
Input Warping for Bayesian Optimization of Non-Stationary Functions

Jasper Snoek

School of Engineering and Applied Sciences, Harvard University

JSNOEK@SEAS.HARVARD.EDU

Kevin Swersky

Department of Computer Science, University of Toronto

KSWERSKY@CS.TORONTO.EDU

Richard Zemel

Department of Computer Science, University of Toronto and Canadian Institute for Advanced Research

ZEMEL@CS.TORONTO.EDU

Ryan P. Adams

School of Engineering and Applied Sciences, Harvard University

RPA@SEAS.HARVARD.EDU

Abstract

Bayesian optimization has proven to be a highly effective methodology for the global optimization of unknown, expensive and multimodal functions. The ability to accurately model distributions over functions is critical to the effectiveness of Bayesian optimization. Although Gaussian processes provide a flexible prior over functions, there are various classes of functions that remain difficult to model. One of the most frequently occurring of these is the class of non-stationary functions. The optimization of the hyperparameters of machine learning algorithms is a problem domain in which parameters are often manually transformed *a priori*, for example by optimizing in “log-space,” to mitigate the effects of spatially-varying length scale. We develop a methodology for automatically learning a wide family of bijective transformations or *warpings* of the input space using the Beta cumulative distribution function. We further extend the warping framework to multi-task Bayesian optimization so that multiple tasks can be warped into a jointly stationary space. On a set of challenging benchmark optimization tasks, we observe that the inclusion of warping greatly improves on the state-of-the-art, producing better results faster and more reliably.

1 Introduction

Bayesian optimization is a strategy for the global optimization of noisy, black-box functions. The goal is to find the minimum of an expensive function of interest as quickly as possible. Bayesian optimization fits a surrogate model that estimates the expensive function, and a proxy optimization is performed on this in order to select promising locations to query. Naturally, the ability of the surrogate to accurately model the underlying function is crucial to the success of the optimization routine. Recent work in machine learning has revisited the idea of Bayesian optimization (e.g., Osborne et al., 2009; Brochu et al., 2010; Srinivas et al., 2010; Hutter et al., 2011; Bergstra et al., 2011; Bull, 2011; Snoek et al., 2012; Hennig and Schuler, 2012) in large part due to advances in the ability to efficiently and accurately model statistical distributions over large classes of real-world functions. Gaussian processes (GPs) (see, e.g., Rasmussen and Williams, 2006) provide a powerful framework to express flexible prior distributions over smooth functions, yielding accurate estimates of the expected value of the function at any given input, but crucially also uncertainty estimates over that value. These are the two main components that enable the exploration and exploitation tradeoff that makes Bayesian optimization so effective.

A major limitation of the most commonly used form of Gaussian process regression is the assumption of stationarity — that the covariance between two outputs is invariant to translations in input space. This assumption simplifies the regression task, but hurts the ability of the Gaussian process to model more realistic non-stationary functions. This presents a challenge for Bayesian optimization, as many problems of interest are inherently non-stationary. For example, when optimizing the hyperparameters of a

machine learning algorithm, we might expect the objective function to have a short length scale near the optimum, but have a long length scale far away from the optimum. That is, we would expect bad hyperparameters to yield similar bad performance everywhere (e.g., classifying at random) but expect the generalization performance to be sensitive to small tweaks in good hyperparameter regimes.

We introduce a simple solution that allows Gaussian processes to model a large variety of non-stationary functions that is particularly well suited to Bayesian optimization. We automatically learn a bijective *warping* of the inputs that removes major non-stationary effects. This is achieved by projecting each dimension of the input through the cumulative distribution function of the Beta distribution, while marginalizing over the shape of the warping. Our approach is computationally efficient, captures a variety of desirable transformations, such as logarithmic, exponential, sigmoidal, etc., and is easily interpretable. In the context of Bayesian optimization, understanding the parameter space is often just as important as achieving the best possible result and our approach lends itself to a straightforward analysis of the non-stationarities in a given problem domain.

We extend this idea to multi-task Bayesian optimization (Swersky et al., 2013) so that multiple tasks can be warped into a jointly stationary space. Thus, tasks can be warped onto one another in order to better take advantage of their shared structure.

In the empirical study that forms the experimental part of this paper, we show that modeling non-stationarity is extremely important and yields significant empirical improvements in the performance of Bayesian optimization. For example, we show that on a recently introduced Bayesian optimization benchmark (Eggenberger et al., 2013), our method outperforms all of the previous state-of-the-art algorithms on the problems with continuous-valued parameters. We further observe that on four different challenging machine learning optimization tasks our method outperforms that of Snoek et al. (2012), consistently converging to a better result in fewer function evaluations. As our methodology involves a transformation of the inputs, this strategy generalizes to a wide variety of models and algorithms. Empirically, modeling non-stationarity is a fundamentally important component of effective Bayesian optimization.

2 Background and Related Work

2.1 Gaussian Processes

The Gaussian process is a powerful and flexible prior distribution over functions $f : \mathcal{X} \rightarrow \mathbb{R}$ which is widely used for non-linear Bayesian regression. An attractive property of

the Gaussian process in the context of Bayesian optimization is that, conditioned on a set of observations, the expected output value and corresponding uncertainty of any unobserved input is easily computed.

The properties of the Gaussian process are specified by a mean function $m : \mathcal{X} \rightarrow \mathbb{R}$ and a positive definite covariance, or kernel, function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$. Given a finite set of training points $I_N = \{\mathbf{x}_n, y_n\}_{n=1}^N$, where $\mathbf{x}_n \in \mathcal{X}$, $y_n \in \mathbb{R}$, the predictive mean and covariance under a GP can be respectively expressed as:

$$\mu(\mathbf{x}; I_N) = m(\mathbf{X}) + K(\mathbf{X}, \mathbf{x})^\top K(\mathbf{X}, \mathbf{X})^{-1}(\mathbf{y} - m(\mathbf{X})), \quad (1)$$

$$\Sigma(\mathbf{x}, \mathbf{x}'; I_N) = K(\mathbf{x}, \mathbf{x}') - K(\mathbf{X}, \mathbf{x})^\top K(\mathbf{X}, \mathbf{X})^{-1} K(\mathbf{X}, \mathbf{x}'). \quad (2)$$

Here $K(\mathbf{X}, \mathbf{x})$ is the N -dimensional column vector of cross-covariances between \mathbf{x} and the set \mathbf{X} . The $N \times N$ matrix $K(\mathbf{X}, \mathbf{X})$ is the Gram matrix for the set \mathbf{X} resulting from applying the covariance function $K(\mathbf{x}, \mathbf{x}')$ pairwise over the set $\{\mathbf{x}_n\}_{n=1}^N$. The most common choices of covariance functions $K(\mathbf{x}, \mathbf{x}')$ are functions of $r(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|$, such as the automatic relevance determination (ARD) exponentiated quadratic covariance

$$K_{SE}(\mathbf{x}, \mathbf{x}') = \theta_0 \exp(-r^2) \quad r = \sum_{d=1}^D (x_d - x'_d)^2 / \theta_d^2, \quad (3)$$

or the ARD Matérn 5/2 kernel advocated for hyperparameter tuning with Bayesian optimization by Snoek et al. (2012):

$$K_{M52}(\mathbf{x}, \mathbf{x}') = \theta_0 \left(1 + \sqrt{5}r^2 + \frac{5}{3}r^2 \right) \exp \left\{ -\sqrt{5}r^2 \right\}. \quad (4)$$

Such covariance functions are invariant to translations along the input space and thus are *stationary*.

2.2 Non-stationary Gaussian Process Regression

Numerous approaches have been proposed to extend GPs to model non-stationary functions. Gramacy (2005) proposed a Bayesian treed GP model which accommodates various complex non-stationarities through modeling the data using multiple GPs with different covariances. Various non-stationary covariance functions have been proposed (e.g., Higdon et al., 1998; Rasmussen and Williams, 2006). Previously, Sampson and Guttorp (1992) proposed projecting the inputs into a stationary latent space using a combination of metric multidimensional scaling and thin plate splines. Schmidt and O’Hagan (2003) extended this warping approach for general GP regression problems using a flexible GP mapping. Spatial deformations of two dimensional inputs have been studied extensively in the spatial statistics

literature (Anderes and Stein, 2008). Bornn et al. (2012) project the inputs into a higher dimensional stationary latent representation. Snelson et al. (2003) apply a warping to the output space, \mathbf{y} , while Adams and Stegle (2008) perform input-dependent output scaling with a second Gaussian process.

Compared to these approaches, our approach is relatively simple, yet as we will demonstrate, flexible enough to capture a wide variety of nonstationary behaviours. Our principal aim is to show that addressing nonstationarity is a critical component of effective Bayesian optimization, and that any advantages gained from using our approach would likely generalize to more elaborate techniques.

2.3 Multi-Task Gaussian Processes

Many problems involve making predictions over multiple datasets (we will henceforth refer to these prediction problems as tasks). When the datasets share an input domain, and the mappings from inputs to outputs are correlated, then these correlations can be used to share information between different tasks and improve predictive performance. There have been many extensions of Gaussian processes to the multi-task setting, e.g., Goovaerts (1997); Alvarez and Lawrence (2011). However, a basic and surprisingly effective approach is to assume that each task is derived from a single latent function which is transformed to produce each output (Teh et al., 2005; Bonilla et al., 2008).

Formally, this approach involves combining a kernel over inputs $K(\mathbf{x}, \mathbf{x}')$ and a kernel over task indices $K(t, t')$, $t = \{1, \dots, T\}$ via a product to form the joint kernel:

$$K((\mathbf{x}, t), (\mathbf{x}', t')) = K_T(t, t')K(\mathbf{x}, \mathbf{x}'). \quad (5)$$

We infer the elements of $K_T(t, t')$ directly using the spherical parametrization of a covariance matrix (Osborne, 2010; Pinheiro and Bates, 1996).

2.4 Bayesian Optimization

Bayesian optimization is a general framework for the global optimization of noisy, expensive, black-box functions (Mockus et al., 1978), see Brochu et al. (2010) or Lizotte (2008) for an in-depth explanation and review. The strategy relies on the use of a relatively cheap probabilistic model that can be queried liberally as a surrogate in order to more effectively evaluate an expensive function of interest. Bayes’ rule is used to derive the posterior estimate of the true function, given observations, and the surrogate is then used to determine, via a proxy optimization over an *acquisition function*, the next most promising point to query. Using the posterior mean and variance of the probabilistic model, the acquisition function generally expresses a tradeoff between exploitation and exploration. Numerous acquisition functions and combinations thereof have been

proposed (e.g., Kushner, 1964; Srinivas et al., 2010; Hoffman et al., 2011).

In this work, we follow the common approach, which is to use a GP to define a distribution over objective functions from the input space to a loss that one wishes to minimize. Our approach is based on that of Jones (2001). Specifically, we use a GP surrogate, and the *expected improvement* acquisition function (Mockus et al., 1978). Let $\sigma^2(\mathbf{x}) = \Sigma(\mathbf{x}, \mathbf{x})$ be the marginal predictive variance of a GP, and define

$$\gamma(\mathbf{x}) = \frac{f(\mathbf{x}_{\text{best}}) - \mu(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta)}{\sigma(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta)}, \quad (6)$$

where $f(\mathbf{x}_{\text{best}})$ is the lowest observed value. The expected improvement criterion is defined as

$$a_{EI}(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta) = \sigma(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta) (\gamma(\mathbf{x})\Phi(\gamma(\mathbf{x})) + \mathcal{N}(\gamma(\mathbf{x}); 0, 1)). \quad (7)$$

Here $\Phi(\cdot)$ is the cumulative distribution function of a standard normal, and $\mathcal{N}(\cdot; 0, 1)$ is the density of a standard normal. Note that the method proposed in this paper is independent of the choice of acquisition function and do not affect its analytic properties.

2.5 Multi-Task Bayesian Optimization

When utilizing machine learning in practice, a single model will often need to be trained on multiple datasets. This can happen when e.g., new data is collected and a model must be retrained. In these scenarios we can think of each dataset as a different task and use multi-task Gaussian processes to predict where to query next. In Krause and Ong (2011), this idea was applied to find peptide sequences that bind to molecules for vaccine design, while in Swersky et al. (2013) it was applied to hyperparameter optimization. In these cases it was shown that sharing information between tasks can be extremely beneficial for Bayesian optimization. Other approaches include Bardenet et al. (2013), which finds a joint latent function over tasks explicitly using a ranking model, and Hutter et al. (2011) which uses a set of auxiliary task features to improve prediction.

3 Input Warping

We assume that we have a positive definite covariance function $K(\mathbf{x}, \bar{\mathbf{x}})$, where $\mathbf{x}, \bar{\mathbf{x}} \in [0, 1]^D$ due to projecting a bounded input range to the unit hypercube. In practice, when tuning the hyperparameters of an algorithm, e.g., the regularization parameter of a support vector machine, researchers often first transform the input space using a monotonic function such as the natural logarithm and then perform a grid search in this transformed space. Such an optimization in “log-space” takes advantage of *a priori*

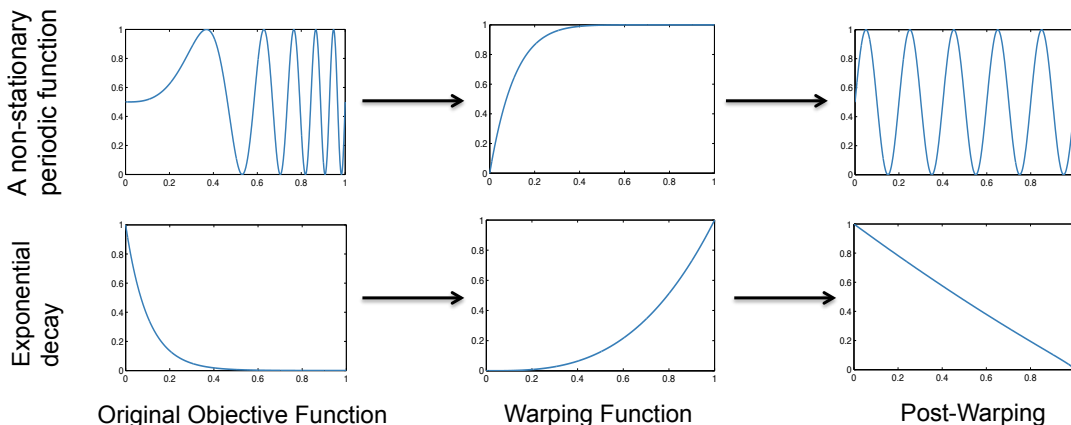


Figure 1. Two examples of how input warping using the Beta CDF can transform a non-stationary function into a stationary one. The warping function maps the original inputs on the horizontal axis to new inputs shown on the vertical axis. The effect is to stretch and contract regions of the input space in such a manner as to remove non-stationarity.

knowledge of the non-stationarity that is inherent in the input space. Often however, the non-stationary properties of the input space are not known *a priori* and such a transformation is generally a crude approximation to the ideal (unknown) transformation. Our approach is to instead consider a class of bijective warping functions, and estimate them from previous objective function evaluations. We can then use commonly-engineered transformations—such as the log transform—to specify a prior on bijections. Specifically, we change the kernel function to be $K(w(\mathbf{x}), w(\tilde{\mathbf{x}}))$,

$$\begin{aligned} w_d(\mathbf{x}_d) &= \text{BetaCDF}(\mathbf{x}_d; \alpha_d, \beta_d), \\ &= \int_0^{\mathbf{x}_d} \frac{u^{\alpha_d-1}(1-u)^{\beta_d-1}}{B(\alpha_d, \beta_d)} du, \end{aligned} \quad (8)$$

where BetaCDF refers to the Beta cumulative distribution function and $B(\alpha, \beta)$ is the normalization constant. That is, $w : [0, 1]^D \rightarrow [0, 1]^D$ is a vector-valued function in which the d th output dimension is a function of the d th input dimension, and is specified by the cumulative distribution function of the Beta distribution. Each of these D bijective transformations from $[0, 1]$ to $[0, 1]$ has a unique shape, determined by parameters $\alpha_d > 0$ and $\beta_d > 0$. The Beta CDF has no closed form solution for non-integer values of α and β , however accurate approximations are implemented in many statistical software packages.

Alternatively, one can think of input warping as applying a particular kind of non-stationary kernel to the original data. Examples of non-stationary functions and their corresponding ideal warping that transforms them into stationary functions are shown in Figure 1.

Our choice of the Beta distribution is motivated by the fact that it is capable of expressing a variety of monotonic warpings, while still being concisely parameterized. In general,

there are many other suitable choices.

3.1 Integrating over warpings

Rather than assume a single, explicit transformation function, we define a hierarchical Bayesian model by placing a prior over the shape parameters, α_d and β_d , of the bijections and integrating them out. We treat the collection $\{\alpha_d, \beta_d\}_{d=1}^D$ as hyperparameters of the covariance function and use Markov chain Monte Carlo via slice sampling, following the treatment of covariance hyperparameters from Snoek et al. (2012). We use a log-normal distribution, i.e.

$$\log(\alpha_d) \sim \mathcal{N}(\mu_\alpha, \sigma_\alpha) \quad \log(\beta_d) \sim \mathcal{N}(\mu_\beta, \sigma_\beta), \quad (9)$$

to express a prior for a wide family of desirable functions. Figure 2 demonstrates example warping functions arising from sampling transformation parameters from various instantiations of this prior. Note that the geometric mean or median of the zero-mean log-normal distribution for the α_d and β_d corresponds to the identity transform. With this prior the model centers itself on the identity transformation of the input space. In the following empirical analysis we use this formulation with a variance of 0.75. A nice property of this approach is that a user can easily specify a prior when they expect a specific form of warping, as we show in Figure 2.

3.2 Multi-Task Input Warping

When training the same model on different datasets, certain properties, such as the size of the dataset, can have a dramatic effect on the optimal hyperparameter settings. For example, a model trained on a small dataset will likely require more regularization than the same model trained on a

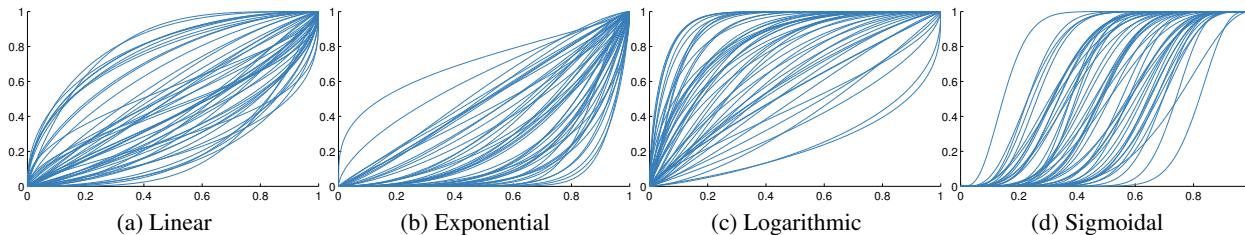


Figure 2. Each figure shows 50 warping functions resulting from the Beta CDF where the shape parameters α and β are sampled from a log-normal prior with a different mean and variance. The flexible Beta CDF captures many desirable warping functions and adjusting the prior over input warpings allows one to easily encode prior beliefs over the form of non-stationarity. For example, choosing $\mu_\alpha = \mu_\beta = 0$ and $\sigma_\alpha = \sigma_\beta = 0.5$ expresses a prior for slight or no warping (2a). Setting $\mu_\alpha = 0$, $\sigma_\alpha = 0.25$ and $\mu_\beta = \sigma_\beta = 1$ or $\mu_\alpha = \sigma_\alpha = 1$, $\mu_\beta = 0$ and $\sigma_\beta = 0.25$ expresses a prior for approximately exponential (2b) or logarithmic (2c) warping functions respectively. Approximately sigmoidal (2d) warpings that contract the outer regions of the space while expanding the center can be expressed as a prior with $\mu_\alpha = \mu_\beta = 2$ and $\sigma_\alpha = \sigma_\beta = 0.5$. One can also express logit shaped warpings (not shown here).

larger dataset. In other words, it is possible that one part of the input space on one task can be correlated with a different part of the input space on another task. To account for this, we allow each task to have its own set of warping parameters. Inferring these parameters will effectively try to warp both tasks into a jointly stationary space that is more suitably modeled by a standard multi-task kernel. In this way, large values on one task can map to small values on another, and vice versa.

4 Empirical Analyses

Our empirical analysis is comprised of three distinct experiments. In the first experiment, we compare to the method of Snoek et al. (2012) in order to demonstrate the effectiveness of input warping. In the second experiment, we compare to other hyperparameter optimization methods using a subset of the benchmark suite found in Eggenberger et al. (2013). Finally, we show how our multi-task extension can further benefit this important setting.

4.1 Comparison to Stationary GPs

Experimental setup We evaluate the standard Gaussian process expected improvement algorithm (GP EI MCMC) as implemented by Snoek et al. (2012), with and without warping. Following their treatment, we use the Matérn 5/2 kernel and we marginalize over kernel parameters θ using slice sampling (Murray and Adams, 2010). We repeat three of the experiments¹ from Snoek et al. (2012), and perform an experiment involving the tuning of a deep convolutional neural network² on a subset of the popular CIFAR-10 data set (Krizhevsky, 2009). The deep network consists of three convolutional layers and two fully connected layers and

¹See Snoek et al. (2012) for details of these experiments.

²We use the Deepnet package from <https://github.com/nitishsrivastava/deepnet>

we optimize over two learning rates, one for each layer type, six dropout regularization rates, six weight norm constraints, the number of hidden units per layer, a convolutional kernel size and a pooling size for a total of 21 hyperparameters. On the logistic regression problem we also compare to warping the input space *a priori* using the log-transform (optimizing in log-space).

Results Figure 3 shows that in all cases, dealing with non-stationary effects via input warpings greatly improves the convergence of the optimization. Of particular note, on the higher-dimensional convolutional network problem (Figure 3d) input warped Bayesian optimization consistently converges to a better solution than Bayesian optimization with a stationary GP.

In Figure 4 we plot examples of some of the inferred warpings. For logistic regression, Figure 4a shows that our method learns different logarithmic-like warpings for three dimensions and no warping for the fourth. Figure 4b shows how the posterior distribution over the learning rate warping evolves, becoming more extreme and more certain, as observations are gathered. Figure 4c shows that on both convolutional and dense layers, the intuition that one should log-transform the learning rates holds. For transformations on weight norm constraints, shown in Figure 4d, the weights connected to the inputs and outputs use a sigmoidal transformation, the convolutional-layer weights use an exponential transformation, and the dense-layer weights use a logarithmic transformation. Effectively, this means that the most variation in the error occurs in the medium, high and low scales respectively for these types of weights. Especially interesting are the wide variety of transformations that are learned for dropout on different layers, shown in Figure 4e. These show that different layers benefit from different dropout rates, which was also confirmed on test set error, and challenges the notion that they should just be set to 0.5 (Hinton et al., 2012).

Input Warping for Bayesian Optimization

| Experiment | # Evals | SMAC | Spearmint | TPE | # Evals | Spearmint + Warp |
|---------------------|---------|----------------------------------|------------------------------------|-------------------|---------|--------------------------------------|
| Branin (0.398) | 200 | 0.655 ± 0.27 | 0.398 ± 0.00 | 0.526 ± 0.13 | 40 | 0.398 ± 0.00 |
| Hartmann 6 (-3.322) | 200 | -2.977 ± 0.11 | -3.133 ± 0.41 | -2.823 ± 0.18 | 100 | -3.3166 ± 0.02 |
| Logistic Regression | 100 | 8.6 ± 0.9 | 7.3 ± 0.2 | 8.2 ± 0.6 | 40 | 6.88 ± 0.0 |
| LDA (On grid) | 50 | 1269.6 ± 2.9 | 1272.6 ± 10.3 | 1271.5 ± 3.5 | 50 | 1266.2 ± 0.1 |
| SVM (On grid) | 100 | 24.1 ± 0.1 | 24.6 ± 0.9 | 24.2 ± 0.0 | 100 | 24.1 ± 0.1 |

Table 1. We evaluate our algorithm on the continuous-valued parameter benchmarks proposed in Eggenberger et al. (2013). We compare to Sequential Model Based Algorithm Configuration (SMAC) (Hutter et al., 2011), the Tree Parzen Estimator (TPE) (Bergstra et al., 2011) and Spearmint (Snoek et al., 2012). The results for SMAC, Spearmint and TPE are reproduced from Eggenberger et al. (2013). Following the standard protocol for these benchmarks, each algorithm was run ten times for the given number of evaluations, and the average validation loss and standard deviation are reported. The algorithm with the lowest validation loss is shown in bold. We note that on some of the benchmarks our algorithm converges to a solution in far fewer evaluations than the protocol allows.

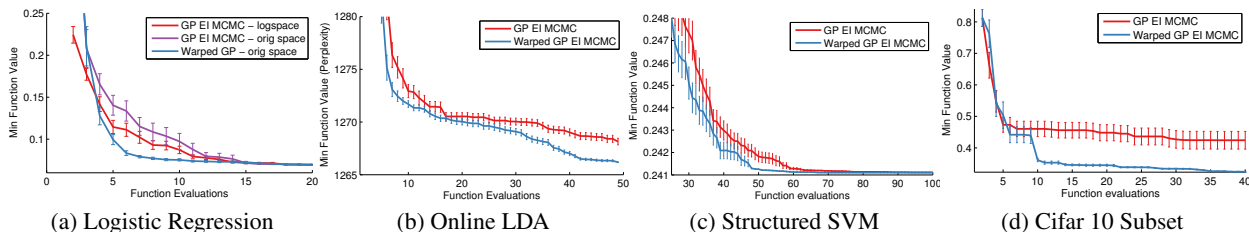


Figure 3. An empirical comparison of Bayesian optimization following the standard Gaussian process expected improvement algorithm (GP EI MCMC) and our strategy (Warped GP EI MCMC) for modeling input non-stationarity. The methods are compared on four challenging problems involving the optimization of the hyperparameters of popular machine learning algorithms.

It is clear that the learned warpings are non-trivial. In some cases, like with learning rates, they agree with intuition, while for others like dropout they yield surprising results. Given the number of hyperparameters and the variety of transformations, it is highly unlikely that even experts would be able to determine the whole set of appropriate warpings. This highlights the utility of learning them automatically.

4.2 HPOLib Continuous Benchmarks

Experimental setup In our next set of experiments, we tested our approach on the subset of benchmarks over continuous inputs from the HPOLib benchmark suite (Eggenberger et al., 2013). These benchmarks are designed to assess the strengths and weaknesses of several popular hyperparameter optimization schemes. All of the tested methods perform Bayesian optimization, however the underlying surrogate models differ significantly. The SMAC package (Hutter et al., 2011) uses a random forest, the Hyperopt package (Bergstra et al., 2011) uses the tree Parzen estimator, and the Spearmint package (Snoek et al., 2012) uses a Gaussian process. For our experiments, we augmented the Spearmint package with input warping.

Results Table 1 shows the results, where all but the warped results are taken from Eggenberger et al. (2013).

Overall, input warpings improve the performance of the Gaussian process approach such that it does at least as well as every other method, and in many cases better. Furthermore, the standard deviation also decreases significantly in many instances, meaning that the results are far more reliable. Finally, it is worth noting that the number of function evaluations required to solve the problems is also drastically reduced in many cases.

Interestingly, the random forest approach in SMAC also naturally deals with nonstationarity, albeit in a fundamentally different way, by partitioning the space in a non-uniform manner. There are several possibilities to explain the performance discrepancy. Unlike random forests, Gaussian processes produce a smooth function of the inputs, meaning that EI can be locally optimized via gradient methods, so it is possible that better query points are selected in this way. Alternatively, the random forest is not a well-defined prior on functions and there may be overfitting in the absence of parameter marginalization. Further investigation is merited to tease apart this discrepancy.

4.3 Multi-Task Warping

Experimental setup In this experiment, we apply multi-task warping to logistic regression and online LDA (Hoffman et al., 2010) in a similar manner to Swersky et al.

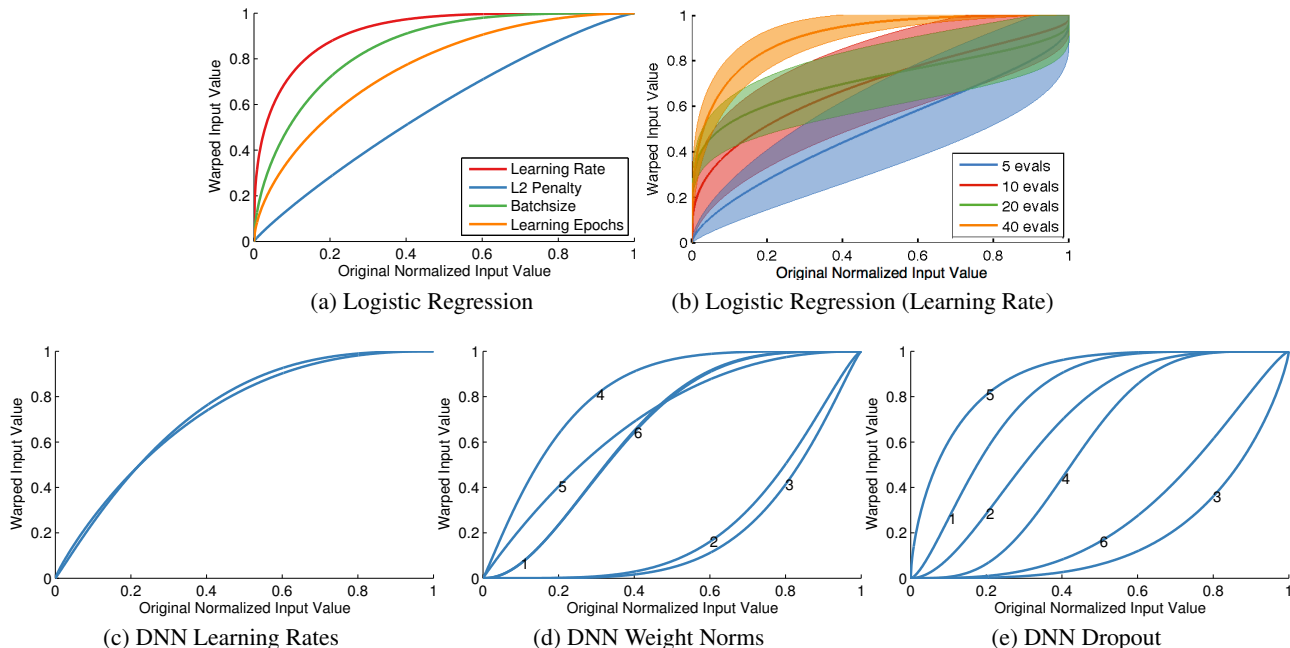


Figure 4. Example input warpings learned for the logistic regression problem (Figures 4a,4b) and the parameters of the deep convolutional neural network (Figures 4c, 4e, 4d). Each plot shows the mean warping, averaged over 100 samples, of each of the parameters. Figure 4b shows the warping learned on the learning rate parameter for logistic regression with different numbers of observations, along with the standard deviation. Each curve in Figures 4d and 4e is annotated with the depth of the layer that each parameter is applied to.

(2013). In the logistic regression problem, a search over hyperparameters has already been completed on the USPS dataset, which consists of 6,000 training examples of handwritten digits of size 16×16 . It was demonstrated that it was possible to use this previous search to speed up the hyperparameter search for logistic regression on the MNIST dataset, which consists of 60,000 training examples of size 28×28 .

In the online LDA problem, we assume that a model has been trained on 50,000 documents and that we would now like to train one on 200,000 documents. Again, it was shown that it is possible to transfer information over to this task, resulting in more efficient optimization.

Results In Figure 5 we see that warped multi-task Bayesian optimization (warped MTBO) outperforms multi-task Bayesian optimization (MTBO) without warping, and performs far better than single-task Bayesian optimization (STBO) that does not have the benefit of a prior search. On logistic regression it appears that ordinary MTBO gets stuck in a local minimum, while warped MTBO is able to consistently escape this by the 20th function evaluation.

In Figure 5a we show the mean warping learned for each task/hyperparameter combination (generated by averaging

over samples from the posterior). The warping of the L_2 penalty on the USPS model favours configurations that are toward the higher end of the range. Conversely, the warping on the MNIST dataset favours relatively lower penalties. This agrees with intuition that a high regularization with less data is roughly equivalent to low regularization with more data. Other observations also agree with intuition. For example, since USPS is smaller each learning epoch consists of fewer parameter updates. This can be offset by training for more epochs, using smaller minibatch sizes, or increasing the learning rate relative to the same model on MNIST.

5 Conclusion

In this paper we develop a novel formulation to elegantly model non-stationary functions using Gaussian processes that is especially well suited to Bayesian optimization. Our approach uses the cumulative distribution function of the Beta distribution to warp the input space in order to remove the effects of mild input-dependent length scale variations. This approach allows us to automatically infer a variety of warpings in a computationally efficient way. In our empirical analysis we see that an inability to model non-stationary functions is a major weakness when using stationary kernels in the GP Bayesian optimization framework. Our sim-

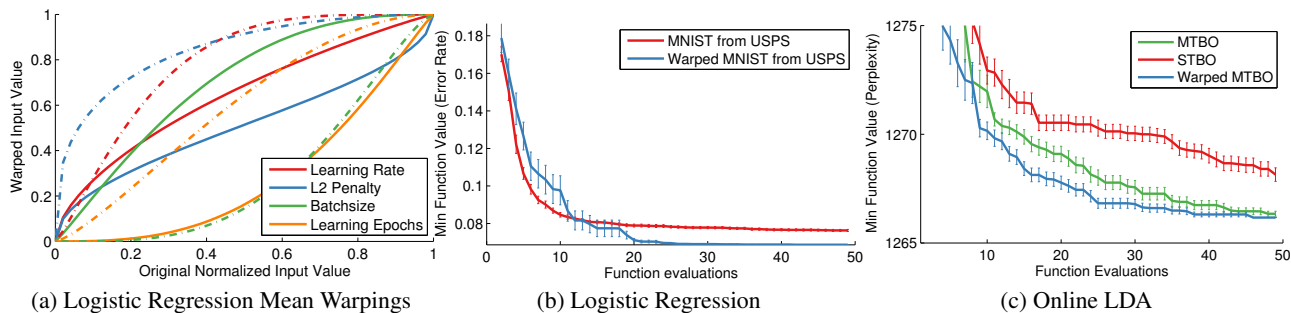


Figure 5. Multi-task warping applied to logistic regression and online LDA. For logistic regression, the model is trained on USPS first and then the search is transferred to MNIST. For online LDA, the data is subsampled and the model is learned, then the search transferred to the full dataset. In both cases, the warped version substantially outperforms multi-task Bayesian optimization with no warping. In Figure 5a, we show the mean warping learned by each task for each parameter. The solid lines indicates the MNIST task, while the dashed lines indicate the USPS task.

ple approach to learn the form of the non-stationarity significantly outperforms the standard Bayesian optimization routine of Snoek et al. (2012) both in the number of evaluations it takes to converge and the value reached. As an additional bonus, the method finds good solutions more reliably. Our experiments on the continuous subset of the HPOLib benchmark (Eggenberger et al., 2013) shows that input warping performs substantially better than state-of-the-art baselines on these problems.

A key advantage of our approach is that the learned transformations can be analyzed *post hoc*, and our analysis of a convolutional neural network architecture leads to surprising insights that challenge established doctrine. Post-training analysis is becoming a critical component of neural network development. For example, the winning Imagenet 2013 (Deng et al., 2009) submission (Zeiler and Fergus, 2013) used *post hoc* analysis to correct for model defects. The development of interpretable Bayesian optimization strategies can provide a unique opportunity to facilitate this kind of interaction. An interesting follow-up would be to determine whether consistent patterns emerge across architectures, datasets and domains.

In Bayesian optimization, properly characterizing uncertainty is just as important as making predictions. GPs are ideally suited to this problem because they offer a good balance between modeling power and computational tractability. In many real world problems, however, the assumptions made by the Gaussian processes are often violated, nullifying many of their benefits. In light of this, many opt to use frequentist models instead, which offer minimax-type guarantees. Our emphasis in this work is to demonstrate that it is possible to stay within the Bayesian framework and thus enjoy its characterization of uncertainty, while still overcoming some of the limitations associated with the conventional GP approach. In future work we intend to experiment with more elaborate models of non-stationarity to see

if these yield further improvements.

Acknowledgements

The authors would like to thank Nitish Srivastava for providing help with the Deepnet package. Jasper Snoek is a fellow in the Harvard Center for Research on Computation and Society. During his time at the University of Toronto, Jasper Snoek was supported by a grant from Google. This work was funded by DARPA Young Faculty Award N66001-12-1-4219, an Amazon AWS in Research grant, the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Canadian Institute for Advanced Research (CIFAR).

References

- Michael A. Osborne, Roman Garnett, and Stephen J. Roberts. Gaussian processes for global optimization. In *Learning and Intelligent Optimization*, 2009.
- Eric Brochu, Tyson Brochu, and Nando de Freitas. A Bayesian interactive optimization approach to procedural animation design. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2010.
- Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. In *International Conference on Machine Learning*, 2010.
- Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization 5*, 2011.
- James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Bálázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*. 2011.
- Adam D. Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, (3-4): 2879–2904, 2011.

- Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, 2012.
- Philipp Hennig and Christian J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13, 2012.
- Carl E. Rasmussen and Christopher Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.
- Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Multi-task bayesian optimization. In *Advances in Neural Information Processing Systems*, 2013.
- K. Eggenberger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *Advances in Neural Information Processing Systems Workshop on Bayesian Optimization in Theory and Practice*, 2013.
- Robert B. Gramacy. *Bayesian treed Gaussian process models*. PhD thesis, UC Santa Cruz, 2005.
- D. Higdon, J. Swall, and J. Kern. Non-stationary spatial modeling. *Bayesian Statistics*, 6, 1998.
- Paul D. Sampson and Peter Guttorp. Nonparametric estimation of nonstationary spatial covariance structure. *Journal of the American Statistical Association*, 87(417):108–119, 1992.
- Alexandra M. Schmidt and Anthony O’Hagan. Bayesian inference for nonstationary spatial covariance structures via spatial deformations. *Journal of the Royal Statistical Society Series B*, 65:743–758, 2003.
- Ethan B. Anderes and Michael L. Stein. Estimating deformations of isotropic gaussian random fields on the plane. *The Annals of Statistics*, 36(2):719–741, 04 2008.
- Luke Bornn, Gavin Shaddick, and James V. Zidek. Modeling nonstationary processes through dimension expansion. *Journal of the American Statistical Society*, 107(497), 2012.
- Edward Snelson, Carl Edward Rasmussen, and Zoubin Ghahramani. Warped Gaussian processes. In *Advances in Neural Information Processing Systems*, 2003.
- Ryan P. Adams and Oliver Stegle. Gaussian process product models for nonparametric nonstationarity. In *International Conference on Machine Learning*, 2008.
- Pierre Goovaerts. *Geostatistics for Natural Resources Evaluation*. Oxford University Press, 1997.
- Mauricio A Alvarez and Neil D Lawrence. Computationally efficient convolved multiple output gaussian processes. *Journal of Machine Learning Research*, 12, 2011.
- Yee Whye Teh, Matthias Seeger, and Michael I. Jordan. Semi-parametric latent factor models. In *International Conference on Artificial Intelligence and Statistics*, 2005.
- Edwin V. Bonilla, Kian Ming A. Chai, and Christopher K. I. Williams. Multi-task Gaussian process prediction. In *Advances in Neural Information Processing Systems*, 2008.
- Michael A. Osborne. *Bayesian Gaussian Processes for Sequential Prediction, Optimisation and Quadrature*. PhD thesis, University of Oxford, 2010.
- José C. Pinheiro and Douglas M. Bates. Unconstrained parameterizations for variance-covariance matrices. *Statistics and Computing*, 6:289–296, 1996.
- Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2, 1978.
- Dan Lizotte. *Practical Bayesian Optimization*. PhD thesis, University of Alberta, Edmonton, Alberta, 2008.
- H. J. Kushner. A new method for locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86, 1964.
- Matthew Hoffman, Eric Brochu, and Nando de Freitas. Portfolio allocation for Bayesian optimization. In *Uncertainty in Artificial Intelligence*, 2011.
- Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21, 2001.
- Andreas Krause and Cheng Soon Ong. Contextual Gaussian process bandit optimization. In *Advances in Neural Information Processing Systems*, 2011.
- Rémi Bardenet, Mátyás Brendel, Balázs Kégl, and Michèle Sebag. Collaborative hyperparameter tuning. In *International Conference on Machine Learning*, 2013.
- Iain Murray and Ryan P. Adams. Slice sampling covariance hyperparameters of latent Gaussian models. In *Advances in Neural Information Processing Systems*. 2010.
- Alex Krizhevsky. Learning multiple layers of features from tiny images. *Technical report, Department of Computer Science, University of Toronto*, 2009.
- Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.
- Matthew Hoffman, David M. Blei, and Francis Bach. Online learning for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems*, 2010.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, 2009.
- Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional neural networks. *arXiv preprint arXiv:1311.2901*, 2013.