Manifold Gaussian Processes for Regression

Roberto Calandra*, Jan Peters*,†, Carl Edward Rasmussen‡ and Marc Peter Deisenroth§

†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; Hajighasemi 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
Supervised learning integrating out a latent space is considered the task of learning a regression function $F$ by latent space $L$ as shown in Figure 1b. In a full Bayesian framework, the latent space $L$ 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 $H$, 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 $H$ allows learning a simpler mapping $G:H \rightarrow Y$. For high-dimensional inputs, the data often lies on a lower-dimensional manifold $H$, e.g., due to non-discriminant or strongly correlated covariates. The lower-dimensional feature space $H$ 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

---

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 $x_n \in X \subseteq \mathbb{R}^D$ and respective outputs $y_n \in Y \subseteq \mathbb{R}$, where $y_n = F(x_n) + w$, $w \sim \mathcal{N}(0, \sigma^2_w)$, $n = 1, \ldots, N$. The training data is denoted by $X$ and $Y$ for the inputs and targets, respectively. We consider the task of learning a regression function $F:X \rightarrow Y$. 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 $L$. The function $F$ can then be decomposed into $F = G \circ M$, where $M:X \rightarrow L$ and $G:L \rightarrow Y$, as shown in Figure 1b. In a full Bayesian framework, the latent space $L$ is integrated out to solve the regression task $F$, which is often analytically intractable.

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.

---

Figure 1: Different regression settings to learn the function $F:X \rightarrow Y$. (a) Standard supervised regression. (b) Regression with an auxiliary latent space $L$ that allows to simplify the task. In a full Bayesian framework, $L$ would be integrated out, which is analytically intractable. (c) Decomposition of the overall regression task $F$ into discovering a feature space $H$ using the map $M$ and a subsequent (conditional) regression $G|M$. (d) Our mGP learns the mappings $G$ and $M$ jointly.
and fully defined by a mean function \( m \) and a covariance function \( k \) with \( \Lambda \) and a measurement noise variance. Specifically, we use the squared exponential covariance function with Automatic Relevance Determination (ARD) \( k \). Using the chain-rule, the corresponding gradient can be computed analytically as

\[
\frac{\partial \text{NLML}(\theta)}{\partial \theta} = \frac{\partial \text{NLML}(\theta)}{\partial \theta^k} \frac{\partial k}{\partial \theta} \quad ,
\]

which allows us to optimize the hyperparameters using Quasi-Newton optimization, e.g., L-BFGS [Liu and No-cedal, 1989].

### 2.3. Manifold Gaussian Processes

In this section, we describe the mGP model and its parameters \( \theta_{mGP} \) 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 [14], the mGP considers the overall regression as a composition of functions

\[
F = G \circ M.
\]

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 \( \mathcal{H} \subseteq \mathbb{R}^2 \), 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 \( H = M(\mathcal{X}) \) 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

\[
\tilde{k}(x_p, x_q) = k(M(x_p), M(x_q)),
\]

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 \( x_s \) can then be derived from the predictive distribution of a standard GP in Equation (2) as

\[
p(F(x_s)|D, x_s) = p((G \circ M)(x_s)|D, x_s)
\]

\[
\quad = N(\mu(M(x_s))),\sigma^2(M(x_s)))
\]

\[
\quad = \frac{1}{2} Y^T (\tilde{K} + \sigma^2_w I)^{-1} Y + \frac{1}{2} \log |\tilde{K} + \sigma^2_w I|
\]

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

\[
\frac{\partial \text{NLML}(\theta)}{\partial \theta} = \frac{\partial \text{NLML}(\theta)}{\partial \theta^k} \frac{\partial k}{\partial \theta} \quad ,
\]

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

#### 2.3.2. Training

We train the mGP by jointly optimizing the parameters \( \theta_M \) of the transformation \( M \) and the GP hyperparameters \( \theta_G \). For learning the parameters \( \theta_{mGP} = [\theta_M, \theta_G] \), we minimize the NLML as in the standard GP regression. Considering the composition of the mapping \( F = G \circ M \), the NLML becomes

\[
\text{NLML}(\theta_{mGP}) = -\log p(Y|X, \theta_{mGP})
\]

\[
\quad \approx \frac{1}{2} Y^T (K_{\theta_{mGP}} + \sigma^2_w I)^{-1} Y + \frac{1}{2} \log |K_{\theta_{mGP}} + \sigma^2_w I|
\]

Note that \( K_{\theta_{mGP}} \) depends on both \( \theta_G \) and \( \theta_M \), unlike \( K_\theta \) from Equation (7), which depends only on \( \theta_G \). The analytic gradients \( \partial \text{NLML} / \partial \theta \) of the objective in Equation (14)
with respect to the parameters \( \theta_G \) are computed as in the standard GP, i.e.,
\[
\frac{\partial \text{NLML}(\theta_{mGP})}{\partial \theta_G} = \frac{\partial \text{NLML}(\theta_{mGP})}{\partial K_{\theta_{map}}} \frac{\partial K_{\theta_{map}}}{\partial \theta_G}.
\]
(14)
The gradients of the parameters \( \theta_M \) of the feature mapping are computed by applying the chain-rule
\[
\frac{\partial \text{NLML}(\theta_{mGP})}{\partial \theta_M} = \frac{\partial \text{NLML}(\theta_{mGP})}{\partial K_{\theta_{map}}} \frac{\partial K_{\theta_{map}}}{\partial \theta_M} \frac{\partial H}{\partial \theta_M},
\]
(15)
where only \( \partial H/\partial \theta_M \) depends on the chosen input transformation \( M \), while \( \partial K_{\theta_{map}}/\partial H \) is the gradient of the kernel matrix with respect to the \( Q \)-dimensional GP training inputs \( H = M(X) \). Similarly to standard GP, the parameters \( \theta_{mGP} \) 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
\[
T_i(Z) = \sigma(W_iZ + B_i),
\]
(16)
where \( Z \) is the input of the layer, \( \sigma \) is the transfer function, and \( W_i \) and \( B_i \) are the weights and the bias of the layer, respectively. Therefore, the input transformation \( M \) of Equation (10) is \( M(X) = (T_l \circ \ldots \circ T_1)(X) \). The parameters \( \theta_M \) of the neural network \( M \) are the weights and biases of the whole network, so that \( \theta_M = \{W_1, B_1, \ldots, W_l, B_l\} \). The gradients \( \partial H/\partial \theta_M \) 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 can be successfully applied to learning discontinuous functions, a daunting undertaking with an off-the-shelf covariance function, due to its underlying smoothness assumptions. For the training set, we consider the step function
\[
y = F(x) + w, \quad w \sim \mathcal{N}(0, 0.01^2),
\]
(18)
For training, 100 inputs points are sampled from \( \mathcal{N}(0, 1) \) 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 1a. 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 1b, 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 \( H = \alpha X \), where the elements of \( \alpha \) are randomly sampled from a normal distribution.
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

\[ y = 1 - \mathcal{N}(x_2|3, 0.5^2) - \mathcal{N}(x_2| -3, 0.5^2) + \frac{x_1}{100} \]  \hspace{1cm} (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 3e shows the intensity map of the feature space after the mGP transformed the inputs using a neural network with the identify transfer function. Figure 3f shows the intensity map of the feature when the log-sigmoid transfer function is used. Both transformations tend to make the feature space smoother 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 3c 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...
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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Training set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NLML</td>
<td>RMSE</td>
</tr>
<tr>
<td>GP SE-ARD</td>
<td>−2.46</td>
<td>0.40 × 10⁻³</td>
</tr>
<tr>
<td>GP NN</td>
<td>−1.57</td>
<td>1.52 × 10⁻³</td>
</tr>
<tr>
<td>mGP (log-sigmoid)</td>
<td>−6.61</td>
<td>0.37 × 10⁻⁴</td>
</tr>
<tr>
<td>mGP (identity)</td>
<td>−5.60</td>
<td>0.79 × 10⁻⁴</td>
</tr>
<tr>
<td>RandEmb + GP SE-ARD</td>
<td>−0.47</td>
<td>0.84 × 10⁻³</td>
</tr>
</tbody>
</table>

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
training 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 over-fitting. 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Training set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NLML</td>
<td>RMSE</td>
</tr>
<tr>
<td>GP SE-ARD</td>
<td>-4.30</td>
<td>2.78×10⁻³</td>
</tr>
<tr>
<td>mGP (log-sigmoid)</td>
<td>-4.31</td>
<td>2.76×10⁻³</td>
</tr>
</tbody>
</table>

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-linear 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.


