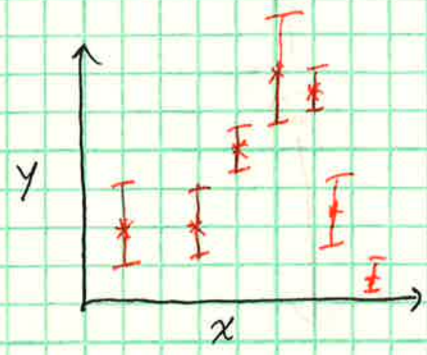
Fitting Models to Data

Sean Bryan, sean.a.bryan@asu.edu

General Goal

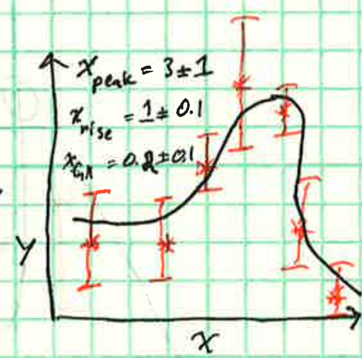


→ Think of a model

→ Write a code to calculate it →

Use the data to fix the parameters of the model

This Talk



Plan to Present the Theory Behind This

Why know the theory? Fitting can go wrong in the real world, and the theory helps debug.

→ Toy model, what if we had one or just a few data points, and our "model" was that they were all really just one number?

→ Linear fitting, what if the model is a line or if the model and data can be transformed into that case? Very fast and exact algorithm.

→ Non-linear fitting, what if the model is general, i.e. the hand drawn cartoon above? Very general, but slow algorithm. Convergence...

Vocabulary Note

Because my background is physics/astronomy, I'll use alot of language from the statistics community. Still, this entire set of notes goes over what is "really" an example of supervised machine learning.

Toy Model

Seems contrived, but actually common to have...

- Data with gaussian error bars -

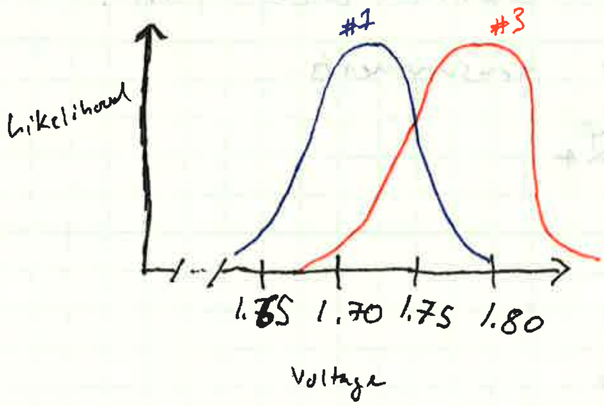
(At the end of the talk with non-linear fitting, we can actually relax this assumption. But... getting ahead of the story!)

Say we have a voltmeter with noise at the 0.05 Volt level. Say there is a voltage we want to measure, and we measure it four times. It's a single, true, unchanging voltage, but since our voltmeter has noise we get

	#1	#2	#3	#4
Measured Voltage:	1.72	1.71	1.79	1.73
Noise:	0.05	0.05	0.05	0.05

The boss just wants... the voltage... not a table. Given this dataset, what is the best one number to report? Well, we should vary the reported single voltage V_{best} until it is most likely given the four observations.

Just looking at measurement #1 and #3



(meant to draw two gaussians of the same width... sorry!)

General Gaussian Function: $\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

So, if we reported a value V_{best}, the likelihood of that being the "best" value to report given measurement #1 is

$$\frac{1}{\sqrt{2\pi(0.05)^2}}\, e^{-\frac{(V_{best}-1.72)^2}{2(0.05)^2}} \equiv \mathcal{L}_1$$

The likelihood given that measurement #3 happened is

$$\frac{1}{\sqrt{2\pi(0.05)^2}}\, e^{-\frac{(V_{best}-1.79)^2}{2(0.05)^2}} \equiv \mathcal{L}_3$$

Since probabilities multiply, the total likelihood that a value V_{best} is the right value to report is $\mathcal{L}_1 \times \mathcal{L}_3$. Now, we want to vary V_{best} until that likelihood is maximized. Looking back at the graph, reporting something like 1.72 will have $\mathcal{L}_1 \sim$ peak but $\mathcal{L}_3 \sim$ tiny. Reporting 1.79 will have $\mathcal{L}_1 \sim$ tiny but $\mathcal{L}_3 \sim$ peak. However, reporting $(1.72+1.79)/2 = 1.755$ will have $\mathcal{L}_1 = \mathcal{L}_3 \sim$ half peak, which will maximize the total product $\mathcal{L}_1 \times \mathcal{L}_3$.

So... report the average. OK... we knew that!

In general, for all four measurements.

$$\mathcal{L}_{TOT} = \mathcal{L}_1 \times \mathcal{L}_2 \times \mathcal{L}_3 \times \mathcal{L}_4$$

$$= \prod_i \mathcal{L}_i$$

Let's take the log of this

$$\ln(\mathcal{L}_{TOT}) = \ln\left(\prod_i \mathcal{L}_i\right)$$

$$= \sum_i \ln(\mathcal{L}_i)$$

An individual $\ln(\mathcal{L}_i)$ has

$$\ln\left(\frac{1}{\sqrt{2\pi\sigma_i^2}}\, e^{-\frac{(V_{best} - V_i)^2}{2\sigma_i^2}}\right)$$

$$= \ln\left(\frac{1}{\sqrt{2\pi\sigma_i^2}}\right) - \frac{(V_{best} - V_i)^2}{2\sigma_i^2}$$

The idea is to maximize $\ln(\mathcal{L}_{TOT})$ by varying V_{best}, so it is safe to ignore that first term because it doesn't change if you vary V_{best}. This means $\ln(\mathcal{L}_{TOT})$ can be written

$$\ln(\mathcal{L}_{TOT}) = -\sum_i \frac{(V_{best} - V_i)^2}{2\sigma_i^2}$$

So, to find the best V_{best}, take the derivative of $\ln(\mathcal{L}_{TOT})$ with respect to V_{best}, set it equal to zero, solve that equation for V_{best}, verify that the solution is a maximum in $\ln(\mathcal{L}_{TOT})$ by checking the second ~~derivative~~

... math.

$$V_{best}^{\text{max likelihood}} = \frac{\sum_i \frac{V_i}{\sigma_i^2}}{\sum_i \frac{1}{\sigma_i^2}} \xrightarrow[\text{equal } \sigma_i]{} \frac{1}{N}\sum V_i$$

the average.

Again... report the average... we knew that. Why?!

OK, time for something useful, linear fitting.

Linear fitting

Using the language of the toy model, we have measurements y_i (like the voltages before) at some x_i, and we want to tell the boss what line best describes all of the measurements. Each measurement has noise σ_i.

The model is

$$y_i^{model} = m x_i + b$$

The goal is to vary m and b until the model y_i^{model} is the most likely given all the measurements x_i, and y_i with noise σ_i. As before.

$$\mathcal{L}_{TOT} = \mathcal{L}_1 \times \mathcal{L}_2 \times \ldots \times \mathcal{L}_N = \prod_i \mathcal{L}_i$$

$$\ln(\mathcal{L}_{TOT}) = \sum_i \ln(\mathcal{L}_i)$$

$$\ldots$$

$$= - \sum \frac{\left[y_i - (m x_i + b)\right]^2}{2 \sigma_i^2}$$

Skipping the math means skipping:

→ Take derivative of $\ln(\mathcal{L}_{TOT})$ with respect to m and b

→ Set that all equal to zero

→ Solve the system of equations for m and b

→ Take the second derivitive of $\ln(\mathcal{L}_{TOT})$ to verify that the resulting m and b really do maximize $\ln(\mathcal{L}_{TOT})$.

Solutions for least-squares fit of a straight line:

$$a = \frac{1}{\Delta}\begin{vmatrix} \sum \frac{y_i}{\sigma_i^2} & \sum \frac{x_i}{\sigma_i^2} \\ \sum \frac{x_i y_i}{\sigma_i^2} & \sum \frac{x_i^2}{\sigma_i^2} \end{vmatrix} = \frac{1}{\Delta}\left(\sum \frac{x_i^2}{\sigma_i^2}\sum \frac{y_i}{\sigma_i^2} - \sum \frac{x_i}{\sigma_i^2}\sum \frac{x_i y_i}{\sigma_i^2}\right)$$

$$b = \frac{1}{\Delta}\begin{vmatrix} \sum \frac{1}{\sigma_i^2} & \sum \frac{y_i}{\sigma_i^2} \\ \sum \frac{x_i}{\sigma_i^2} & \sum \frac{x_i y_i}{\sigma_i^2} \end{vmatrix} = \frac{1}{\Delta}\left(\sum \frac{1}{\sigma_i^2}\sum \frac{x_i y_i}{\sigma_i^2} - \sum \frac{x_i}{\sigma_i^2}\sum \frac{y_i}{\sigma_i^2}\right)$$

$$\Delta = \begin{vmatrix} \sum \frac{1}{\sigma_i^2} & \sum \frac{x_i}{\sigma_i^2} \\ \sum \frac{x_i}{\sigma_i^2} & \sum \frac{x_i^2}{\sigma_i^2} \end{vmatrix} = \sum \frac{1}{\sigma_i^2}\sum \frac{x_i^2}{\sigma_i^2} - \left(\sum \frac{x_i}{\sigma_i^2}\right)^2$$

Uncertainties in coefficients:

$$\sigma_a^2 = \frac{1}{\Delta}\sum \frac{x_i^2}{\sigma_i^2} \qquad \sigma_b^2 = \frac{1}{\Delta}\sum \frac{1}{\sigma_i^2}$$

I don't know who first derived this, but it's presented well in the book by Bevington and Robinson from Case in Cleveland. (My alma mater!)

Key Points:

→ This is exact, no approximations were made, and there is no convergence to wait for

→ This is fast, all you do is ~N or so adds and multiplys

→ The math is messy, but thankfully someone else did it, but this can be easily generalized to any, multivariable, model, as long as the model is linear in the parameters

ex. $y_i^{model} = a + b\cos(x_i) + c\sin(t_i)$

→ numpy.polyfit partially implements this algorithm

Non-linear Fitting.

Say you have a very fancy model, possibly the model is a long fancy numerical code. Or, more boringly, say it's only a little fancy like

$$y_i^{model} = a \cos(b x_i + c)$$

Before, a key step in the linear fitting derivation was taking derivitives of $\ln(\mathcal{L}_{TOT})$ with respect to the parameters. Now that our model is non-linear, those derivitives would either come out very ugly, or in the case of a numerical code they would be impossible to take.

Instead, we will literally, actually, vary around the parameters until $\ln(\mathcal{L}_{TOT})$ is at a maximum value. And, to get error bars for the parameters, we will literally, actually, vary them a little more.

There are lots of ways to do this. The one I like best is the Monte Carlo Markov Chain (MCMC).

1) Calculate $\ln(\mathcal{L}_{TOT}(\text{params}))$

2) Throw some random numbers to generate a step to a new point in parameter space

$$z = \text{params} + (\text{random numbers})$$

3) Calculate the likelihood at that proposed step $\ln(\mathcal{L}_{TOT}(z))$

4) Throw a random number α between 0 and 1

5) Calculate the ratio $\frac{\mathcal{L}_{TOT}(z)}{\mathcal{L}_{TOT}(\text{params})}$,

that is $e^{\ln(\mathcal{L}_{TOT}(z)) - \ln(\mathcal{L}_{TOT}(\text{params}))}$

6) a) If $\alpha < e^{\ln(\mathcal{L}_{TOT}(z)) - \ln(\mathcal{L}_{TOT}(\text{params}))}$

↳ move in parameter space to z

b) else

↳ stay in parameter space at "params"

7) Repeat by going back to step 1

So... ok that was random. To see why this is useful, let's leave the whiteboard and pen and paper, and go (Finally!!!) to python.