Variational Hilbert Regression with Applications to Terrain Modeling

Vitor Guizilini and Fabio Ramos

Abstract The ability to generate accurate terrain models is of key importance in a wide variety of robotics tasks, ranging from path planning and trajectory optimization to environment exploration and mining applications. This paper introduces a novel regression methodology for terrain modeling that takes place in a Reproducing Kernel Hilbert Space, and can approximate arbitrarily complex functions using Variational Bayesian inference. A sparse kernel is used to efficiently project input points into a high-dimensional feature vector, based on cluster information generated automatically from training data. Each kernel maintains its own regression model, and the entire set is simultaneously optimized in an iterative fashion as more data is collected, to maximize a global variational bound. Additionally, we show how kernel parameters can be jointly learned alongside the regression model parameters, to achieve a better approximation of the underlying function. Experimental results show that the proposed methodology consistently outperforms current state-of-the-art techniques, while maintaining a fully probabilistic treatment of uncertainties and high scalability to large-scale datasets.

1 Introduction

Any sort of autonomous task is predicated on a system's ability to sense its surroundings, as a way to understand its current state before deciding what its next action should be in order to achieve a determined goal. For mobile robot navigation, an accurate model for terrain representation is crucial, since it dictates which areas the vehicle can traverse safely and which ones would result in a collision, or some other sort of unfavorable outcome. The main challenges in creating an accurate terrain model lie in the presence of measurement uncertainties, data incompleteness and

Vitor Campanholo Guizilini and Fabio Tozeto Ramos
School of Information Technologies, The University of Sydney, Australia.
e-mail: {vitor.guizilini;fabio.ramos}@sydney.edu.au

handling details in unstructured areas, which becomes even more challenging in large-scale scenarios, which is the case for most current applications. Any sensor is inherently inaccurate, and these inaccuracies are compounded by uncertainty in vehicle localization, as new data is collected during navigation to produce a global representation of observed areas. Data point density rapidly decays as we move away from the sensor, creating low-resolution areas in which large portions of the environment within range remain unobserved. Terrain details (i.e. sharpness and sudden discontinuities) should be correctly captured for an accurate representation of observed data, however if the model is too complex it might overfit and produce poor estimates under testing conditions.

In robotics, state-of-the-art terrain representations were originally limited to tools such as elevation maps (EMs) and triangulated irregular networks (TINs). EMs [15, 34] are 2D grid-based structures that model 3D space in terms of a scalar estimate corresponding to the height of each cell, an arrangement also known as a 2.5D representation. Their main advantage is simplicity, however this comes at the cost of drawbacks such as: inability to handle abrupt elevation changes [35]; lack of a statistically proper way to handle spatial correlation or measurement uncertainty [16]; and scalability issues to larger datasets [9]. Over the years, these drawbacks have been partially addressed in several ways, such as [34], that proposes an extension capable of handling multiple surfaces and hanging objects, and [16, 7], that attempts to incorporate uncertainty estimates in the EM model. TINs [28], on the other hand, sample a set of points that capture important aspects of the observed terrain surface, which are then connected to their nearest neighbors to produce a triangular network model. While better able to capture sudden elevation changes and more efficient when dealing with flat areas, TINs struggle with dense sensor data [34], due to the huge memory footprint produced by highly textured surfaces, and are still unable to formally incorporate spatial dependencies and measurement uncertainty.

The use of Gaussian Processess (GPs) [27], a powerful Bayesian non-parametric regression learning technique, for terrain modeling has become popular in recent years, because they are able to handle most of the above mentioned issues. They provide a continuous elevation function that can be sampled at arbitrary resolutions, incorporate uncertainty in an statistically sound way and represent spatially correlated data in an appropriate manner. Essentially, a GP implements an interpolation technique commonly used in statistics, called *Krigging* [18], which is the best linear unbiased estimate based on the underlying stochastic model of spatial correlation between data points (further discussion on other interpolation techniques for grid-based methods can be found in [14]). The standard GP framework, however, scales cubically with the number of training points, thus rendering it computationally too expensive for large-scale datasets, and most kernel covariance functions are too smooth to correctly model terrain details and sudden discontinuities. Non-stationary covariance functions [35, 22] can be used to produce variable smoothness throughout the input space, and sparse approximations [25, 33, 23] reduce computational complexity by projecting data into a subset of inducing points, or using GP ensembles.

In this paper we propose a novel regression methodology for terrain modeling that operates on a Reproducing Kernel Hilbert Space (RKHS) [30], generated by the projection of input data into a high-dimensional feature vector. In this high-dimensional space, real world complexity can be represented using very simple models, which results in efficient training and query times. Although this approach has already been explored before, most notably in [36] for a functional regression setting and in [12] for rough terrain estimation using space-carving kernels, here we develop a fully probabilistic sparse approximation that is suitable for online learning and can be readily applied to large-scale datasets. Variational Bayesian inference is used for the probabilistic treatment of variables, thus providing a measure of uncertainty that quantifies the model's confidence in relation to its own estimates. The compact support kernel introduced in [10] serves to decouple different dimensions of the RKHS, which enforces non-stationarity in the input space and allows the optimization of different localized regression models, while still maximizing a single global variational bound. Furthermore, we show how kernel parameters can be jointly learned alongside regression model parameters, to achieve a better variational approximation of the underlying function. Finally, we test the proposed methodology in two large-scale unstructured real datasets, and show that it outperforms other commonly used terrain modeling techniques both in terms of accuracy and computational speed.

2 Methodology

We assume a training dataset $\mathcal{D} = \{X, \mathbf{y}\}$ composed of N observed input points $X = \{\mathbf{x}_n\}_{n=1}^N$, with $\mathbf{x}_n \in \mathcal{R}^D$, and their corresponding output values $\mathbf{y} = \{y_n\}_{n=1}^N$, with $y_n \in \mathcal{R}$. The objective is to produce a continuous function $g(.)$ that approximates a hypothetical underlying latent function $f(.)$, from which $\mathcal{D}$ was sampled. This latent function can be defined as:

$$y_n = f(\mathbf{x}_n) + \varepsilon = \mathbf{w}^T \mathbf{x}_n + \varepsilon \quad , \tag{1}$$

where ε is the system noise and $\mathbf{w}$ is the parameter vector, that is optimized based on the information contained in $\mathcal{D}$ and defines the regression model. For a standard linear regression model, $\mathbf{w}$ is a $D+1$ column vector and $\mathbf{x}_n$ is augmented so that $\tilde{\mathbf{x}}_n = \{1, x_n^1, x_n^2, \ldots, x_n^D\}$. The use of basis functions can further improve its predictive powers [2], however for notation simplicity here we assume a non-biased linear regression model.

2.1 Bayesian Inference

We start by modeling ε as a zero-mean Gaussian distribution with variance σ_ε^2, i.e. $\varepsilon \sim \mathcal{N}(\mathbf{0}, \sigma_\varepsilon^2)$. The probability of an output y, given the corresponding input $\mathbf{x}$, model parameters $\mathbf{w}$ and noise variance σ_ε, is given by:

$$p(y|\mathbf{x}, \mathbf{w}, \sigma_\varepsilon^2) = \mathcal{N}(y|\mathbf{w}^T\mathbf{x}_n, \sigma_\varepsilon^2) = \frac{1}{\sqrt{2\pi\sigma_\varepsilon^2}} \exp\left(\frac{1}{2\sigma_\varepsilon^2}(\mathbf{y}_n - \mathbf{w}^T\mathbf{x}_n)^2\right) \quad . \tag{2}$$

Furthermore, if we consider the entire training dataset $\mathcal{D}$ and assume they are sampled independently from $f(.)$, the log-likelihood function can be expressed as:

$$\ln p(\mathbf{y}|X, \mathbf{w}, \sigma_\varepsilon^2) = \prod_{n=1}^{N} \ln\mathcal{N}(y|\mathbf{w}^T\mathbf{x}_n, \sigma_\varepsilon^2) = \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E(\mathbf{w}) \quad , \tag{3}$$

where $\beta = \sigma_\varepsilon^{-2}$ is the noise precision and $E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(y_n - \mathbf{w}^T\mathbf{x}_n\right)^2$ is the sum of squared errors loss function. Based on the above equation, we can conclude that maximizing this log-likelihood is equivalent to minimizing the sum of squared errors between model estimates $\mathbf{w}^T\mathbf{x}_n$ and ground-truth information y_n. The gradient of Equation 3 is given by:

$$\nabla \ln p(\mathbf{y}|X, \mathbf{w}, \sigma_\varepsilon^2) = \sum_{n=1}^{N}\left(y_n - \mathbf{w}^T\mathbf{x}_n\right)\mathbf{x}_n^T \quad . \tag{4}$$

Setting this gradient to zero and solving for $\mathbf{w}$, we obtain the optimal parameters as $\mathbf{w}_{ML} = (X^TX)^{-1}X^T\mathbf{y}$. However, this direct optimization has the known effect of inducing over-fitting if model complexity is not carefully selected [2], since simply maximizing the likelihood function will always lead to excessively complex models. Because of that, usually a Bayesian formulation is used [4], that is able to automatically determine model complexity based on training data alone. This is done by placing a prior distribution on the model parameters themselves, such that $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}_0, S_0)$. To facilitate the treatment, we can assume a zero-mean distribution and a single precision parameter α, so that $p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}I)$ The posterior distribution $p(\mathbf{w}|\mathbf{y}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}_t, S_t)$, calculated using the information contained in $\mathcal{D}$, has the following mean and variance values:

$$\boldsymbol{\mu}_t = S_t\left(S_{t-1}^{-1}\boldsymbol{\mu}_{t-1} + \beta X^T\mathbf{y}\right) = \beta S_t X^T\mathbf{y} \tag{5}$$

$$S_t^{-1} = S_{t-1}^{-1} + \beta X^TX = \alpha I + \beta X^TX \quad . \tag{6}$$

It can be shown [2] that maximizing this new formulation with respect to $\mathbf{w}$ is equivalent to minimizing the sum of squared errors with the addition of a quadratic regularization term $\frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$, with $\lambda = \alpha/\beta$.

2.2 Variational Bayesian Approximation

The explicit calculation of these distributions quickly becomes intractable as models get more complex, particularly when hyper-parameters such as α and β are integrated alongside model parameters $\mathbf{w}$. Because of that, considerable work has been done in approximation techniques, such as Markov Chain Monte Carlo [8] and Variational

Bayesian [21, 2, 6] methods. Here we will focus on Variational Bayesian methods, that introduce a new distribution $q(.)$ that is optimized to approximate $p(.)$, such that $p(Z|X) \approx q(Z)$ according to a distance metric $D(q,p)$, usually the Kullback-Leibler (KL) divergence, defined as:

$$D_{KL}(q||p) = \sum_{Z} q(Z) \log \frac{q(Z)}{p(Z|X)} \quad . \tag{7}$$

The prior on $\mathbf{w}$ and β is a conjugate normal inverse-gamma distribution parameterized on α, defined as $p(\mathbf{w},\beta|\alpha) = \mathcal{N}(\mathbf{w}|0,(\alpha\beta)^{-1}I)\mathrm{Gam}(\beta|a_0,b_0)$. Similarly, α receives the hyper-prior $p(\alpha) = \mathrm{Gam}(\alpha|c_0,d_0)$. There is no analytic solution to these posteriors, due to the introduction of hyper-priors, so a variational treatment becomes necessary. The variational bound to be maximized is:

$$\mathcal{L}(q) = \int\int\int q(\mathbf{w},\alpha,\beta) \ln \frac{p(\mathbf{y}|X,\mathbf{w},\beta)p(\mathbf{w},\beta|\alpha)p(\alpha)}{q(\mathbf{w},\alpha,\beta)} d\mathbf{w} d\alpha d\beta \leq \ln p(\mathcal{D}) , \tag{8}$$

where $p(\mathcal{D})$ is the model evidence. To maximize this bound, we assume the variational distribution $q(\mathbf{w},\alpha,\beta)$, which approximates the posterior $p(\mathbf{w},\alpha,\beta|\mathcal{D})$, can be factored as $q(\mathbf{w},\beta)q(\alpha)$. The variational posterior for $\mathbf{w}$ and β that maximizes this variational bound, while keeping $q(\alpha)$ fixed, is given by:

$$\ln q^*(\mathbf{w},\beta) = \ln \mathcal{N}(\mathbf{w}|\mathbf{w}_t,\beta^{-1}V_t)\mathrm{Gam}(\beta|a_t,b_t) \quad , \tag{9}$$

with:

$$\begin{gathered} a_t = a_{t-1} + \frac{N}{2} \quad , \quad b_t = b_{t-1} + \frac{1}{2}\left(\sum_n (y_n - \mathbf{w}_n^T \mathbf{x}_n)^2 + \frac{c_t}{d_t}\mathbf{w}_t^T \mathbf{w}_t\right) \quad , \\ V_t^{-1} = \frac{c_t}{d_t} I + \sum_n \mathbf{x}_n \mathbf{x}_n^T \quad , \quad \mathbf{w}_t = V_t \sum_n \mathbf{x}_n y_n \quad . \end{gathered} \tag{10}$$

Similarly, the variational posterior for α, while keeping $q(\mathbf{w},\beta)$ fixed, is given by:

$$\ln q^*(\alpha) = \ln \mathrm{Gam}(\alpha|c_t,d_t) \quad , \tag{11}$$

with:

$$c_t = c_{t-1} + \frac{D}{2} \quad , \quad d_t = d_{t-1} + \frac{1}{2}\left(\frac{a_t}{b_t}\mathbf{w}_t^T \mathbf{w} + \mathrm{Tr}(V_t)\right) \quad . \tag{12}$$

The complete variational bound can then be rearranged to assume the form:

$$\begin{aligned} \mathcal{L}(q)_t = & -\frac{N}{2}\ln 2\pi - \frac{1}{2}\sum_n \left(\frac{a_t}{b_t}(y_n - \mathbf{w}_t^T \mathbf{x}_n)^2 + \mathbf{x}_n^T V_t \mathbf{x}_n\right) + \frac{1}{2}\ln|V_t| + \frac{D}{2} \\ & -\ln\Gamma(a_{t-1}) + a_{t-1}\ln b_{t-1} - b_{t-1}\frac{a_t}{b_t} + \ln\Gamma(a_t) - a_t \ln b_t + a_t \\ & -\ln\Gamma(c_{t-1}) + c_{t-1}\ln d_{t-1} + \ln\Gamma(c_t) - c_t \ln d_t \quad . \end{aligned} \tag{13}$$

This bound is maximized by iteratively updating over V_t, $\mathbf{w}_t$, a_t, b_t, c_t and d_t until $\mathcal{L}(q)$ converges. The predictive distribution is evaluated by approximating the posterior $p(\mathbf{w},\beta|\mathcal{D})$ by its variational counterpart $q(\mathbf{w},\beta)$, such that:

$$p(y|\mathbf{x},\mathcal{D}) \approx \mathrm{St}\left(y|\mathbf{w}_t^T\mathbf{x},(1+\mathbf{x}^T V_t\mathbf{x})^{-1},2a_t\right) \quad , \tag{14}$$

which is a Student's t-distribution with mean $\mathbf{w}_t^T\mathbf{x}$, precision $(1+\mathbf{x}^T V_t\mathbf{x})^{-1}a_t/b_t$ and $2a_t$ degrees of freedom. This variational Bayesian formulation lends itself naturally to online learning, since new information can be incorporated as the iterative process described previously is performed. Several extensions that further improve online learning have also been proposed, such as a forgetting scheme that allows its use in dynamic environments [29, 13] and asynchronous streaming support [3].

2.3 Variational Hilbert Regression

The Hilbert Maps (HM) framework, introduced in [26], proposes the modeling of arbitrarily complex functions by projecting input coordinates $\mathbf{x}$ into a *Reproducing Kernel Hilbert Space* (RKHS) [31], using a feature vector $\Phi(\mathbf{x})$. The dot product of these feature vectors can be used to approximate popular kernels found in the literature [27], and by operating only in terms of kernel evaluations it is possible to avoid calculations in this high-dimensional (and possibly infinite) feature space. The original implementation, alongside subsequent extensions [10, 5, 11], focus on classification tasks, such as occupancy mapping, and here we propose what is to the best of our knowledge the first application of the Hilbert Maps framework for regression tasks.

In [10] a sparse derivation was developed, which used clustering to place inducing points throughout observed areas of the input space, that served as anchors for different dimensions of the feature vector. Sparseness is achieved using the kernel proposed in [19], which vanishes exactly to zero after a certain distance threshold. The feature vector $\Phi(\mathbf{x})$ and sparse kernel $k(\mathbf{x},\mathcal{M})$ are defined as:

$$\Phi(\mathbf{x},\mathcal{M}) = \begin{bmatrix} k(\mathbf{x},\mathcal{M}_1) \\ k(\mathbf{x},\mathcal{M}_2) \\ \vdots \\ k(\mathbf{x},\mathcal{M}_M) \end{bmatrix} \quad , \quad \begin{array}{l} k(\mathbf{x},\mathcal{M}_i) = \\ \begin{cases} \frac{2+\cos(2\pi r_i)}{3}(1-r_i)+\frac{1}{2\pi}\sin(2\pi r_i) & \text{if } r_i < 1 \\ 0 & \text{if } r_i \geq 1 \end{cases} \end{array} \quad , \tag{15}$$

where $\mathcal{M} = \{\mathcal{M}_i\}_{i=1}^M = \{\boldsymbol{\mu}_i,\Sigma_i\}_{i=1}^M$ is the cluster set extracted from $\mathcal{D}$, each one containing mean $\boldsymbol{\mu}$ and covariance Σ values calculated from their corresponding input points subset and used to obtain the distance threshold $r_i = \sqrt{(\mathbf{x}-\boldsymbol{\mu}_i)^T\Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)}$. Efficient data storage structures, such as kd-trees [20], can be used to quickly search for clusters within this threshold, thus eliminating irrelevant kernel calculations.

For the proposed regression methodology, each cluster is augmented to include a parameter vector $\boldsymbol{\theta} = \{\mathbf{w}, V, a, b, c, d\}$, such that $\mathcal{M}_i = \{\boldsymbol{\mu}, \Sigma, \boldsymbol{\theta}\}_i$. Note that each cluster has its own regression model, that is initialized empty (i.e. with zeros) and updated as new information is observed. Assuming that each new observation $\{\mathbf{x}, y\}_*$ is drawn independently from the underlying function $f(.)$ and belongs to a single cluster, the global variational bound can be expressed as the sum $\mathcal{L}(\mathcal{M}) = \sum_i \mathcal{L}(q_i)$ (i.e. the feature vector $\Phi(\mathbf{x}_*, \mathcal{M})$ is orthogonal), with each local component described by Equation 13. To enforce sparsity during updates, and maintain non-stationarity in different areas of the input space, these components are scaled by the corresponding feature vector $\Phi(\mathbf{x}_*, \mathcal{M})$, so that:

$$\mathcal{L}(\mathcal{M})_{t+1} = \mathcal{L}(\mathcal{M})_t + \max_{\Theta} \left[\sum_{i=1}^{M} \Phi_i(\mathbf{x}_*, \mathcal{M}) \Big(\mathcal{L}(\mathcal{M}_i)_{t+1} - \mathcal{L}(\mathcal{M}_i)_t \Big) \right] , \qquad (16)$$

where Θ represents all model parameters $\{\boldsymbol{\theta}\}_{i=1}^{M}$, that are optimized to maximize the increase in the global variational bound from timestep t (prior to new data incorporation) to $t+1$ (after new data incorporation). The intuition is that points closer to a cluster should contribute with a larger rate of change for that particular model. Similarly, parameters of clusters with no points within its distance threshold (i.e. $\Phi_i(\mathbf{x}, \mathcal{M}) = 0$) remain unchanged, since there is no relevant information to be incorporated. Note that multiple points can be incorporated simultaneously, by optimizing Equation 16 with the summed contribution of each new observation scaled by their corresponding feature vector.

Prediction is performed in a similar way, using the approximated distribution in Equation 14 to generate mean μ_* and variance v_*^2 estimates for any query point $\mathbf{x}_*$. Each cluster contributes with one estimate, that is scaled according to its corresponding feature vector $\Phi(\mathbf{x}, \mathcal{M})$ dimension and normalized by the total sum of the feature vector, so that nearby clusters contribute more strongly to the final estimate:

$$\mu_* = \frac{1}{||\Phi(\mathbf{x}_*, \mathcal{M})||_1} \sum_{i=1}^{M} \Phi(\mathbf{x}_*, \mathcal{M}_i) \mathbf{w}_i^T \mathbf{x}_* \qquad (17)$$

$$v_*^2 = \frac{1}{||\Phi(\mathbf{x}_*, \mathcal{M})||_1} \sum_{i=1}^{M} \Phi(\mathbf{x}_*, \mathcal{M}_i) \frac{b_i}{a_i} (1 + \mathbf{x}_*^T V_i \mathbf{x}_*) \qquad (18)$$

A consequence of the sparseness enforced by Φ is that query points outside the distance threshold of all clusters will not be considered by any model, which results in an ill-defined behavior in Equations 17 and 18. A solution to this scenario is to set a minimum number of nearest clusters whose models will be considered, regardless of their distance thresholds. In the experiments we used a simple heuristic in which $r_i/\bar{r}$ nearest neighbors are considered, where r_i is the distance threshold to the nearest cluster and $\bar{r}$ is the average distance between clusters. This formulation ensures that more clusters are used to estimate areas further away, thus producing smoother results that do not depend on any model in particular, while nearby regions still retain their sharpness, since fewer clusters are used for estimates.

2.4 Joint Kernel Learning

Given an initial dataset $\mathcal{D}$, several different clustering techniques [17, 1, 32] can be used to generate cluster centers, and by extension covariance values. In this work we employ the ASK-Means algorithm described in [11], because it produces similar cluster densities and can automatically determine how many clusters are necessary to properly describe input data. Once the kernel parameters $\{\boldsymbol{\mu}, \Sigma\}_{i=1}^{M}$ are generated, the methodology described previously is used to separately learn the regression model parameters $\Theta = \{\boldsymbol{\theta}\}_{i=1}^{M}$ that together compose the Variational Hilbert Regression model.

In this section we briefly discuss a few techniques that enable the joint learning of kernel and regression model parameters $\{\boldsymbol{\mu}, \Sigma, \boldsymbol{\theta}\}_{i=1}^{M}$, as a way to further increase $\mathcal{L}(\mathcal{M})$ values during the training process or decrease model complexity without significantly compromising results.

2.4.1 Switching

Switching refers to changing the index of a given training point $\mathbf{x}_k$, so it belongs to a new cluster. By extension, this change also modifies the kernel parameters of both clusters, which can be performed efficiently using routines for incremental calculation of mean and variance [24]. The maximization process for the global variational bound now includes the cluster selection for $\mathbf{x}_k$, and is of the form:

$$\mathcal{L}(\mathcal{C})_{t+1} = \mathcal{L}(\mathcal{C})_t + \max_{\Theta,\mathcal{C}} \left[\sum_{j=1}^{C} \Phi_j(\mathbf{x}, \mathcal{C}) \Big(\mathcal{L}(\mathcal{C}_j)_{t+1} - \mathcal{L}(\mathcal{C}_j)_t \Big) \right] , \tag{19}$$

where $\mathcal{C}$ represents the subset of clusters considered to receive $\mathbf{x}_k$ (note that the remainder of $\mathcal{M}$ is not considered here, since these clusters are not affected by the switching process). As a way to avoid excessive computational costs, switching is triggered by calculating the residual error $\zeta_k^i = (\mu_k^i - y_k)^2$ of $\mathbf{x}_k$ in relation to its current cluster $\mathcal{M}_i$. If this value is above a certain error threshold, its nearby clusters (i.e. within distance threshold) are considered as the subset $\mathcal{C}$ in Equation 19.

2.4.2 Splitting

Splitting refers to separating one cluster into two or more, each one receiving a subset of its corresponding points. This is particularly useful when dealing with more complex areas, with shapes that cannot be accurately captured by the regression model.[1] The splitting process does not affect any other clusters, and therefore its

[1] The proposed framework can be trivially extended to support different regression models for each cluster, including the learning of which regression models should be used, however this was not explored in this paper.

maximization process can be simplified to:

$$\mathcal{L}(\mathcal{M}_k)_{t+1} = \mathcal{L}(\mathcal{M}_k)_t + \max_{\Theta,\mathcal{C}} \left[\sum_{j=1}^{C} \Phi_j(\mathbf{x},\mathcal{C}) \Big(\mathcal{L}(\mathcal{C}_j)_{t+1} - \mathcal{L}(\mathcal{M}_k)_t \Big) \right] , \tag{20}$$

where $\mathcal{M}_k$ denotes the cluster to be split, and $\mathcal{C}$ is the subset of clusters produced from its corresponding points. Note that this process introduces a new dimension to the feature vector Φ, thus increasing the overall complexity of the underlying model. To avoid over-fitting, new clusters are only generated if assigned a minimum number of points, which can be calculated based on average point density for different areas of the input space. As a way to avoid excessive computational costs, splitting is triggered by calculating the average residual value $\bar{\zeta}^i = \frac{1}{k} \sum_k \zeta_k^i$ of all k points in $\mathcal{M}_i$ in relation to its own regression model. If this value is above a certain threshold, its points are considered for the generation of sub-clusters $\mathcal{C}$ in Equation 20.

2.4.3 Merging

Merging refers to combining two or more clusters into a single one, with the same kernel and regression parameters. Differently from previous techniques, merging does not attempt to maximize the global variational bound, since it decreases the complexity of the underlying model. However, it can determine which clusters can be safely removed without degrading estimates above a given error threshold, thus promoting a trade-off between accuracy and speed that can be useful for certain applications. Pseudo-code for the checking and merging of a cluster $\mathcal{M}_i$, given an error threshold ζ_{max}, is shown in Algorithm 1, and its effects on terrain mapping results can be seen in Figure 3.

Algorithm 1: Pseudo-Code for cluster merging within the VHR framework.

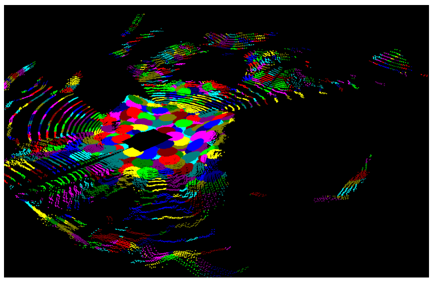
```
input  : $\mathcal{M}$ : Cluster Set
         $i$ : Index of cluster considered for merging
output : $\mathcal{M}$ : Updated cluster set
1 $\mathcal{C} \leftarrow nearest(\mathcal{M}_i)$ % Find nearest neighbors
2 for $\mathcal{C}_j$ in $\mathcal{C}$ do
3     $\zeta_i^j = \frac{1}{U}\sqrt{\sum_u \left(\mu_u^j - y_u\right)^2}$ , $\{\mathbf{x},y\}_u \in \mathcal{M}_i$   % Residual error of $\mathcal{M}_i$ in relation to $\mathcal{C}_j$
4     $\zeta_j^i = \frac{1}{V}\sqrt{\sum_v (\mu_v^i - y_v)^2}$ , $\{\mathbf{x},y\}_v \in \mathcal{C}_j$   % Residual error of $\mathcal{C}_j$ in relation to $\mathcal{M}_i$
5     if $\left(\zeta_i^j + \zeta_j^i\right)/2 < \zeta_{max}$ then
6         $\mathcal{M}_i \leftarrow \mathcal{C}_j$   % Merge clusters
7     end
8 end
9 $\{\boldsymbol{\mu}, \Sigma, \boldsymbol{\theta}\} \leftarrow \mathcal{M}_i$   % Update cluster kernel and model parameters
```

3 Experimental Results

In this section the proposed Variational Hilbert Regression (VHR) is used to address the problem of terrain modeling, in which 2D inputs $\mathbf{x} = \{x, y\}_{n=1}^{N}$ are mapped into a probability distribution $f(\mathbf{x}) = \mathcal{N}(\mu, \nu^2)$ for elevation values. Two different large-scale publicly available datasets were considered, with VHR results being compared to both standard linear (LI) and cubic (CI) interpolation techniques and also a GP framework for the modeling of large-scale terrains, as shown in [35].

The first dataset considered here was obtained from the website `http://asrl.utias.utoronto.ca/datasets/3dmap/a100_dome.html`, and is entitled *Rover*. It contains 252616 points obtained from a laser rangefinder mounted on top of a panning unit, to produce 3D slices of the surrounding environment. This dataset covers an area of roughly $30m \times 30m$, however due to terrain obstructions a large portion remains unobserved, as it can be seen in Figure 1a, where the 1589 clusters generated from the original pointcloud are depicted. Figure 1b presents the terrain modeling results obtained using a GP framework, based on the work of [35]. Due to the number of training points, a sparse approximation was necessary [33], in which cluster centers were used as inducing points to decrease computational cost during training and inference. Similarly, terrain modeling results obtained using the proposed VHR framework are depicted in Figure 2, also colored by variance.

As expected, areas close to training points have lower variance values (shades of blue), and as we move away from these observed areas variance values steadily increases (shades of red, until saturation). Interestingly, in the VHR framework variance values vary even in areas close to training points, according to the accuracy of nearby local regression models that contribute to the feature vector. As observed data diverges from current estimates, the resulting variance values increase (i.e. different shades of blue on the bottom left image in Figure 2), which might lead to switching and splitting, as detailed in Sections 2.4.1 and 2.4.2 respectively. It is also


(a) 1589 Clusters generated from the original pointcloud (252616 points).

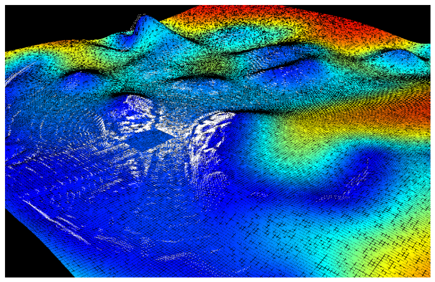

(b) Sparse GP results (mean estimates, colored by variance).

Fig. 1: Terrain modeling results for the *Rover* dataset using a Sparse GP framework. White dots indicate the points used to train the regression model.

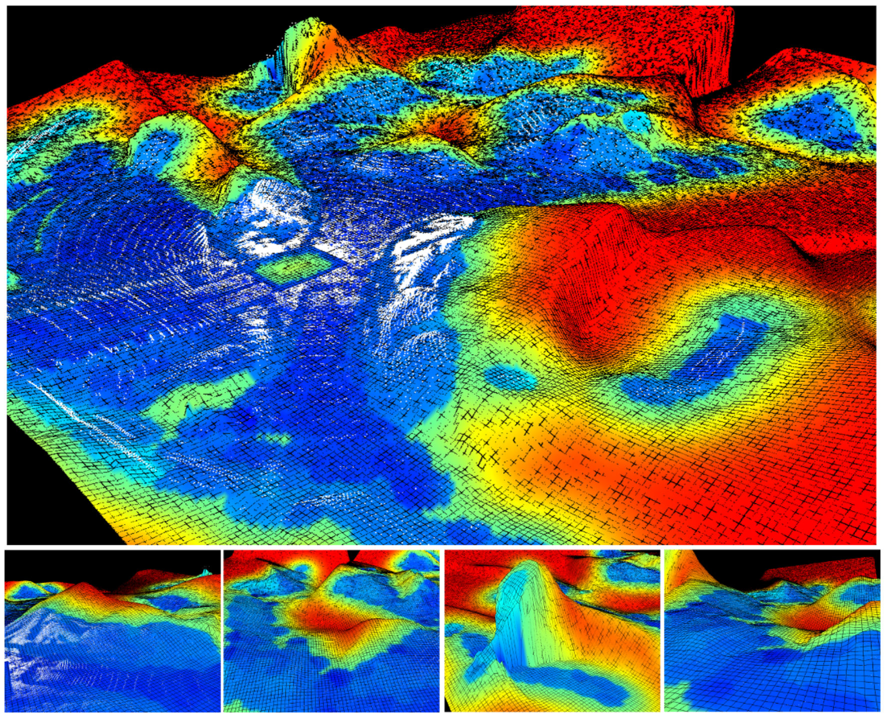
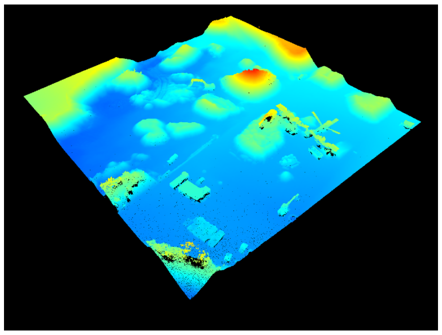


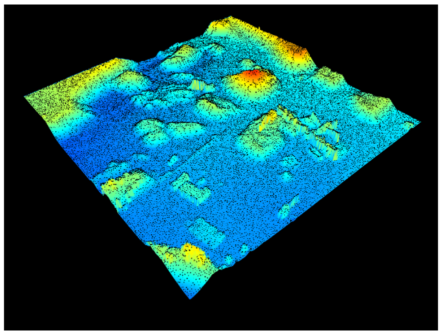
Fig. 2: Terrain modeling results for the *Rover* dataset using the proposed framework (colored by variance). White dots indicate the points used to train the regression model.

worth noting that the model produced by the Sparse GP (SGP) framework is much smoother, and therefore does not capture fine details in well-defined areas of the input space. The localized aspect of the VHR framework, on the other hand, produces variable smoothness levels, that maintains the sharpness in well-defined areas while smoothing out areas with higher variance. Additionally, the VHR framework does not converge back to a single mean value as it moves away from observed data, but maintains the behavior of nearby structures as dictated by their corresponding local regression models, which tends to produce a visually better representation of unobserved terrains.

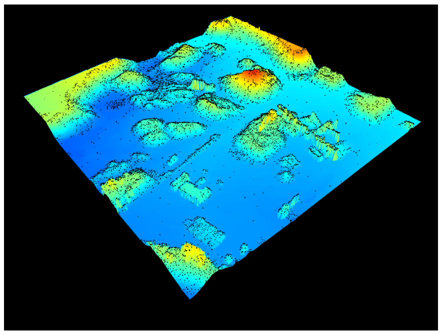
The second dataset, entitled *Aerial*, was obtained from the website `http://www.pointcab-software.com/en/downloads/`. It contains 39460480 points collected by an UAV flying over a construction site, using a camera sensor. It covers an area of roughly $140m \times 140m$, with well-distributed training points throughout the entire input space (Figure 3a). The residual error for each training point, in relation to the trained VHR model, is depicted in Figure 4a, where we can see that most of this residual error is concentrated in a few points (in fact, 90% of it is attributed to 47854 points, only 1.34% of the total). Figure 3 also depicts VHR terrain modeling



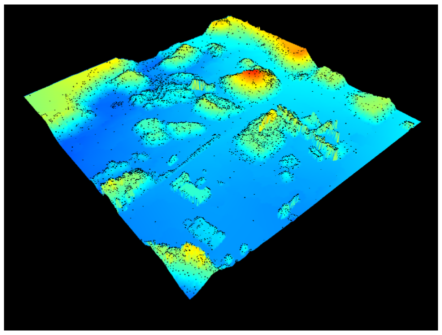
(a) Original pointcloud (39460480 points)



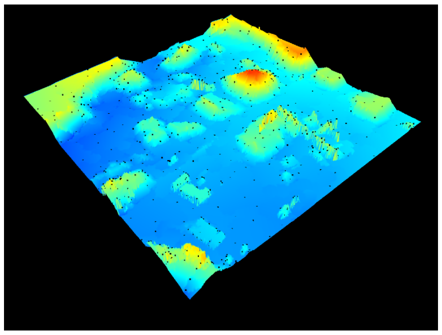
(b) 30895 clusters ($\zeta_{max} = 0.000$)



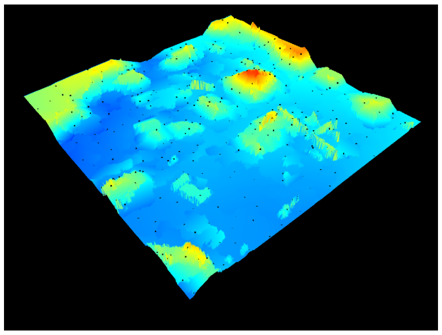
(c) 15131 clusters ($\zeta_{max} = 0.005$)



(d) 9580 clusters ($\zeta_{max} = 0.010$)

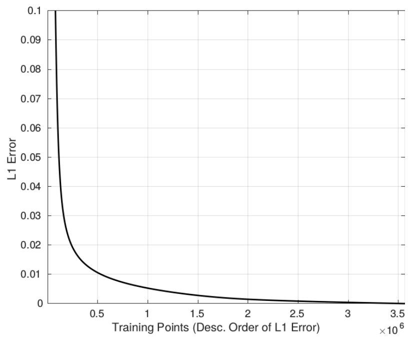


(e) 1082 clusters ($\zeta_{max} = 0.050$)

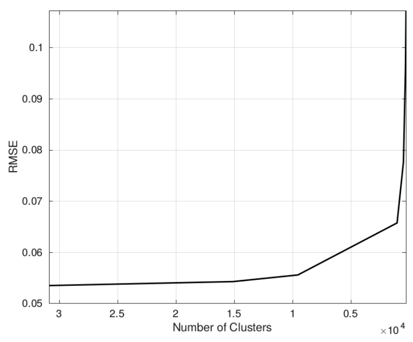


(f) 537 clusters ($\zeta_{max} = 0.100$)

Fig. 3: Terrain modeling results for the *Aerial* dataset using the proposed VHR framework (colored by height), with different number of clusters. Black dots indicate cluster centers.



(a) L1 error for training points, in descending order.



(b) RMSE with different number of clusters.

Fig. 4: Training error results under different circumstances using the proposed VHR framework, for the *Aerial* dataset (see Figure 3).

results using different number of clusters, which is achieved by varying the error threshold ζ_{max} (see Algorithm 1). If $\zeta_{max} = 0$ all clusters are considered, and as we increase this value more clusters are merged together to decrease the complexity of the resulting regression model. The corresponding degradation in terms of accuracy is shown in Figure 4b, where we can see that the merging technique proposed in 2.4.3 is able to detect and remove clusters with lower impact on the overall regression model (a decrease of 30% in the number of clusters produces an RMSE increase of only 3.88%). Naturally, at some point there are no more low impact clusters left, and further removals will start to have a more significant effect on overall model accuracy.

Table 1 presents quantitative results for different terrain modeling techniques when applied to both datasets, as data becomes sparser.[2] For 100%, all points are used for training and residual error calculation. For other levels, that percentage of points (determined randomly) is used solely for training purposes and the remaining ones only for residual error calculation (computational times for inference were calculated based on a $0.1m$ resolution grid). The RMSE results testify to each algorithm's ability to extrapolate over available data to infer the state of unobserved areas of the input space. As expected, standard interpolation techniques are unable to deal with sparser data, with the residual error rapidly increasing as more training points are removed. This increase is significantly smaller for the SGP and VHR frameworks, indicating better interpolative powers, however VHR consistently outperforms SGP in terms of residual error.

The computational times presented in Table 1 validate the VHR framework's ability to scale to larger datasets, especially when compared to the non-parametric

[2] All computations were performed on a i7/2.60 x 8 GHz notebook, with multi-threading enabled wherever possible. Due to lack of memory, training data was downsampled by 5 in the *Rover* dataset and by 25 in the *Aerial* dataset for tests using the SGP framework.

Table 1: Quantitative results for different terrain modeling techniques, with varying levels of data sparsity.

Metric	Sparsity	*Rover*				*Aerial*			
		LI	**CI**	**SGP**	**VHR**	**LI**	**CI**	**SGP**	**VHR**
RMSE (m)	100%	0.201	0.211	0.186	0.149	0.101	0.113	0.087	0.054
	90%	0.211	0.217	0.194	0.152	0.122	0.144	0.095	0.057
	70%	0.234	0.263	0.217	0.162	0.165	0.198	0.104	0.059
	50%	0.298	0.324	0.239	0.171	0.203	0.249	0.115	0.065
Training Time (s)	100%	—	—	4705.2	0.975	—	—	31936.3	9.266
	90%	—	—	3573.8	0.825	—	—	23216.7	7.172
	70%	—	—	1985.1	0.760	—	—	16759.1	5.500
	50%	—	—	1001.7	0.495	—	—	10845.8	3.938
Inference Time (s)	100%	0.602	0.643	46.340	1.587	2.748	3.382	807.432	8.694
	90%	0.361	0.490	42.945	1.423	2.470	2.865	498.374	8.019
	70%	0.303	0.368	37.296	1.234	1.578	1.978	259.128	7.273
	50%	0.217	0.276	28.254	1.092	0.934	1.107	168.109	6.558

SGP framework. As expected, interpolation techniques are much faster, since no training is required, only linear and cubic calculations based on nearest neighbors. The SGP framework, even when using a sparse approximation, required substantially more training time (by several orders of magnitude) to generate optimized model parameters, and inference was also noticeably slower. We attribute this increase in performance provided by the VHR framework to the decoupling between different local regression models, which produces efficient parameter updates during variational training. In [35] a similar local approximation using *kd*-trees is proposed as a way to maintain computational complexity low and bounded, which would further decrease SGP computational times, however this was not explored in this paper. Regardless, RMSE results for the VHR framework are consistently smaller, which makes it a more attractive terrain modeling technique both in terms of accuracy and computational power. Interestingly, for the VHR framework, as the number of training points decreases inference times become longer when compared to training times, due to the large number of query points necessary to produce a reconstructed model of the surveyed area, given the requested resolution.

4 Conclusion

This paper introduces a novel regression methodology based on sparse projections to a Reproducing Kernel Hilbert Space. Observed data is clustered and different local regression models are attributed to each cluster, thus allowing non-stationary behaviors throughout the input space and decreasing overall computational complexity. We employ a Variational Bayesian approach to uncertainty estimation, and training is conducted via the maximization of a single global variational bound that is dependent on the parameters for each local regression model. Furthermore, we show how kernel and model parameters can be jointly optimized, to produce a more accurate approximation to the underlying function or to further decrease computational costs by removing unnecessary clusters. The proposed VHR framework is tested in two different large-scale terrain modeling datasets, with results that surpass other commonly used terrain modeling techniques. Future work will focus on the use of different regression models and kernels, such as the space-carving kernel proposed in [12], and the active selection of regression models given local point distribution.

Acknowledgements

This research was supported by funding from the Faculty of Engineering & Information Technologies, The University of Sydney, under the Faculty Research Cluster Program.

References

1. D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1027–1035, 2007.
2. C. Bishop. *Pattern Recognition and Machine Learning (Information and Statistics*. Springer-Verlag New York, Inc., 2006.
3. T. Broderick, N. Boyd, A. Wibisono, A. Wilson, and M. Jordan. Streaming variational bayes. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1727–1735, 2013.
4. B. Carlin and T. Louis. *Bayesian Methods for Data Analysis*. Chapman and Hall, 3 edition, 2008.
5. K. Doherty, J. Wang, and B. Englot. Probabilistic map fusion for fast, incremental occupancy mapping with 3d Hilbert maps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016.
6. J. Drugowitsch. Variational bayesian inference for linear and logistic regression. Technical report, arXiv:1310.5438, 2013.
7. P. Frankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart. Robot-centric elevation mapping with uncertainty estimates. In *Proceedings of the International Conference on Climbing and Walking Robots (CLAWAR)*, 2014.
8. W. Gilks, S. Richardson, and D. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, 1 edition, 1995.
9. J. Gu, Q. Cao, and Y. Huang. Rapid traversability assessment in 2.5d grid-based map on rough terrain. *International Journal of Advanced Robotic Systems*, 5(4), 2008.
10. V. Guizilini and F. Ramos. Large-scale 3d scene reconstruction with Hilbert maps. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2016.
11. V. Guizilini and F. Ramos. Unsupervised feature learning for 3d scene reconstruction with occupancy maps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

12. R. Hadsell, J. Bagnell, D. Huber, and M. Hebert. Space-carving kernels for accurate rough terrain estimation. *International Journal of Robotics Research (IJRR)*, 29(8):981–996, 2010.
13. A. Honkela and H. Valpola. On-line variational bayesian learning. In *International Symposium on Independent Component Analysis and Blind Signal Separation (ICA)*, 2003.
14. D. Kidner, M. Dorey, and D. Smith. What's the point? interpolation and extrapolation with a regular grid dem. In *Proceedings of the International Conference on Geocomputation*, 1999.
15. E. Krotkov and R. Hoffman. Terrain mapping for a walking planetary rover. *IEEE Transactions on Robotics and Automation*, 10(6):278–292, 1994.
16. S. Lacroix, A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury, M. Herrb, and R. Chatila. Autonomous rover navigation on unknown terrains: Functions and integration. *International Journal of Robotics Research (IJRR)*, 21(10):917–942, 2002.
17. S. Lloyd. Least-squares quantization in pcm. In *IEEE Transactions on Information Theory*, volume 28, pages 129–136, 1982.
18. G. Matheron. Principles of geostatistics. *Economic Geology*, 58:1246–1266, 1963.
19. A. Melkumyan and F. Ramos. A sparse covariance function for exact Gaussian process inference in large datasets. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1936–1942, 2009.
20. M. Muja and D. Lowe. Fast approximate nearest neighbours with automatic algorithm configuration. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, volume 4, pages 331–340, 2009.
21. K. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 1 edition, 2012.
22. C. Plagemann, K. Kersting, and W. Burgard. Nonstationary Gaussian process regression using point estimates of local smoothness. In *Proceedings of the European Conference on Machine Learning (ECML)*, 2008.
23. C. Plagemann, S. Mischke, S. Prentice, K. Kersting, N. Roy, and W. Burgard. Learning predictive terrain models for legged robot locomotion. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2008.
24. P. Pba. Formulas for robust, one-pass parallel computation of covariances and arbitrary-order statistical moments. Technical report, Sandia National Laboratories, 2008.
25. J. Quinonero-Candelas and C. Rasmussen. An unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research (JMLR)*, 6:1939–1959, 2005.
26. F. Ramos and L. Ott. Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. In *Proceedings of Robotics: Science and Systems (RSS)*, 2015.
27. C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.
28. I. Rekleitis, J. Bedwani, D. Gingras, and E. Dupuis. Experimental results for over-the-horizon planetary exploration using a lidar sensor. 2008.
29. M. Sato. Online model selection based on the variational bayes. *Neural Computation*, 13(7):1649–1681, 2001.
30. B. Schölkopf, K. Muandet, K. Fukumizu, S. Harmeling, and J. Peters. Computing functions of random variables via reproducing kernel Hilbert space representations. *Statistics and Computing*, 25(4):755–766, 2015.
31. B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2001.
32. D. Sculley. Web-scale k-means clustering. In *Proceedings of the International Conference on World Wide Web (WWW)*, volume 19, pages 1177–1178, 2010.
33. E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1257–1264, 2006.
34. R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2006.
35. S. Vasudevan, F. Ramos, E. Nettleton, and H. Durrant-Whyte. Gaussian process modeling of large-scale terrain. *Journal of Field Robotics (JFR)*, 26(10):812–840, 2010.
36. M. Yuan and T. Cai. A reproducing kernel Hilbert space approach to functional linear regression. *The Annals of Statistics*, 38(6):3412–3444, 2010.