Assignment 7: Curve Fitting and Error Analysis

due: April 4, 2020 – 11:59 PM

GOAL: The first part of the exercise includes understanding how to fit noisy data with a low order polynomial to tease out the general trend. The second part looks for oscillatory behavior using Fourrier analysis and again filtering out high frequency (wiggly) noise by a high frequency cut-off.

I Introduction

Statistical measure are used to determine if a model (or parameterization) is good and justifiable fit to the data. All good science requires this particularly when the data is noisy. Machine learning is dramatic example of this! The common test is fit and test the χ^2.

The `chi square` or χ^2 is the sum over the square of the residual , $r_i = F(x_i, \theta_j) - y_i$, of the fit relative the measure measurements y_i when the parameterize model function $F(x, \theta_n)$ with parameters $\theta_0, \theta_1, \cdots$ weighted by the expected mean error σ_i for each measurement at x_i. This is the one dimensional example. In machine learning the there is general huge vector of measured or marked training data $\vec{x}_i$ and target vector of outcomes: $(\vec{Y} = \vec{F}(\vec{x}, \theta_n))$. (e.g. Images in and number out). Here we have one function outcome. This fit is found by minimizing the chi square for this fit:

$$\chi^2 = \sum_{i=1}^{N_0} \frac{r_i^2}{\sigma_i^2} = \sum_{i=1}^{N_0} \frac{(y_i - F(x_i, \theta_n))^2}{\sigma_i^2} \tag{1}$$

The number of fitting parameters θ_n with $n = 0, 1, 2, \cdots, N$ should general be much less that the input data $N + 1 < N_0$ if the model is well chosen [1]. The question is does the fit justify the theory or model for the data. Because non-linar fits and the estimates of standard deviation σ_i is very tricky, here we do a very simple case of linear fits. Machine learning fits have huge number of parameters and test data to define the function to minimize and elaborate recursive fitting alternating linear and non-linear methods. BUT as I have emphasized at the core is a lot of basic linear algebra.

We consider only the case where the model is parameterized by a linear sum of parameters ($\theta_n = c_n$) for known functions.

$$F(x, c_n) = \sum_{n=0}^{N} c_n f_n(x) \tag{2}$$

Also we assume standard deviations, $\sigma_i's$, known and Gaussian distributed. Now gradient descent is linear algebra. Things are hardly ever this simple but this is the common starting point for modeling and error analysis! In this case the `goodness of fit` is measured by the reduced chi square per degree of freedom

$$\chi^2_{reduced} = \frac{\chi^2}{N - N_0 - 1} \tag{3}$$

[1] The conventions here are N_0 data points and $N + 1$ fitting parameters!

Note that for $N = N_0 + 1$, one should always get a perfect fit so $\chi = 0$ independent of the assumed errors telling us nothing useful. As a rule of thumb $\chi^2_{reduced} \gg 1$ is a bad fit. Either the model is wrong or the variance is underestimated. If $\chi^2_{reduced} = O(1)$, the model is a good fit assuming the variance estimates are correct. If $\chi^2_{reduced} < 1$, the model is "over-fitting" the data or the error variance has been overestimated [2]

II Coding Exercise #1: Polynomial Fits and Error Estimate

Recall in class we used gnuplot to show that temperature is rising in the data for the interval [1900 : 2015] with the commands:

```
plot "Complete_TAVG_summary.txt" using 1:2:3 with errorbars
f(x) = a + b*(x-1750)
fit [1900:2015] f(x) "Complete_TAVG_summary.txt" using 1:2:3 via a,b
replot f(x)
```

This exercise is to write your own program to do a polynomial fit to a discrete data file with N_0 values y_i, x_i, σ_i for $i = 0, 1, 2, \cdots N_0 - 1$. With errors (e.g. σ_i) you should calculate the χ^2 of our fit to see if you are able to reliably extract information from the fit.

You should use the file `Complete_TAVG_summary.txt` to test of the code. In addition on GitHub in file `HW7code` there are gnuplot scripts `plotitPoly` that solve this problem. You should write a C-code and test it against the gnuplot [3]. Of course you can test first with a smaller set of data! The problem here is to fit the N_0 data values in the time interval [1900 : 2015] of the file `Complete_TAVG_summary.txt`. The program should allow for any reasonable polynomial fit to

$$y = f(x) = c_0 + c_1 x + c_2 x^2 + \cdots c_N x^N \tag{4}$$

for $N = 0, 1, 2, \cdots N_0 - 1$

1. First fit the data with an exact fit ($N + 1 = N_0$, $\chi^2 = 0$) given by:

$$f(x) = \sum_{j=0}^{N_0-1} y_j \prod_{i \neq j} \frac{x - x_i}{x_j - x_i} \tag{5}$$

   (You should avoid doing this product with $O(N^2)$ operations! See EC526 Lecture notes and references online on how to do this.)

2. Next take the number of parameters to be much less than N_0 ($N \ll N_0$) and get a least square fit. Here don't let N be too large. Only try $N = 0$ (constant), $N = 1$ (linear), $N = 2$ s

[2] See Lecture 3 at http://astronomy.swin.edu.au/ cblake/stats.html for nice introduction to this subject

[3] More examples of gnuplot scripts for the double pendulum are moved into `HW8code` for the next problem set.

(quadratic), and $N = 3$ (cubic). Now the function is given Eq. 4. The chi square is now defined to be

$$\chi^2 = \sum_{i=0}^{N_0} \frac{(y_i - f(x_i))^2}{\sigma_i^2} = \sum_{i=0}^{N_0} \frac{(y_i - \sum_{n=0}^{N} c_n x^n)^2}{\sigma_i^2} \tag{6}$$

Visually a good $\chi^2_{reduced}$ means the fit generally goes through the error bars.

We can find the values of $c's$ that minimize the chi-square by solving the N+1 linear equations [4],

$$\sum_{m=0}^{N} A_{nm} c_m = b_n \tag{7}$$

defined by

$$A_{nm} = \sum_{i=0}^{N_0-1} \frac{x_i^{n+m}}{\sigma_i^2} \quad , \quad b_n = \sum_{i=0}^{N_0-1} \frac{x_i^n y_i}{\sigma_i^2} \tag{8}$$

for all $n, m = 0, 1, 2, ..., N$. You can solve this linear system with Gaussian elimination using the code from Problem Set #4.

The deliverables for this exercise are:

- Write a C (or C++) code to do the polynomial fit with appropriate input for `Complete_TAVG_summary.txt` and an out put file to plot the answer. It should run from a makefile.
- A fit to the first 20 terms data set in the range [1900 : 1920] using Eq. 5.This may be expensive; you can try a smaller range: 32 to 64 points is ok. Submit a plot of the fit against the data.
- A fit data set in the range [1900 : 2015] by minimizing a χ^2 for $N = 0, 1, 2, 3$. Plot each fit against each other and report the value of χ^2 and $\chi^2_{reduced}$.
- (Extra Credit) For fun you may try to fit the data set in the range [1900 : 2015] by the exact expression in Eq. 5. This is probably difficulty. Then compare it with a Jacobi solution to the linear equations which will be easier. It would fun to discuss this over zoom.

[4] By the way the proof of Eq.7 take one line of algebra. Namely it is equivalent to setting all derivatives with respect c_n,

$$\frac{1}{2}\frac{\partial \chi^2}{\partial c_n} = -\sum_{i=0}^{N_0} \frac{x_i^n (y_i - \sum_m c_m x_i^m)}{\sigma_i^2} = \sum_m \sum_{i=0}^{N_0} \frac{x_i^n x_i^m c_m}{\sigma_i^2} - \sum_{i=0}^{N_0} \frac{x_i^n y_i}{\sigma_i^2} = 0 \, .$$

to zero! See Lecture on `Curve Fitting` for more details. If you start with an over complete constraints trying to fit, $Mc \simeq y$, where there is a small vector of constants $c = \{c_0, \cdots, c_N\}$ and a large number of measurements $y = \{y0, \cdots, y_{N_0-1}\}$. M is $N \times N_0$ non-square matrix. The the least square problem is give by the linear algebra problem $Ac = A^T y$ for the $N \times N$ square matrix $A = M^T M$.

III Coding Exercise #2: Fitting Climate to Trigonometric Function

Linear fits an fit to any set of function with linear unknow coefficients.

$$y = f(x) = \sum_{n=0}^{N} c_n f_n(x) \tag{9}$$

Above the polynomial fit [5] we used function $f_n(x) = x^n$. Here we are going to use a bunch of cos() and sin() functions.

$$f(x) = a_0 + \sum_{n=1}^{N} \left[a_n \cos(\omega n) + b_n \sin(\omega n) \right]. \tag{10}$$

where again to keep to linear fits, ω is fix. In fitting `Complete_TAVG_summary.txt` let's choose $\omega = 0.01$. Again on GitHub in file `HW7code` there are gnuplot scripts `plotitFT` that solve this problem. You should write a C-code and test it against the gnuplot perhaps at first against a small set of data!

The deliverables for this exercise are:

- Write a C (or C++) code to do the trigonametric fit with appropriate input for `Complete_TAVG_summary.txt` and an out put file to plot the answer. It should run from a makefile (Exending the one above if you want).
- A fit data set in the range $[1900 : 2015]$ by minimizing a χ^2 for $N = 0, 2, 4, 8$. Plot each fit against each other and report the value of χ^2 and $\chi^2_{reduced}$.
- (Extra Credit) If you getting bored in self isolation you could describe how might find a better value of ω. For example after you do the linear fit of a_n, b_n, you could try to further minimize χ^2 by varying ω in a gradient descent. Now your are beginning down the long (and interesting) hard road to algorithms in Machine Learning with both linear and non-linear parameters [6]! If you want you could alternated a few relaxation steps on the linera parameters linear on a_n, b_n and steps in gradient descent on ω. Maybe even use Newton's method on ω! It might work or it might go off in silly direction. Fun to try?

Heads up: HW8 will return to a full Fourier Expansion and the double pendulum. I have started to put relevant code on GitHUB in preparation. One step at a time and Take Care of yourelves!

[5] The linear fit is called linear regration! Jargon Jargon. Who cares?

A Comment on Residuals: Notation, Notation, ...

In HW6 we defined the residual for a solving a linear equation $Ax = b$ as

$$r[i] = b[i] - AT[i] = b[i] - (T[i] - \frac{1}{2}[T[i-1] + T[i+1]]). \tag{11}$$

for a 1-d Laplace problem. The general expresssoin in linear algebra is

$$r = b - Ax \quad \text{or} \quad r[i] = b[i] - \sum_j A[i][j]\, x[j] \tag{12}$$

where the matrix product is give explicitly on the right. The idea see how well you are approxationg the relation $Ax \simeq b$ by subtracting $r = b - Ax \simeq 0$. Here the "unknow" parameters are $x[i]$.

The **exactly the same** defintion can be applied to our curve fitting problem $r_i = F(x_i, \theta_n) - y_i \simeq 0$. WARNING: Here parameters are θ_n and in our polynomial and FT fits c_n! Notation Notation.

A standard measure of convergence (or goodness of fit) is the quadratic from (or in statistic chi-square! [7])

$$\chi^2 = \frac{1}{2}\sum_i r[i] * r[i]/\sigma_i^2 \tag{13}$$

when there is not measure of statistical error (e.g. solving $Ax = b$ for x) we set $\sigma_i^2 = 1$

A special case in HW5 was to use the Jabic iteratstion

$$x_{new} = (1 - A)x + b \tag{14}$$

that convergnes to $x_{neew} = x$ implies $0 = -Ax + b$ in this case we can look at the change in the value of x as we iterate,

$$\Delta x = x_{new} - x = (1 - A)x + b - x = b - Ax = r \tag{15}$$

So for Jacobi, the standard residual is easy to find in each cycle of the iteraaton by just printing out the change! In other iterative methods Monitoring the change Δx is alway useful way to monitor convergence but for other iterations such as Gauss-Seidel, red-black, over relaation, Conjugate Gradient, Multi-grid etc, it defines a new (and useful) "residual". Lot's of folks still call this **the** residual even when it actually is a different measure of convergence for each method!

An asside: In solving the "Ax = b" problem the **residual is not the error**! The error is the difference, $e = x_{exact} - x$ relative to exact solution x_{exact} that satisfies $Ax_{exact} = b$. Of course in genaral we don't know this. However there is neat realatoin we will use later:

$$Ae = A(x_{exact} - x) = b - Ax = r \tag{16}$$

or $Ae = r$ at each iteration! Stay tunned!

[7] Hey it is not MY FAULT that the same idea is defined slightly different in each tribal community of scientists!

B Comment for your project

This chi square test should be use in many (of not all) of your projects to fit to the runtime of your algorithms as function of problem size n. In reality determining the scaling behavior to larger systems on complex hardware is not easy because running the same code over and over does not give the same wall clock performance for a variety of reasons. It is a good idea to treat the computer as system with random noise and at least give fit to this system. Do to this you must run many trials at each size n and compute the mean and standard deviation before fitting to scaling using the χ^2 method. The new feature is the multiple runs are done not only to get fitting data BUT also to give estimates of the standard deviations σ_k to define the χ^2.

In class we will give a code as a warm up exercise to try this out. We can choose an example from our past homework to study scaling as function of size n to do a careful estimate of the scaling of complexity (or time) $T[n]$. To do this run run multiple trial runs $k = 1, 2, \cdots, N_{trials}$ and fit $T[n]$ to the mean time and the standard deviations:

$$T_{mean}[n] = \frac{1}{N_{trials}} \sum_{k=1}^{N_{trials}} T_k[n] \quad , \quad \sigma_n^2 = \frac{1}{N_{trials} - 1} \sum_{k=1}^{N_{trials}} (T_k[n] - T_{mean}[n])^2 \tag{17}$$

The fitting now requires a model assumption such as $T[n] = c_0 + c_1 n^\theta + c_2 nlogn + \cdots$ and fitting you data by minimizing

$$\chi^2 = \sum_n \frac{(T_{mean}[n] - T[n])^2}{\sigma_n^2} \tag{18}$$

This is non-linear fit in c_0, c_1, θ_0, c_2 but gnuplot will do this fit for you. In class let's find a good example to try this on.