Inverse Theory: Exercise B

Patrick Eriksson, Simon Pfreundschuh

December 18, 2017

This repository contains the second exercise for the PhD course in inverse theory. It can be retrieved from `github.com/simonpf/inverse_theory_exercise_b/` or from the command line:

```
git clone https://github.com/simonpf/inverse_theory_exercise_b
```

After cloning the repository (or downloading it) run the `get_data.sh` script to download the data for this exercise. This will create and populate the `data` subdirectory in your exercise folder.

```
chmod +x ./get_data.sh
./get_data.sh
```

1 Repository Structure

- `exercise_b.py`: The python exercise template. This should be a good starting point if you are planning to do the exercise in python.
- `exercise_b.m`: The MATLAB exercise template. This should be a good starting point if you are planning to do the exercise in MATLAB.
- `data/`: Subdirectory containing the data for the exercise.
  - `/data/python/`: Contains all data required for this exercise in the form of `numpy` arrays.
  - `/data/matlab/`: Contains the `exercise_b.mat` which holds all data required for this exercise.
  - `/data/plots/`: Additional data required by the provided plotting functions.

- `utils/`: Subdirectory containing additional code mainly for plotting of the results.

- `doc/`: This exercise description and related content.

2 Background and Summary

In this exercise you will retrieve the *integrated ice column density* or *ice water path* (IWP) from passive microwave observations from the *Global Preciptiation Measurement* (GPM) *Microwave Imager*. Eliasson et al. [1] argue that measuring the bulk mass of ice in the atmosphere constitutes an important gap in the current global climate observation system, which leads to large differences in the IWP estimates of climate models. Retrieving IWP from passive microwave sensors provides global coverage at a much higher frequency than can be achieved with active sensors. Compared to LIDAR observations, microwave observations also have the advantage of being able to penetrate through thick clouds.

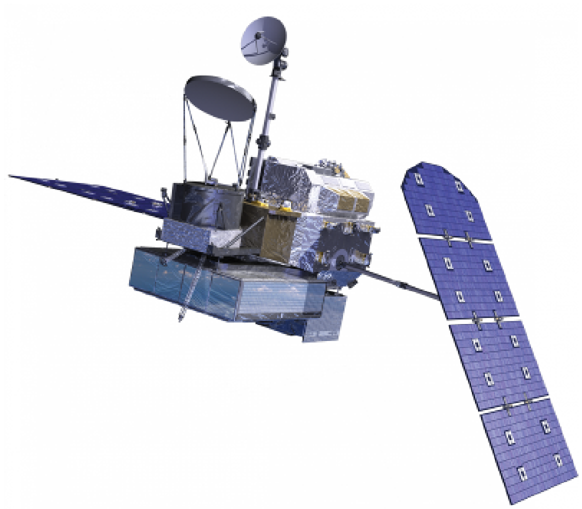

Figure 1: The Global Precipitation Measurement Microwave Imager onboard the GPM core observatory satellite.

[1] Simulations performed and kindly provided by Bengt Rydberg.

3 Methods

The measurement of cloud properties from passive microwave observations is based on the interaction of thermal microwave radiation with clouds through scattering. While possible, modeling scattering in a radiative transfer model is computationally too costly to be performed *during* the retrieval. In this exercise we are therefore considering two methods that use a *precomputed* database [2] consisting of pairs $\{(\mathbf{y}_i, x_i)\}_{i=1}^{n}$ of simulated brightness temperatures $\mathbf{y}_i$ and corresponding IWP values x_i. The ensemble of atmospheric states from which the database is computed, was generated from profiles of ice water content obtained from the DARDAR[3] dataset. This is to ensure that the ensemble follows a physically meaningful a priori distribution.

3.1 Bayesian Monte Carlo Integration

The basic idea of *Bayesian Monte Calo Integration* (BMCI) or simply *Monte Carlo Integration* as a retrieval method is to use importance sampling to transform samples from the a priori distribution to samples from the posterior distribution.

Consider the expected value $\mathcal{E}_{x|\mathbf{y}}(f(x))$ of a function f computed with respect to the a posteriori distribution $p(x|\mathbf{y})$:

$$\int f(x')p(x'|\mathbf{y})\,dx' \tag{1}$$

Using Bayes theorem, the integral can be computed as

$$\int f(x')p(x'|\mathbf{y})\,dx' = \int f(x')\frac{p(\mathbf{y}|x')p(x')}{\int p(\mathbf{y}|x'')\,dx''}\,dx' \tag{2}$$

To simplify notation, we introduce the weighting function $w(\mathbf{y}, x)$:

$$w(\mathbf{y}, x) = \frac{p(\mathbf{y}|x')}{\int p(\mathbf{y}|x'')\,dx''} \tag{3}$$

Note that the second integral in (2) is just the expectation value $\mathcal{E}_x\{f(x)w(\mathbf{y}, x)\}$ of the function f weighted with the weighting function $w(\mathbf{y}, x)$ but with

[2] Eliasson, S., S. A. Buehler, M. Milz, P. Eriksson and V. O. John Assessing observed and modelled spatial distributions of ice water path using satellite data

[3] `http://www.icare.univ-lille1.fr/projects/dardar`

respect to the a priori distribution. The integral in (1) can thus be approximated by a sum of $f(x_i)w(\mathbf{y}, x_i)$ over a simulation database, which is distributed according to the a priori distribution $p(x)$:

$$\int f(x')p(x'|\mathbf{y})\, dx' \approx \sum_{i=1}^{n} f(\mathbf{x}_i)w(\mathbf{y}, x_i) \tag{4}$$

3.1.1 The Weighting Function

Assuming that our forward model simulations $(\mathbf{y}_i, x_i)$ are exact up to a zero-mean, Gaussian error with covariance matrix $\mathbf{S}_e$, the weights are given by

$$w(\mathbf{y}, x_i) = \frac{1}{C} \cdot \exp\left\{ -\frac{(\mathbf{y} - \mathbf{y}_i)^T \mathbf{S}_e^{-1} (\mathbf{y} - \mathbf{y}_i)}{2} \right\} \tag{5}$$

for some normalization factor C. The normalization factor is found to be $C = \sum_{n=1}^{n} w(\mathbf{y}, x_i)$ to ensure that $\mathcal{E}_{x|\mathbf{y}}\{1\} = 1$.

3.1.2 The Retrieval

The above approach can be used to retrieve various statistics of the posterior distribution. The most basic are the mean and the variance:

$$\bar{x} = \mathcal{E}_{x|\mathbf{y}}\{x\} \approx \sum_{i=1}^{n} w(\mathbf{y}, x_i) x_i \tag{6}$$

$$\mathrm{var}(x) = \mathcal{E}_{x|\mathbf{y}}\{(x - \bar{x})^2\} \approx \sum_{i=1}^{n} w(\mathbf{y}, x_i)(x_i - \mathcal{E}_{x|\mathbf{y}}\{x\})^2 \tag{7}$$

But it is even possible to approximate the cumulative distribution function of the a posteriori distribution using:

$$F_{x|\mathbf{y}}(x') = \int_{-\infty}^{x'} p(x)\, dx \tag{8}$$

$$= \mathcal{E}_{x|\mathbf{y}}\{\mathbf{I}_{x<x'}\} \tag{9}$$

$$\approx \sum_{x_i < x'} w(\mathbf{y}, x_i) \tag{10}$$

3.2 Machine Learning

Given the database, the simplest way of setting up an IWP retrieval is probably by learning the inverse method $x = R(\mathbf{y})$ from the data. This can be done using machine learning methods. For regression, machine learning methods are trained in a *supervised manner*, that is by minimizing a given loss function over a training set.

The training set in this case will be the simulation database $\{(\mathbf{y}_i, x_i)\}_{i=1}^{n}$. This nomenclature is a bit unfortunate because in machine learning the input is usually denoted by $\mathbf{x}$ and the output to learn by y.

For regression the most commonly used loss function is the mean squared error loss. Statistically, this may be seen as training a maximum likelihood estimator of the mean of a conditional Gaussian distribution. While this perspective would even allow us to treat the retrieval problem in a Bayesian way, we will not pursue this statistical interpretation here.

3.2.1 Neural Network 101

Neural networks are a general computing model that compute a vector of output activations $\mathbf{y}$ from an input vector $\mathbf{x}$ by propagating the input activations through a sequence $i = 1, \ldots, n$ of layers with associated learnable weight matrices $\mathbf{W}_i$ and bias vectors $\mathbf{b}_i$:

$$\mathbf{x}_0 = \mathbf{x} \tag{11}$$

$$\mathbf{x}_i = f_i\left(\mathbf{W}_i \mathbf{x}_{i-1} + \mathbf{b}_i\right) \tag{12}$$

$$\mathbf{y} = \mathbf{x}_n \tag{13}$$

The f_i s here are the activation functions of each layer. Non-linear activation functions allow the network to learn complex, non-linear mappings from the input vector $\mathbf{x}$ to the output $\mathbf{y}$.

Neural networks are trained by finding the weight matrices $\mathbf{W}_1, \ldots \mathrm{W_n}$$ and bias vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ that minimize the mean of a given loss function $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ over a training set. All commonly used training algorithms are based on a technique called *backpropagation* to compute the gradients of the training loss with respect to the weights and biases of each layers. These gradients are then used to update the learnable parameters. For large datasets, stochastic (batch) gradient descent (SGD) is usually a good algorithm to start with. In each training step, SGD computes gradients on randomized subsets of the training set and uses them to update the weights.

While this exercise is clearly not the right place for a complete introduction to neural networks, modern machine learning packages usually only require you to provide training data and choose the loss function and training method, so this is hopefully enough to get you started.

3.2.2 Other Machine Learning Methods

Even though neural networks are a pretty hot topic right now, there are many other machine learning methods that might perform just as good especially on regression tasks and moderately sized data sets. Some examples that might be worth considering are:

- regression trees and forests
- boosted regression trees and boosting in general
- support vector machines

4 Exercises

The simulation database for this exercise consists of 350000 pairs $(\mathbf{y}_i, x_i)$ of simulated brightness temperatures $\mathbf{y}_i$ and corresponding ice water path values x_i. Each observation vector $\mathbf{y}_i$ consists of the brightness temperatures of channels $8, 9, 10, 11, 12, 13$ of the GMI radiometer. For this exercise we will assume that the only uncertainty in our simulation database is due to thermal noise in the receiver.

Channel	Center freq $[GHz]$	Polarization	NEDT (K)
8	89	V	0.32
9	89	H	0.31
10	166	V	0.7
11	166	H	0.65
12	183.31 ± 3	V	0.56
13	183.31 ± 7	V	0.47

4.1 Data

The data required to solve this exercise is provided both as `numpy` arrays and MATLAB arrays.

- `y_database`: 35000×6 array containing the simulated brightness temperatures for the retrieval databse.

- `iwp_database`: 35000×1 array containing the IWP values corresponding to the simulated brightness temperatures in `y_database`.
- `y_validation`: Additional simulated brightness temperatures to test the retrieval.
- `iwp_validation`: IWP values corresponding to the brightness temperatures in `y_validation`.
- `gmi_tbs_0, gmi_tbs_1`: Data from two different GMI orbits containing observations of the tropical storm Saola. The observations displayed below show orbit 0, but feel free to use orbit 1 as well.

4.2 BMCI

1. The Database

   Plot the distribution of ice water path values in the database. What is the range of IWP values? What is the reason for the bimodal character of the distribution?

2. Basic Implementation

   Write a function

   ```
   bmci(y_database, x_database, s_o, y)
   ```

   where

   - `y_database`: Matrix containing the simulated observations along its rows.
   - `x_database`: Vetor containing the corresponding IWP values (or any other quantity you may want to retrieve).
   - `s_o`: Matrix containing the covariance matrix $\mathbf{S}_o$ describing the observation uncertainty.
   - `y`: The observations for which to retrieve the ice water path. Given either as a vector (for a single inversion) or as a matrix with the observations along its rows.

   The method should return two vectors containing the expected values and standard deviations of the posterior distributions corresponding to the observations given in `y`.

3. Error Analysis

   Compute and plot the *mean absolute precentage error* (MAPE) and the *mean percentage error* (MPE):

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} \frac{|\bar{x}(\mathbf{y}_i) - x_i|}{\bar{x}(\mathbf{y}_i)} \tag{14}$$

$$MPE = \frac{100\%}{n} \sum_{i=1}^{n} \frac{\bar{x}(\mathbf{y}_i) - x_i}{\bar{x}(\mathbf{y}_i)} \tag{15}$$

   as a function of the retrieved mean of the posterior $\bar{x}(\mathbf{y}_i) > 0$ for the simulated measurements $\mathbf{y}_i, x_i$ contained in the arrays `y_validation` and `iwp_validation`.

   Compute and plot also the mean of the relative error estimated from the standard deviation of the posterior as a function of $\bar{x}(\mathbf{y}_i)$.

   What does this tell you about the retrieval?

4. Retrieving the Posterior CDF

   Write a function `bmci_cdf(y_database, x_database, s_o, y)`, that retrieves the cumulative distribution function of the posterior for a single observation `y`.

   The CDF for the 14325th (0-based indexing!) database entry should look like this:

   Given the CDF of the posterior what would be your *best estimate* if you had to return a single IWP value as the retrieval? How does this compare to the expected value for the validation data set?

5. Apply your Retrieval

   The file `data/tbs_gmi` contains the observerd calibrated brightness temperatures from the (extra-)tropical storm Saola as it tracked south-east of Japan 2017-10-27.

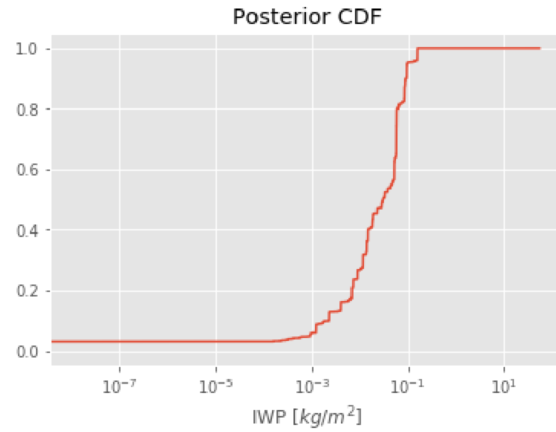

Figure 2: Posterior CDF for entry 14325 in the database.

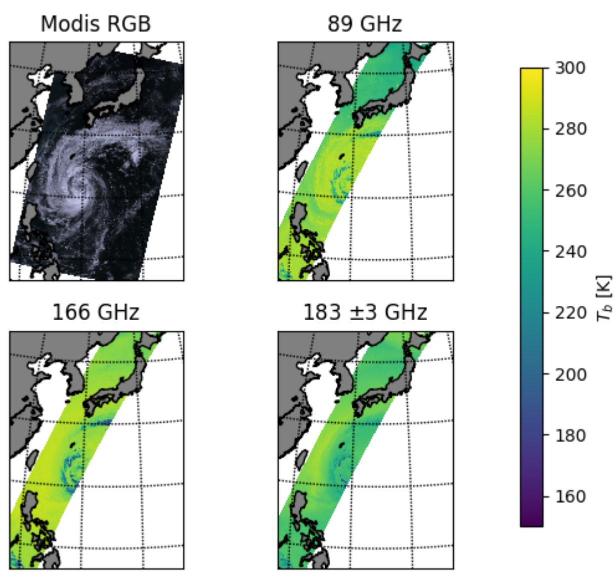

Figure 3: The tropical storm Saola seen from Modis and GMI.

Use your retrieval to retrieve the IWP path from the brightness temperatures. The functions `plot_modis_image` and `plot_gmi_swath` are provided to display the MODIS RGB and your results on a map.

In addition to the expected value, plot also the median and the 10th percentile as a lower bound for the ice water path.

4.3 Machine Learning Methods

In this part of the exercise you should use your favorite machine learning regression method to build an alternative IWP retrieval and compare it to the BMCI retrieval.

In case you are unsure what to pick, two method that should work relatively well more or less right away are *neural networks* or *regression trees*.

If you're using python, you may have a look at the *scikit-learn* examples for (boosted) decision trees or neural networks.

For MATLAB examples can be found for neural networks and regression trees, as well.

1. Comparison to BMCI

   Plot the MAPE of your machine learning retrieval and compare to the results obtained using BMCI.