Homework #3, Data Analysis and Geostatistics
Fall 2020

October 1, 2020

Overview and dataset

Download the dataset of ice velocity as a function of depth for Worthington Glacier, Alaska, located in the course Google Drive folder, under the folder "HW3".

The dataset is in an ascii file called *icevelocity.txt*, with column 1 containing the depth z in [m], and column 2 containing the associated observed ice velocity $v_o(z)$ in [m/yr].

Parameter Optimization

Parametric Statistical Models

A very common family of parametric models are polynomials:

$$v_m(z) = A_0 + A_1 z + A_2 z^2 + \cdots + A_n z^n \tag{1}$$

In this first section of the homework, you will fit polynomials, by finding the optimum parameters A_n, of varying degree n. We want a modeled estimate of ice velocity $v_m(z)$ as a function of depth z.

1. Fit polynomial models of degree 0-4 to the velocity vs. depth data using MATLAB's `polyfit.m` function. Evaluate the model at each measured depth using `polyval.m`. Plot the 5 model curves with the original data, and state the root mean squared error in the legend:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(v_m(z) - v_o(z))^2} \tag{2}$$

2. We fit these 5 models using all of the data for parameter estimation, and therefore we don't have any information about the *uncertainty* in these parameter values, nor do we know the uncertainty in the RMSE for each model. To provide estimates of uncertainties in model parameters, repeat the above, but fit the models to a random sampling of 90% of the original data. Store the parameters for each model, and repeat 1000 times (Monte-Carlo). Report the model parameters in a table, using the mean and standard deviation of the 1000 parameter estimates.

Cross-validation

A much more robust and unbiased approach to estimating the error is called *cross-validation*. Instead of calculating the model error using the same data that we used to fit the model, we fit the model to a random subset (say, 90%) of the total dataset, and use this model to predict at the other locations. The subset of depths that were not used to fit the model (the remaining 10%) are used to calculate the RMSE. Because there will be fluctuations in this error depending on how the dataset is subdivided, we repeat this procedure many times (say, 1000), as we have done before for 1-D data, to get a RMSE probability distribution.

3. Perform a *cross-validation*, using 90% of the data to fit the 5 polynomial models, and the remaining 10% of data to test, repeating 1000 times. Plot the distribution of RMSE values for each degree polynomial.

Non-parametric statistical models

4. Use a moving window average to estimate the velocity as a function of depth, and plot with the data for a window size of 3,10, and 50 meters.
5. Repeat, using a weighted moving window average (non-parametric smooth), for a window size of 3,10, and 50 meters.
6. Find the optimum window size for the weighted moving window average model.

Theoretical Ice Flow Model

Ice deforms according to the flow law of ice, which has the form

$$\dot{\epsilon}_{ij} = A\,\tau_e\,\tau_{ij}^{n-1} \tag{3}$$

where $\dot{\epsilon_{ij}}$ is the strain-rate tensor, and τ_{ij} is the deviatoric stress tensor, and A and n are the flow law parameters. A simple approximation is that of an infinite slab of ice on a slope, for which there are only shear stresses and the flow law reduces to

$$\dot{\epsilon}_{xy} = A\tau_{xy}^n \tag{4}$$

$$\frac{\partial u_x}{\partial z} = A(\rho g z \sin\theta)^n \tag{5}$$

where u_x is the velocity in the downslope direction, z is the depth below the surface, g is the acceleration due to gravity, and $\theta = 10^o$ is the surface slope. This can be integrated to get

$$u(z) = u_{x,surf} - A(\rho g \sin\theta)^n\, z^{n+1} \tag{6}$$

7. Using the measured velocity at a depth of $z = 0$ m for the surface velocity, $u_{x,surf}$, find the optimum values for the flow law parameters A and n, using the grid search (brute-force) method. Note the MATLAB function `polyfit.m` can't be used in this case.
8. Plot the root mean square (RMS) error (mean over all depths) as a function of A and n using MATLAB's `imagesc` and `colorbar` functions.
9. Find the optimum values of A and n using the gradient search method with MATLAB's `fminsearch` function.
10. Randomly sample 90% of the dataset and find the optimum value of A using the gradient search method, and repeat 1000 times. Plot the distribution of A and the RMS error (over all depths) in the model using a relative density histogram.
11. Plot the mean optimum values of A and its standard deviation with vertical errorbars on your figure from #7.
12. For each of A and model RMS error, use the normal distribution model to generate 1000 simulated values with the mean and standard deviations from your Monte-Carlo simulations.
13. Use MATLAB's `kstest2` function to compare the actual distributions from your Monte-carlo parameter fitting (#9), with those simulated assuming a normal distribution (#11).