Probabilistic Machine Learning and 3D Mapping for Soil Variability and Crop Yield Predictions

Sebastian Haan

1 Summary

This software transforms sparse soil measurements and surface measurements into 3D cube prediction of soil properties and their uncertainty. The cubing is performed via Gaussian Processes (GP) with custom 3D spatial kernel to take into account spatial correlations and multiple mean function options that depend on additional covariates (e.g. terrain, vegetation, top soil properties, depth).

2 METHOD AND DESCRIPTION OF PROBABILISTIC FRAMEWORK

Here we propose a probabilistic framework that combines Gaussian Process (GP) model with complex multi-variate mean functions and sparse covariate 3D radial basis kernel to generate 3D cubes of soil properties and their uncertainties. Some of the main advantages and features of this method are:

- generation of predictive cubes (including uncertainty cubes and covariances) at any scale or resolution.
- options for complex mean function models and their uncertainty propagation to GP spatial covariance function.
- considering heteroscedatic input uncertainties (independent uncertainties for each location or measurement) uncertainties (can be location
- considering uncertainties in measurement positions (e.g. depth intervals)
- options to spatially integrate predictions and uncertainties over custom block sizes or area shapes (defined in polygon shapefiles)

2.1 Gaussian Process Model

GPs are a flexible, probabilistic approach using kernel functions with non-parametric priors, and are successfully used in a large range of machine learning problems (see Rasmussen and Williams 2006 for more details). Some of the important advantages of the GP method are that it generates a predictive distribution with a mean $\mu(\Phi)$ and variance $\sigma^2(\Phi)$ and that the GP marginal likelihood function is well defined by the values of their hyper-parameters, which allows it to optimise them exactly. This reasoning about functions under uncertainty and their well-tuned interpolation character allows GPs to work extremely well for sparse data as well as for big data (see (Melkumyan and Ramos 2009) for solving large covariance matrix).

A GP $f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k\left(\mathbf{x}, \mathbf{x}'\right)\right)$ is completely determined by its mean function $m(\mathbf{x}) = \mathrm{E}[f(\mathbf{x})]$

and covariance $k(\mathbf{x}, \mathbf{x}') = \mathrm{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$, essentially placing a multivariate Gaussian distribution over the space of functions that map the input to the output, or informally, to measure the similarity between points as function of, e.g., their distance (the kernel function) and to predict a Gaussian distribution over $f(x^*)$ (i.e., a mean value and variance) at any new sampling location (unseen point) x^* from training data.

2.1.1 Sparse 3D Covariance Function

The choice for an appropriate covariance function is important, as the GP's output directly depends on it. There are many stationary (invariant to translation in input space) and non-stationary covariance functions (for a review see Rasmussen and Williams 2006); the former are typically defined by radial basis functions (RBF) such as the squared-exponential covariance function,

$$k(x, x\prime) = \sigma_0^2 \exp\left(-\frac{1}{2l^2}(x - x')^2\right)$$

which depends on the radial distance between points x and x' and at least two parameters, one defining the length-scale l (multiple dimensions can have multiple length-scales) and the other the signal variance σ_0^2. These parameters of the covariance function are referred to as the hyperparameters Θ of the GP, which can be either given by a fixed covariance scale and noise, or learned from data by optimising the marginal likelihood. The latter allows us also to marginalise over a range of GP hyperparameters such as the length-scale parameters l. Computationally this can be done by defining a grid of hyperparameter values and then weight the posterior distribution with the corresponding marginalised likelihood values. In general, it is advisable to compute the joint inversion at least for a range of l parameters and to compare the reconstructed models against each other.

To handle the computational problem of inverting a large covariance matrix, (Melkumyan and Ramos 2009) proposed an intrinsically sparse covariance function

$$\begin{cases} k(r, l, \sigma_0) = \sigma_0\Big[\frac{1}{3}(2 + \cos(2\pi\frac{r}{l}))(1 - \frac{r}{l}) + \frac{1}{2\pi}\sin(2\pi\frac{r}{l})\Big], & \text{if } r < l. \\ 0, & \text{if } r \geq l. \end{cases}$$

where $\sigma_0 > 0$ is a constant coefficient, $l > 0$ is a given scale and r is the distance between the points. This covariance function can be extended to multiple dimensions (MD) in the following two ways: 1) using direct products for all axes:

$$K_{MD}(\mathbf{x}, \mathbf{x}', \mathbf{l}, \sigma_0) = \sigma_0 \prod_{i=1}^{D} K(x_i, x_i', l_i, 1)$$

, where D is the number of dimensions (here D=3) and $\mathbf{l}$ is the vector of the characteristic lengths. 2) using Mahalanobis distance:

$$\begin{cases} k(r, \sigma_0, \Omega) = \sigma_0\Big[\frac{1}{3}(2 + \cos(2\pi r))(1 - r) + \frac{1}{2\pi}\sin(2\pi r)\Big], & \text{if } r < 1. \\ 0, & \text{if } r \geq 1, \end{cases}$$

where

$$r = \sqrt{(\mathbf{x} - \mathbf{x}')^T \Omega (\mathbf{x} - \mathbf{x}')}$$

and with Ω expressed via diagonal matrix of characteristic lengths:

$$\Omega = diag(\frac{1}{l_1}, \frac{1}{l_2}, ..., \frac{1}{l_D})$$

it follows:

$$r = \sqrt{\sum_{k=1}^{D} \left(\frac{x_k - x'_k}{l_k} \right)^2}$$

.

2.1.2 Including Measurement and Positional Uncertainties

To take into account that positions of the measurements are not point coordinates but have an uncertainty (here along the vertical direction), dX, we need to modify the covariance function by assuming for the vertical coordinate $X_z = N(\mu(X_z)|dX_z)$, and it follows that

$$K(r, l, \sigma_0) = \frac{\sigma_0}{|1 + l^{-1}(dX_i + dX_j)|^{1/2}} \Big[\frac{1}{3}(2 + \cos(2\pi \frac{r}{l + dX_i + dX_j}))(1 - \frac{r}{l + dX_i + dX_j}) +$$

$$\frac{1}{2\pi} \sin(2\pi \frac{r}{l + dX_i + dX_j}) \Big],$$

if $r < l$, and $K(r, l, \sigma_0) = 0$ if $r >= l$.

In addition we have to include measurement uncertainties as a noise term (or regularisation parameter) added to the covariance function:

$$\mathbf{K} = \mathbf{k} + {\sigma_\mathbf{s}}^2 \mathbf{I}$$

where the uncertainties σ of the sensor observations $\mathbf{y}$ are included and can be either a constant σ (same uncertainty for all sensors) or a vector (σ_s) that is given by the individual uncertainty of each sensor observation. In case the sensor uncertainty is unknown, it can be estimated in conjunction with the other GP hyperparameters by maximizing the marginal log-likelihood (see next section).

2.1.3 Hyperparameter Optimisation

One advantage of the Bayesian framework is the ability to calculate the marginal log-likelihood,

$$\mathcal{L}(\Theta) = -\frac{1}{2} \mathbf{y}^T \mathbf{K}_{x,x}^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K}_{x,x}| - \frac{\log 2\pi}{2} \sum_{i=1}^{M} N_i$$

to optimize the parameters Θ of the GPs, which include the hyper-parameters (amplitude, kernel length-scales) of the covariance functions $K(x, x)$, and optionally the noise variances σ_s^2 if not given as input data.

The hyperparamter optimisation algorithm is based on scipy implementation of simplicial homology global optimization (SHGO), which is appropriate for solving general optimization problems to global optimality.

2.1.4 Predictive Distributions and Uncertainties

The predictive distribution y' at new locations x' is given by

$$y'|\mathbf{G}, \mathbf{y} \sim \mathcal{N}(\phi|\mu_{y'|y}, \mathbf{\Sigma}_{\phi|y})$$

with the predictive mean at each location

$$\mu = \mathbf{K}_{x,x'}^T \mathbf{K}_{x,x}^{-1} \mathbf{y}$$

and the predicted covariance

$$\mathbf{\Sigma} = \mathbf{K}_{x',x'} + \sigma_s^2 I - \mathbf{K}_{x,x'}^T \mathbf{K}_{x,x}^{-1} \mathbf{K}_{x,x'}.$$

The predicted variance is given by the diagonal elements of the covariance matrix.

2.2 Mean Functions

If the mean function is not zero but given by a predictive deterministic function $m_\phi = f(\mathbf{x_c})$, the predicted GP posterior distribution is given by

$$y'|\mathbf{G}, \mathbf{y} \sim \mathcal{N}(m_{\phi'} + \mathbf{K}_{x,x'}^T \mathbf{K}_{x,x}^{-1} \mathbf{y} - \mathbf{m}_\phi, \mathbf{K}_{x',x'} + \sigma_s^2 I - \mathbf{K}_{x,x'}^T \mathbf{K}_{x,x}^{-1} \mathbf{K}_{x,x'})$$

The covariates x_c are mostly surface measurements such as DEM, terrain slope, NDVI, radioactivity and the depth z. The uncertainty of the mean model function is used to construct the diagonal noise matrix of the GP $\sigma_\mathbf{s}$ and is proportional up to a constant scaling (determined by the hyperparameter optimisation).

2.2.1 Depth Model

The depth model is estimated via Bayesian Linear Regression and is given either by a linear relationship with depth z

$$m_\phi = f(z) = p_0 + p_1 z$$

or as power-law relation in log-log space

$$\ln[m_\phi] = p_0 + p_1 \ln[z]$$

2.2.2 Bayesian Linear Regression

First, we standardise the data and apply a featurewise power transform scaler via scikit-learn implementation (default option). Power transforms are a family of parametric, monotonic transformations that are applied to make data more like normal distributed. This is useful for modeling issues related to heteroscedasticity (non-constant variance), or other situations where normality is desired. In detail, the Yeo-Johnson transform (Yeo and Johnson 2000) is applied, which supports both positive or negative data. Note that the Bayesian Regression implementation in MLsoil is not limited only to power transform scaler but includes multiple options of scalers, such as StandardScaler or RobustScaler.

After the featurewise scaling of data, a Bayesian Ridge regression is performed using the scikit-learn implementation 'BayesianRidge'. This particular Bayesian ridge implementation is based on the algorithm described in Appendix A of Tipping (2001) where updates of the regularization parameters are done (MacKay 1992). To make the regression more robust, we add an automated feature selection after the initial regression by selecting only features with a ratio of correlation coefficient to standard deviation larger than one. Then a second BayesianRidge regression is made using only the selected features, and the model and coefficients are stored with non-significant coefficients set to zero. Predictions and their uncertainties are then obtained by using the trained model and are scaled back with the powerlaw transform. For more implementation details see `blr.py`.

2.2.3 Probabilistic Neural Network

The probabilistic neural network is implemented by building a custom tensorflow probability model with automatic feature selection for sparsity. For implementation details see `bnn.py`. This method requires a feature-wise standard scaler.

2.2.4 Random Forest

The scikit-learn Random Forest model implementation is applied. No data scaler required. Prediction uncertainties are currently estimates by using the standard deviation and Confidence Intervals of all decision trees. For more implementation details and hyper parameter settings see `rf.py`

2.3 Model Accuracy and Residual Analysis

The residual analysis is performed on the validation data with a split with 10 cross validations. The residual error RE$ = Y_{pred} - Y_{ground}$ is defined as the predicted value minus the ground truth for each data point in the validation set. The normalised Root-Mean-Square-Error (nRMSE) is computed as RMSE divided by the standard deviation of the validation data. The lower the nRMSE value, the better is the prediction. If nRSME turns out equal or larger than 1, the model does not improve the prediction in comparison to the mean function of the GP. To test whether the predicted uncertainties are consistent with the residual error of the prediction, we calculate the ratio

$$\theta = \frac{RE^2}{\sigma^2_{Ypred}}$$

3 Feature Importance

Feature importance can be used for inference as well as feature selection, which is based on their importance ranking or their respective statistical significance. However, feature importance is dependent on which model is applied and can show significant difference for different models, in particular if non-linear or feature interaction effects are taken into account by the model (or not). MLsoil provides a range of feature importance calculation and creates automatically multiple feature importance rankings based on different models as outlined in the following.

3.1 Bayesian Linear Regression

Features are ranked based on their statistical significance which is given for each feature by the ratio of the amplitude to standard deviation of the regression correlation coefficient. The correlation coefficients can be estimated for both, the linear Bayesian Ridge Regression and power-law scaled Bayesian Ridge Regression.

3.2 Random Forest with permutation based feature importance

Feature importances based on impurity can be misleading for high cardinality features (many unique values). An alternative is feature importance based on feature permutation which has no bias towards high-cardinality features and can be computed on a test set. Basically all features are shuffled n-times and the model is refitted to estimate the importances. The permutation feature importance is then defined to be the decrease in a model score when a single feature value is randomly shuffled. Note that this technique is model agnostic. The draw-back is that this permutation method is computationally more costly than others.

References

MacKay, David JC. 1992. "Bayesian Interpolation." *Neural Computation* 4 (3): 415–47.

Melkumyan, Arman, and Fabio Ramos. 2009. "A Sparse Covariance Function for Exact Gaussian Process Inference in Large Datasets." In *IJCAI*, 9:1936–42.

Rasmussen, Carl Edward, and Christopher KI Williams. 2006. *Gaussian Process for Machine Learning*. MIT press.

Tipping, Michael E. 2001. "Sparse Bayesian Learning and the Relevance Vector Machine." *Journal of Machine Learning Research* 1 (Jun): 211–44.

Yeo, In-Kwon, and Richard A Johnson. 2000. "A New Family of Power Transformations to Improve Normality or Symmetry." *Biometrika* 87 (4): 954–59.