GeoBO: A Python package for Multi-Objective Bayesian Optimisation and Joint Inversion in Geosciences

`GeoBO` is build upon a probabilistic framework using Gaussian Process (GP) priors to jointly solve multi-linear forward models. This software generates multi-output 3D cubes of geophysical properties (e.g. density, magnetic susceptibility, mineral concentrations) and their uncertainties from 2D survey data (e.g. magnetics and gravity) and any pre-existing drillcore measurements. The reconstructed 3D model is then used to query the next most promising measurement location given an expensive cost function (e.g. for drillcores). A ranked list of new measurements is proposed based on user-defined objectives as defined in the acquisition function which typically aims to optimize exploration (reducing global model uncertainty) and exploitation (focusing on highly promising regions) while minimizing costs.

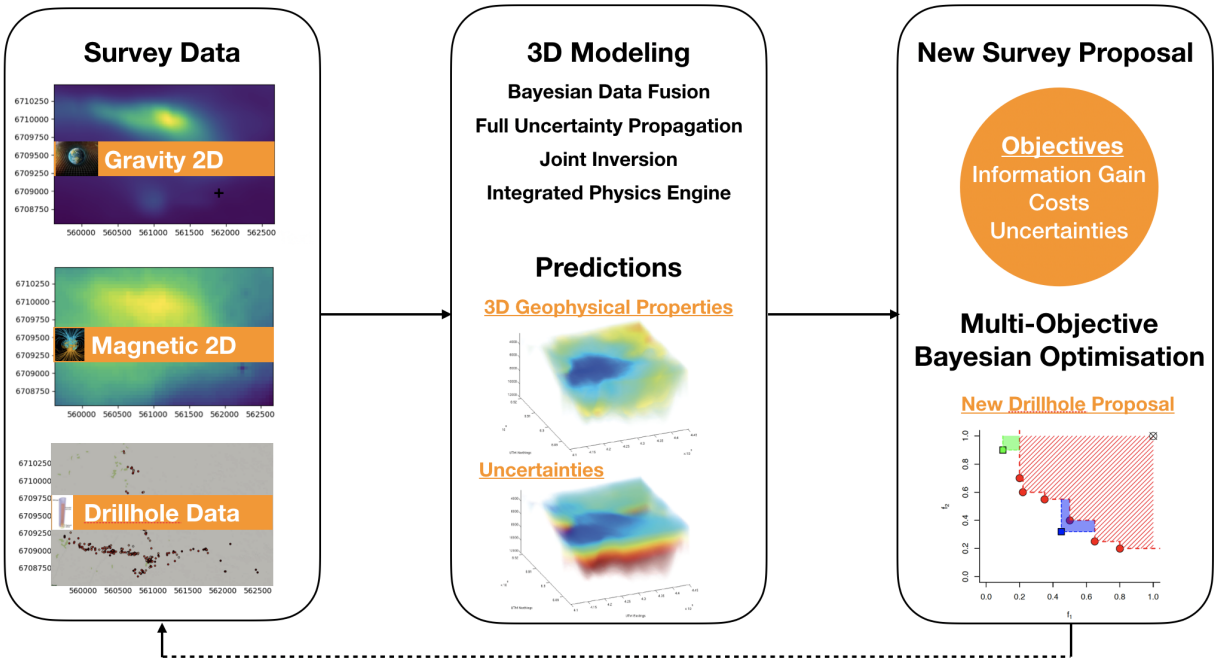


Figure 1: GeoBO Framework

Table of Contents

- Definitions
- Functionality
- Installation And Requirements
  - Requirements
  - Installation
  - Documentation
- Usage and Settings
- Examples and Tests
  - Synthetic Models
  - Drillcore Test Example
  - Results and Output Files
- Options and Customization
  - Custom Linear Forward Models
  - Gaussian Process Kernel Functions
- Literature

- Related Software
- Attribution and Acknowledgments
  - Project Contributors
- License

Definitions

Bayesian Optimisation (BO) is a powerful framework for finding the extrema of objective functions that are noisy, expensive to evaluate, do not have a closed-form (e.g. black-boxfunctions), or have no accessible derivatives. The model used for approximating the objective function is called surrogate model, which is typically based on a Gaussian Process models for tractability. Gaussian Processes define a prior over functions (typically given by a kernel function) and is used to propose points in the search space where sampling is likely to yield an improvement. The specific set of objectives for the improvement are defined in an acquisition function, which guides the search for a user-defined optimum. An example use case scenario is described in a nutshell in OPTIMIZATION_FOR_ACTIVE_SENSORFUSION_IN_A_NUTSHELL.pdf.

Acquisition function

The key of BO is the acquisition function, which typically has to balance between:

a) exploration, i.e., querying points that maximise the information gain and minimize the uncertainty of a model
b) exploitation, i.e. querying points that maximise the reward (e.g. concentrating search in the vicinity locations with high value such as minerals)
c) minimize the number of samples given an expensive cost function for any new measurement.

Forward Models and Joint Inversion

In geology and geophysics, inversion problems occur whenever the goal is to reconstruct the geological conditions, i.e. the 3D distribution of physical rock properties, that give rise to a set of (2D) geophysical observations. Since the number of possible geological configurations is typically greater than the number of observational constraints, the problem is nearly always under-determined. Forward models transform the localized measurement of a remote sensor grid into a 3D representation of geophysical properties of a region. The most common geophysical linear forward model are gravity and magnetic forward models, which are computed using Li's tractable approximation. Joint inversion is simultaneously interpreting multiple (distinct) sensor measurements using a single model to provide a better constrained joint solution rather than taking individual solutions that only satisfy their aspect of data on their own.

Functionality

GeoBO's probabilistic framework includes all steps from prior selection, data fusion and inversion, to sensor optimisation and real world model output. The main functionalities of GeoBO are summarised in the following:

- Joint probabilistic inversion tool by solving simultaneously multi-linear forward models (e.g. gravity, magnetics) using cross-variances between geophysical properties (cross-variance terms can be specified by user).
- Output 1: Generation of cubes and computation of complete posterior distribution for all geophysical properties (described by their mean and variance value at each location (cubecell aka voxel).
- Output 2: Generation of ranked proposal list for new most promising drillcores based on global optimisation of acquisition function
- Templates for acquisition function to use in Bayesian Optimisation
- Flexible parameter settings for exploration-exploitation trade-off and inclusion of local 3D cost function in acquisition function

Other features are:

- Generation of simulated geophysical data with a choice of three different models
- Package includes geological survey/drillcore sample as well as synthetic data and functions for synthetic data generation
- Generation of 2D/3D visualisation plots of reconstructed cubes and survey data
- 3D Cube export in VTK format (for subsequent analysis, e.g., in Python or with ParaView)
- Options to include any pre-existing drillcore data
- Included linear forward models: density-to-gravity and magnetic susceptibility-to-magnetic field; custom linear forward models can be added (see Custom Linear Forward Models)
- Library of Gaussian Process (GP) kernels including sparse GP kernels
- Flexible settings for any cube geometry and resolution
- (Optional) Optimization of GP hyperparameters and cross-correlation coefficients via computation of marginal GP likelihood

Example outputs can be found in the directory `examples/results/`.

Installation And Requirements

Installation

To install GeoBO locally using setuptools:

```
python setup.py build
python setup.py install
```

or using pip:

```
pip3 install geobo
```

The installation can be tested by running the example with included synthetic data and default settings:

```
cd geobo/
python main.py tests/settings_example1.yaml
```

Requirements

- python >=3.6
- numpy
- matplotlib
- scikit_image
- scipy
- rasterio
- pandas
- pyvista
- skimage
- PyYAML

Documentation

Documentation conversion is generated using pandoc. The README markdown file can be converted to PDF:

```
pandoc -V geometry:margin=1.0in README.md -o README.pdf
```

A complete API documentation for all modules can be found here:

- `run_geobo.py`
- `inversion.py`
- `kernels.py`
- `cubeshow.py`
- `sensormodel.py`

- `simcube.py`
- `utils.py`

Usage and Settings

1) Change the main settings such as filenames and parameters in `settings.yaml`. These settings specify:

- directory, filenames, and geophysical drillcore properties
- the generated cube's geometry, size, and resolution
- Gaussian Process settings (lengthscale, input data uncertainty, correlation coefficients, kernel function)
- local Earth's magnetic field vector
- Bayesian Optimisation Settings (vertical/non-vertical drillcores, the exploration/exploitation and cost weighting)
- plotting settings
- optional generation of simulated data

2) Then run geobo

```
cd geobo/
python main.py settings.yaml
```

The main functions for the acquisition function can be found in `run_geobo.py`; visualisation functions and VTK export are defined in `cubeshow.py`; inversion functions are defined in `inversion.py`.

Examples and Tests

Synthetic Models

Synthetic geophysical models can be created by setting switching on `gen_simulation` in the settings yaml file. Three different models are so far implemented:

- two-cylindric dipping bodies (`modelname: 'cylinders'`)
- two layer model (`modelname: 'layers2'`)
- three layer model (`modelname: 'layers3'`) For each model a 3D voxel cube with geological structures is generated with density and magnetic susceptibility properties, plus the corresponding 2D gravity and magnetic remote sensor measurements. Other custom models can be included by adding a new model in function `create_syncube()` in `simcube.py`

Result examples of the synthetic models are stored in the subfolder `examples/testdata/synthetic/`. An example settings file is given in `settings_example1.yaml` and can be run by

```
cd geobo/
python main.py tests/settings_example1.yaml
```

Drillcore Test Example

Another examples includes drillcore and gravity/magnetic survey data (`examples/testdata/sample/`). This example can be run with

```
cd geobo/
python main.py tests/settings_example2.yaml
```

and creates the reconstructed density and magnetic susceptibility cubes, uncertainty cubes

Results and Output Files

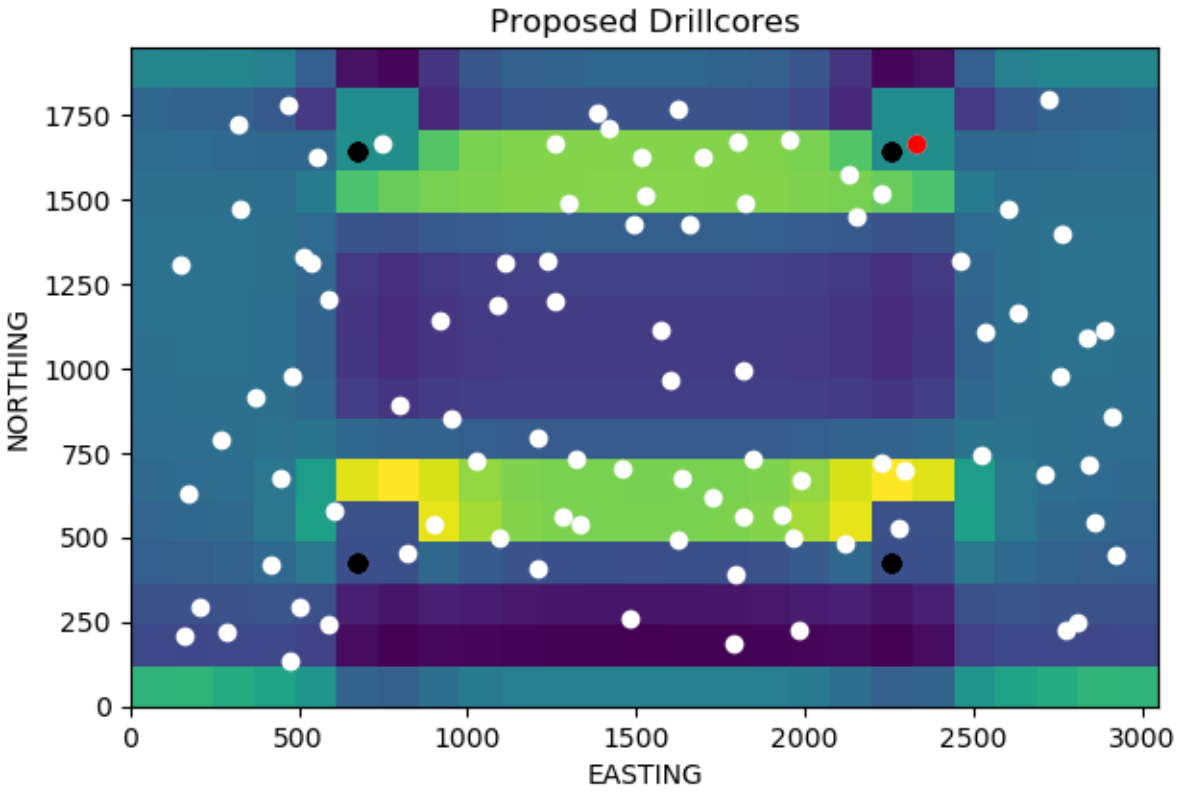
The output results include the generated reconstructed density and magnetic susceptibility cubes and their corresponding uncertainty cubes, visualisations of original survey data and reconstructed properties, and list of new measurement proposals.

List of Joint Inversion Result Outputs:

- 3D Cube files in vtk format (to use, e.g., with PyVista or ParaView): Output of cross-correlated reconstructed properties (density: **cube_density.vtk**, magnetic susceptibility: **cube_magsus.vtk**, property from drill: **cube_drill.vtk**) and their uncertainity cubes in terms of standard deviation (**cube_density_variance.vtk, cube_drill_variance.vtk, cube_drill_variance.vtk**). In case the cross-covariance terms (`gp_coeff` in the settings file) are all set to zero, the solutions for each forward model are independent from each other.
- Optional (Default optiion: plot=True in function `read_surveydata()`): Images of the input gravitational field (**gravfield.png**) and magnetic field (**magfield.png**) and their corresponding downsampled images (**gravfield_downsampled.png, magfield_downsampled.png**)
- Optional (if `plot3d`:True in settings): 3D Preview of reconstructed properties: density (**density-mesh3D.png**), magnetic suscpetibility (**magsus-mesh3D.png**), and drill property (**drill-mesh3D.png**). However, it is recommended to use PyVista or Paraview instead. Future updates for 3D visualisations will be based on PyVista.
- Optional (if `plot_vertical`:True in settings): 2D maps of vertically (along z-axis) mean value of cube properties (**dens_rec2D_loc2.png, magsus_rec2D_loc2.png, drill_rec2D_loc2.png**)

List of Bayesian Optimisation Output:

- List of all new measurement proposals (here for drillcores) ranked from maximum (hightest gain) to minimum of optimisation function. The results are saved as csv file (**newdrill_proposals_non-vertical.csv** or **newdrill_proposals_vertical.csv**) and include for vertical drillcores the position (Easting, Northing) and for non-vertical drillcores position and drill angles (Azimuth, Dip).
- Points of proposed measurement positions on top of reconstructed drill property image (mean projection along z-axis): The output figure (non-vertical drillcores:**newdrill_proposals.png** or vertical drillcores:**newdrill_vertical_proposals.png**) shows the location of the already existing drills as given by input measurements (black points), the new proposed drill positions (white), and the best (maximum of optimsation function) new drill location (red).



Options and Customization

Custom Linear Forward Models

The relationship between a physical system (or its model parameters) $\boldsymbol{P}$ and the observed sensor data $\mathbf{y}$ is described by a linear forward model

$\mathbf{y} = \mathbf{G}\ \boldsymbol{P}$,

where $\mathbf{G}$ is the transformation operator or matrix. The gravitational and magnetic forward model can be determined analytically by using Li's tractable approximation (see Li and Oldenburg 1998) for a 3D field of prisms of constant susceptibility and density, and GeoBO applies this prism shape model to compute the corresponding sensor sensitivity for gravity and anomalous magnetic field related to each prism cell.

The current implementation includes magnetic and gravity forward models, which are defined in the module `sensormodel.py` by the functions `A_sens()`,`grav_func()`, and `magn_func()`. The easiest way to add custom models is to create a new forward model function similar to the included functions `grav_func()` or `magn_func` and to compute the forward model matrix with `A_sens()`, if possible. The custom function need to describe the sensitivity or relationship for a particular point relative to the sensor origin (see, e.g., `grav_func()`).

In general any linear forward model can be added by changing accordingly the forward model matrix as computed by `A_sens()` as long as this function returns the matrix $\mathbf{G}$ that satisfies the linear relation $\mathbf{y} = \mathbf{G}\ \boldsymbol{P}$.

Gaussian Process Kernel Functions

Gaussian Processes (GPs) are a flexible, probabilistic approach using kernel machines and can propagate consistently uncertainties from input to output space under the Bayesian formalism. Another advantage of GPs is that their marginal likelihood function is well defined by the values of their hyper-parameters, and can thus be optimized. The choice for an appropriate covariance (kernel) function is important and there are many stationary (invariant to translation in input space) and non-stationary covariance functions available (for an overview, see, e.g., Rasmussen and Williams 2006). To handle the computational problem of inverting a large covariance matrix, GeoBO uses by default an intrinsically sparse covariance function (Melkumyan et, al. 2009). However, other standard kernel functions are available (see module `kernels.py`), which includes the squared exponential and Matern32 function and their their corresponding multi-kernel covariance functions (see Melkumyan et. al. 2011).

The settings yaml file allows you to choose the kernel function by configuring the parameter `kernelfunc`, which can be set either to 'sparse' (Default), 'exp' (squared exponential) or 'matern32'. New custom kernels can be a added in the module `kernels.py`, which requires to write their covariance function (see as example `gpkernel()`) and cross-covariance function (see as example `gpkernel_sparse()`), and then to add their function name to settings.yaml and to `create_cov()` in `kernels.py`.

The hyperparameters of the GP kernel can be configured in the settings yaml file (see Gaussian Process Settings) and are given by the lengthscale (`gp_lengthscale`), the noise regularization terms (`gp_err`) per forward model, and the cross-covariance amplitude terms which (`w1,w2,w3`) that coorrepond to the correlation coefficients between the model properties (e.g., rock density ,magnetic susceptibility, and drillcore measurements). The mathematical details for construction of the Multi-Kernel Covariance Functions are described in Haan et al 2020.

Bayesian Optimisation Options

To find the optimal new sampling point, GeoBO maximises the objective function (acquisition function) which is defined by the Upper Confidence Bound (UCB)

UCB(x) = m(x) + k *sigma*(x) - b c(x)

with the mean value for the prediction m(x), the variance *sigma*2(x), and a cost function c(x), which is defined by the cost of obtaining a measurement at the sample point x. The parameter k and b define the trade-off in exploration-to-exploitation and gain-to-cost, respectively. For example, maximizing the mean value can be beneficial if the goal is to sample new data at locations with high density or mineral content, and not only where the uncertainty is high. The parameters k and b can be accordingly specified by the user in the settings yaml file. Moreover, the settings allow the user to choose between vertical and non-vertical drillcore; in the latter case GeoBO is optimising also dip and azimuthal angle of the drillcore in addition to drillcore position.

Literature

Sebastian Haan, Fabio Ramos, Dietmar Muller, "Multi-Objective Bayesian Optimisation and Joint Inversion for Active Sensor Fusion", Geophysics, 86(1), pp.1-78. arXiv Preprint

Carl Edward Rasmussen and Christopher KI Williams, Gaussian process for machine learning, MIT press, 2006.

Li Yaoguo and Douglas W Oldenburg, "3d-inversion of gravity data," Geophysics, vol. 63, no. 1, pp. 109–119, 1998.

Arman Melkumyan and Fabio Ramos, "A sparse covariance function for exact gaussian process inference in large datasets.," in IJCAI, 2009, vol. 9, pp. 1936–1942

Armon Melkuyman and Fabio Ramos, "Multi-kernel gaussian processes," in IJCAI, 2011, vol. 22, p. 1408

Reid, A., O. Simon Timothy, E. V. Bonilla, L. McCalman, T. Rawling, and F. Ramos, 2013, Bayesian joint inversions for the exploration of earth resources.: IJCAI, 2877

Eric Brochu, Vlad M Cora, and Nando De Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," arXiv preprint arXiv:1012.2599, 2010.

Related Software

For the inversion part, GeoBO uses a direct inversion method via transformation of Gaussian Process priors, which enables joint inversion but is limited to linear forward models (e.g. gravity, magnetics, drillcores). For solving more complex non-linear forward models (e.g., seismic, or prior geological knowledge), the following bayesian inversion methods can potentially be applied to generate 3D geophysical surrogate models or to further refine GeoBo's 3D posterior model:

- hIPPYlib: an Extensible Software Framework for Large-scale Deterministic and Bayesian Inverse Problems. Publication Link; the software code is available at hippylib.github.io
- Obsidian: a flexible software platform for MCMC sampling of 3-D multi-modal geophysical models on distributed computing clusters. Publication Link; the code for version 0.1.2 of Obsidian is available at https://doi.org/10.5281/zenodo.2580422
- GemPy: open-source stochastic geological modeling and inversion; geoscientific model development. See gempy.org

Attribution and Acknowledgments

Acknowledgments are an important way for us to demonstrate the value we bring to your research. Your research outcomes are vital for ongoing funding of the Sydney Informatics Hub.

If you make use of this code for your research project, please include the following acknowledgment:

"This research was supported by the Sydney Informatics Hub, a Core Research Facility of the University of Sydney."

Project Contributors

Key project contributors to the GeoBO project are:

- Dr. Sebastian Haan (USYD, Sydney Informatics Hub): Expert in machine learning and physics, main contributor and software development of GeoBO.
- Prof. Fabian Ramos (USYD): Computational scientist and research expert in machine learning and bayesian computational techniques.
- Prof. Dietmar Muller (USYD, School of Geoscience): Research expert in geophysics and geoscience applications.
- Dr. Ben Mather (USYD, Sydney Informatics Hub/School of Geoscience): Computational Geophysicist, GeoBO testing.

License

You should have received a copy of the GNU Affero General Public License along with this program (see LICENSE.md). If not, see https://www.gnu.org/licenses/.