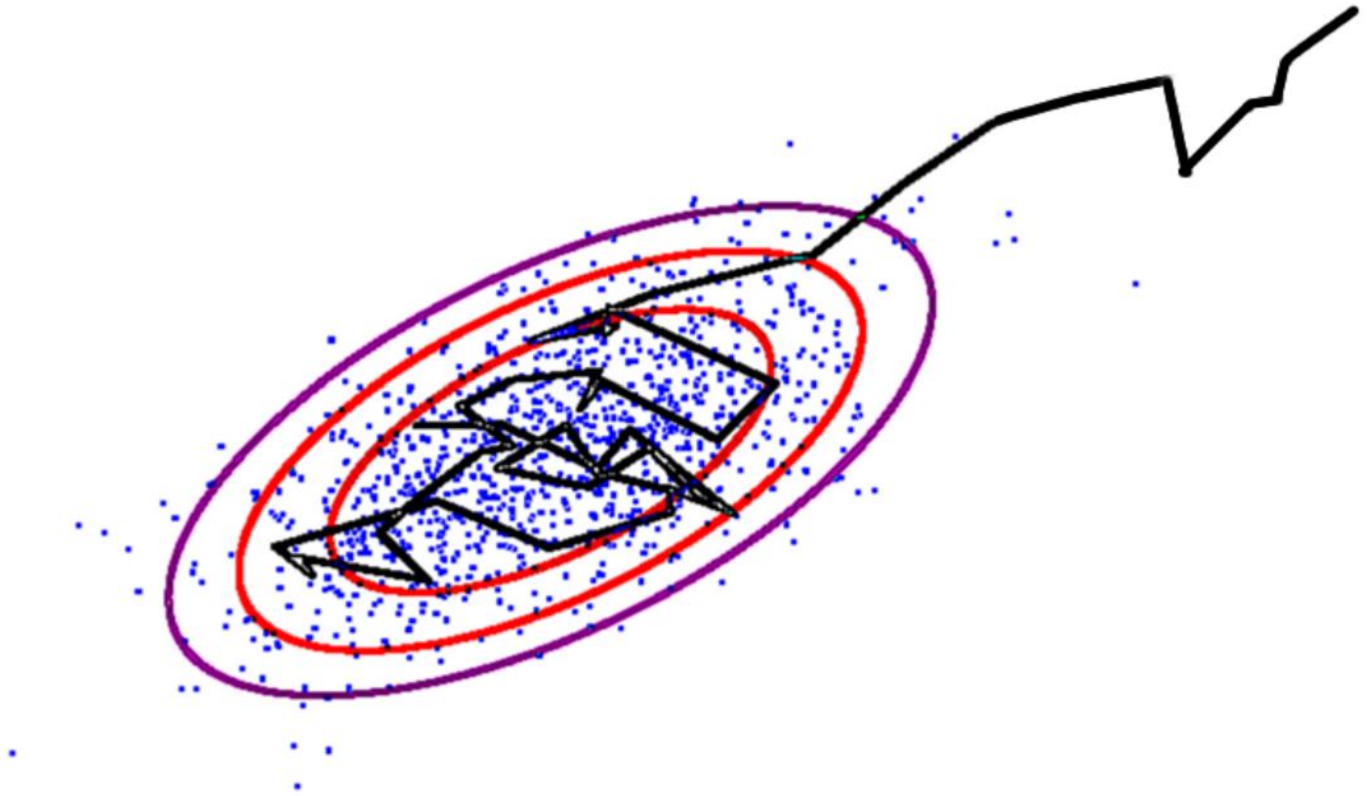
GEO 242: Numerical methods and modeling in the geosciences



Markov chain Monte Carlo

Outline

Maximum likelihood estimation

MCMC and the Metropolis-Hastings approach

MCMC

Once you have set up a robust penalty function, you can use other methods to solve it.

One such method is the Markov Chain Monte Carlo method, which effectively takes a (directed) random walk through parameter space. The areas of parameter space that are visited most frequently are the best models.

Maximum likelihood estimation

MCMC is a 'maximum likelihood' method. It enables you to identify the most likely models.

The output of MCMC estimation is a probability distribution function of each parameter in your model (known as the posterior probability distribution). You obtain this by plotting a histogram of the model parameter values visited by the algorithm.

This can be used to identify the most likely (modal) parameter values, as well as their uncertainties.

Maximum likelihood = best fitting

Assuming that errors in your data are Gaussian, it can be shown that the maximum likelihood model is the best fitting model

e.g. for a model of a straight line, with Gaussian errors, €

$$\epsilon \sim N(0, \sigma^2)$$

$$y = \theta_1 x + \theta_0 + \epsilon$$

https://towardsdatascience.com/linear-regression-91eeae7d6a2e

Maximum likelihood = best fitting

Our model can be recast in terms of a normal distribution

$$y \sim N(\theta_1 x + \theta_0, \sigma^2)$$

with probability distribution function

$$f(y|x;\theta_0,\theta_1,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(y-(\theta_1 x+\theta_0))^2}{2\sigma^2}}$$

https://towardsdatascience.com/linear-regression-91eeae7d6a2e

Maximum likelihood = best fitting

The likelihood function is the product of the probabilities of individual datapoints, x

$$L_X(\theta_0, \theta_1, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \prod_{(x,y)\in X} e^{\frac{-(y-(\theta_1 x+\theta_0))^2}{2\sigma^2}}$$

https://towardsdatascience.com/linear-regression-91eeae7d6a2e

Maximum likelihood = best fitting

You can simplify this hugely by taking logs

$$l_X(\theta_0, \theta_1, \sigma^2) = \log[\frac{1}{\sqrt{2\pi\sigma^2}} \prod_{(x,y)\in X} e^{\frac{-(y-(\theta_1 x+\theta_0))^2}{2\sigma^2}}]$$

$$= \log(\frac{1}{\sqrt{2\pi\sigma^2}}) + \sum_{(x,y)\in X} \log(e^{\frac{-(y-(\theta_1 x+\theta_0))^2}{2\sigma^2}})$$

$$= \log(\frac{1}{\sqrt{2\pi\sigma^2}}) + \sum_{(x,y)\in X} \frac{-(y-(\theta_1 x+\theta_0))^2}{2\sigma^2}$$

$$= \log(1) - \log(\sqrt{2\pi\sigma^2}) - \frac{1}{2\sigma^2} \sum_{(x,y)\in X} [y - (\theta_1 x + \theta_0)]^2$$

$$= -\log(\sqrt{2\pi\sigma^2}) - \frac{1}{2\sigma^2} \sum_{(x,y)\in X} [y - (\theta_1 x + \theta_0)]^2$$

Maximum likelihood = best fitting

The predicted value of our line is given by

$$\hat{y} = \theta_1 x + \theta_0$$

And so, our log-likelihood function is given by

$$l_X(\theta_0, \theta_1, \sigma^2) = -\log(\sqrt{2\pi\sigma^2}) - \frac{1}{2\sigma^2} \sum (y - \hat{y})^2$$

https://towardsdatascience.com/linear-regression-91eeae7d6a2e

Maximum likelihood = best fitting

To maximize the likelihood function is the same as trying to minimize its negative

$$-l_X(\theta_0, \theta_1, \sigma^2) = \log(\sqrt{2\pi\sigma^2}) + \frac{1}{2\sigma^2} \sum(y - \hat{y})^2$$

As σ is a constant, this boils down to minimizing

$$\sum(y - \hat{y})^2$$

i.e. the total squared misfit!

https://towardsdatascience.com/linear-regression-91eeae7d6a2e

Maximum likelihood = best fitting

So, in order to use a maximum likelihood algorithm to solve the problem, we want to maximize the negative of the total squared misfit!

Metropolis-Hastings algorithm

The most commonly used MCMC sampling algorithm, the Metropolis-Hastings algorithm, uses simple rules to guide its parameter space walk:

1) Draw a new step at random (from parameter standard deviations selected in advance)
2) Calculate the likelihood for the new model parameters, p_{new}
3) If the new model is more likely (better fitting), make the step
4) If the new model is less likely, draw a random number between 0 and 1, and if it is smaller than $p_{new}/p_{current}$, accept the step; if not, reject it

“Dumb Metropolis” (cf. Iain Murray)

```
function samples = dumb_metropolis(init, log_ptilde,
  iters, sigma)
D = numel(init);
samples = zeros(D, iters);
state = init;
Lp_state = log_ptilde(state);
for ss = 1:iters
    prop = state + sigma.*randn(size(state));
    Lp_prop = log_ptilde(prop);
    if log(rand) < (Lp_prop - Lp_state)
        state = prop;
        Lp_state = Lp_prop;
    end
    samples(:, ss) = state;
end
```

Metropolis-Hastings advice

Start with randomly-chosen initial parameter values (different each time)

Typically, Metropolis models have a 'burn-in' period, where it takes the algorithm some time to find the area of parameter space where the good models are – exclude this from your histograms!

There are, broadly speaking, two ways to attack MCMC problems – one long chain, or many short ones. The latter is easier to parallelize, but perhaps less 'proper'…