence, 2005; Wang et al., 2008; McHutchon and Rasmussen, 2011; Titsias and Lawrence, 2010).

Table 4: **Smooth function**: NLML per data point for the training set and NLPP per data point for the test set. There is no relevant difference between mGP and standard GP in modeling smooth functions.

Method	Training set		Test set	
	NLML	RMSE	NLPP	RMSE
GP SE-ARD	-4.30	2.78×10^{-3}	-4.42	2.91×10^{-3}
mGP (log-sigmoid)	-4.31	2.76×10^{-3}	-4.42	2.90×10^{-3}

5. Conclusion

The quality of a Gaussian process model strongly depends on an appropriate covariance function. However, designing such a covariance function is challenging for some classes of functions, e.g., highly non-linear functions. To model such complex functions we introduced Manifold Gaussian Processes. The key idea is to decompose the overall regression into learning a feature space mapping and a GP regression that maps from this feature space to the observed space. Both the input transformation and the GP regression are learned jointly and supervisedly by maximizing the marginal likelihood. The mGP is a valid GP for the overall regression task using a more expressive covariance function.

The mGP successfully modeled highly non-liner functions, e.g., step functions or effects of ground contacts in robot locomotion, where standard GPs fail. Applications that profit from the enhanced modeling capabilities of the mGP include robot modeling (e.g., contact and stiction modeling), reinforcement learning, and Bayesian optimization.

Acknowledgments

The research leading to these results has received funding from the European Council under grant agreement #600716 (CoDyCo - FP7/2007–2013). M. P. Deisenroth was supported by a Google Faculty Research Award.

References.

A. C. Damianou and N. D. Lawrence. Deep Gaussian Processes. In *AISTATS*, 2013.

D. Duvenaud, J. R. Lloyd, R. Grosse, J. B. Tenenbaum, and Z. Ghahramani. Structure Discovery in Nonparametric Regression through Compositional Kernel Search. In *ICML*, 2013.

N. HajiGhassemi and M. P. Deisenroth. Approximate Inference for Long-Term Forecasting with Periodic Gaussian Processes. In *AISTATS*, 2014.

A. Hyvärinen and E. Oja. Independent Component Analysis: Algorithms and Applications. *Neural Networks*, 13(4): 411–430, 2000.

N. D. Lawrence. Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models. *JMLR*, 6:1783–1816, November 2005.

D. C. Liu and J. Nocedal. On the Limited Memory BFGS Method for Large Scale Optimization. *Mathematical Programming*, 45(3):503–528, 1989.

D. J. C. MacKay. Introduction to Gaussian Processes. In *Neural Networks and Machine Learning*, volume 168, pages 133–165. 1998.

J. Martens. Deep Learning via Hessian-free Optimization. In *ICML*, 2010.

A. McHutchon and C. E. Rasmussen. Gaussian Process Training with Input Noise. In *NIPS*, 2011.

R. M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1995.

K. Pearson. On Lines and Planes of Closest Fit to Systems of Points in Space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11): 559–572, 1901.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

D. Renjewski. *An Engineering Contribution to Human Gait Biomechanics*. PhD thesis, TU Ilmenau, 2012.

S. T. Roweis and L. K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290 (5500):2323–2326, 2000.

R. Salakhutdinov and G. Hinton. Using Deep Belief Nets to Learn Covariance Kernels for Gaussian Processes. In *NIPS*, 2007.

A. M. Schmidt and A. O'Hagan. Bayesian Inference for Non-stationary Spatial Covariance Structure via Spatial Deformations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(3):743–758, 2003.

E. Snelson and Z. Ghahramani. Variable Noise and Dimensionality Reduction for Sparse Gaussian Processes. In *UAI*, 2006.

E. Snelson, C. E. Rasmussen, and Z. Ghahramani. Warped Gaussian Processes. In *NIPS*, 2004.

J. Snoek, R. P. Adams, and H. Larochelle. Nonparametric Guidance of Autoencoder Representations using Label Information. *JMLR*, 13:2567–2588, 2012.

J. Snoek, K. Swersky, R. S. Zemel, and R. P. Adams. Input Warping for Bayesian Optimization of Non-stationary Functions. In *ICML*, 2014.

J. B. Tenenbaum, V. De Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, 2000.

M. K. Titsias and N. D. Lawrence. Bayesian Gaussian Process Latent Variable Model. In *AISTATS*, 2010.

P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and Composing Robust Features with Denoising Autoencoders. *ICML*, 2008.

N. Wahlström, T. B. Schön, and M. P. Deisenroth. Learning Deep Dynamical Models From Image Pixels. In *SYSID*, 2015.

J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian Process Dynamical Models for Human Motion. *PAMI*, 30(2):283–298, 2008.

A. G. Wilson and R. P. Adams. Gaussian Process Kernels for Pattern Discovery and Extrapolation. *ICML*, 2013.