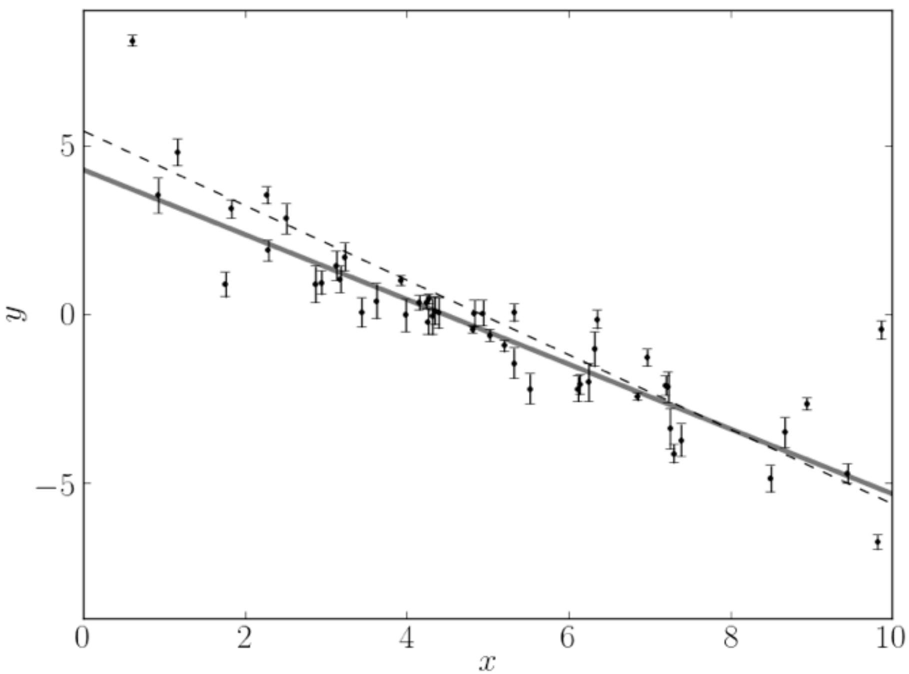
Fitting A Line: χ^2

Fitting a Line: χ^2

- Fitting a model to data is a crucial scientific technique
- Even simply fitting a line is deceptively difficult, as inference relies on subjective choices and assumptions by definition



Credit: http://dan.iel.fm/emcee/current/

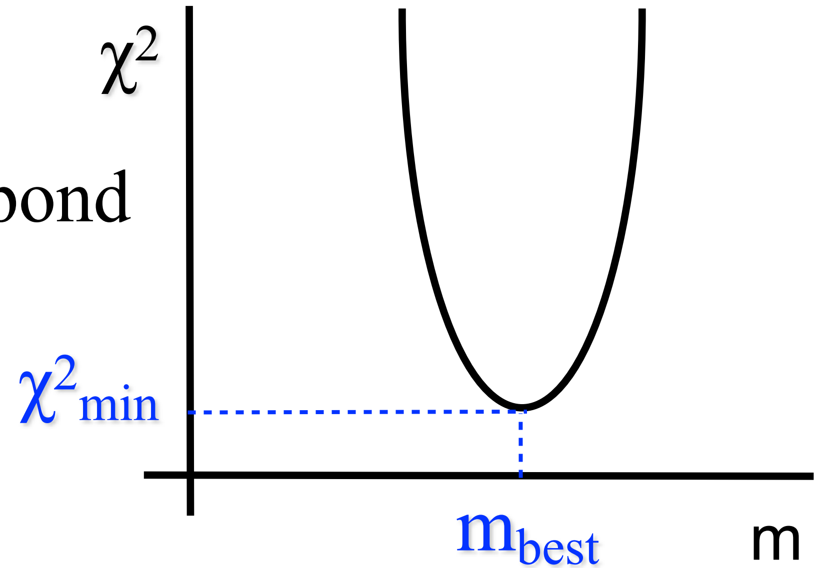
- One of the most common approaches adopted for fitting models to data is use of the χ^2 statistic, introduced by Pearson in 1900
- χ^2 is imperfect, and, among other things, using χ^2 to derive confidence intervals for *bad* fits can be problematic

Fitting a Line: χ^2

- The basic χ^2 approach is:
- bin your data into i = 1, 2, 3…*n* bins on the x-axis
  - note that even bin size is a subjective choice!
- In those x-bins, derive "observed" ("O") y-values and their *variances* (i.e. σ^2, the square of their *standard deviations* assuming Gaussian-distributed noise)
- For your model fit, derive your "expected" ("E") y-values in each x-bin (e.g., for fitting a straight-line model these would be generated by $y = E = mx + b$)
- Calculate $\chi^2 = \Sigma_i (O_i - E_i)^2/\sigma_i^2$ for a grid of your model parameters (e.g., for a straight line create a grid in *m* and *b*) and record $\chi^2(m,b)$ for each model fit

Fitting a Line: minimum χ^2 and degrees of freedom

- To determine the best fit model, simply find the set of model parameters that correspond to the *smallest* value of χ^2 (which we'll call χ^2_{min})



- A critical value associated with statistical fitting, and with the χ^2 goodness-of-fit approach, is the number of degrees of freedom, which we'll call *dof*
- Typically, if you are fitting for *n* bins of x-values and you are fitting *k* model parameters then $dof = n-k-1$
- The -1 is because we estimated one parameter set already from the data, which was the mean y-values in the x-bins

χ^2 hypothesis testing and confidence intervals

- To determine confidence limits (CLs) for your best model, calculate the probability that χ^2 for each set of model parameters exceeds a chosen value ($P(\chi^2) > \alpha$). Note that if $P(\chi^2_{min}) < \alpha$ then your model is *rejected as a fit to the data*
- A typical ("1σ") acceptance level is $\alpha = 0.32$; $P(\chi^2) > 0.32$ means that the χ^2 corresponding to those model parameters falls within the most probable 68% of the χ^2 distribution
- The fraction of the χ^2 distribution > some χ^2 value is given by *scipy.stats.chi2.sf(*χ^2*,dof)*. For χ^2 values enclosed within your CLs, *scipy.stats.chi2.sf(*$\chi^2(m,b)$*,dof)* $> \alpha$
- By finding the contour for which *scipy.stats.chi2.sf(*χ^2*,dof)* $= \alpha$, you determine the CLs on your parameters

Some χ^2 issues and the somewhat better $\Delta\chi^2$

- Using χ^2 as a goodness-of-fit statistic for confidence limits (CLs) depends on many assumptions, such as
  - were your initial errors really normally distributed? Only Gaussian noise properties will result in a χ^2 statistic drawn from the χ^2 distribution
  - is your model a good fit? If your best-fit model is almost rejected, then your CLs become tiny
- To circumvent this, it is common to derive $\Delta\chi^2$ ($\Delta\chi^2=\chi^2-\chi^2_{min}$) which is, itself, distributed according to χ^2 (So, $\Delta\chi^2=1$ gives $\alpha = 0.32$ CLs for a 1-parameter fit, $\Delta\chi^2=2.3$ for a 2-parameter fit, $\Delta\chi^2=3.5$ for a 3-parameter fit etc.)
  - Try *scipy.stats.chi2.sf(1,1)* and *scipy.stats.chi2.sf(2.3,2)*
- We'll improve on this method after Thanksgiving

Python tasks

1. In my week13 directory in Git is a file of (x,y) data called "line.data". Each of the 10 columns corresponds to an x bin of $0 < x < 1$, $1 < x < 2$, $2 < x < 3$ up to $9 < x < 10$. Each of the 20 rows is a y measurement in that x bin
   - Read in the file using *np.loadtxt* and find the mean and variance of the y measurements in each bin of x using *np.mean* and *np.var*
   - Note that *np.mean* and *np.var* can take an *axis* and that you need to pass *ddof=1* to *np.var*, because the mean has already been estimated once from the data
2. The data have been drawn from a straight line of the form $y = mx + b$ and scattered according to a Gaussian
   - Find a range of m and b values that *could* fit the data (e.g., by plotting some $y = mx + b$ model lines)

Python tasks

3. Determine χ^2 ($= \Sigma_i (O_i - E_i)^2/\sigma_i^2$) for a grid of *m* and *b* that corresponds to your range of values from above
4. Plot each of your parameters (*m* and *b*) against χ^2 and determine the best-fit model parameters
   - the best-fit parameters correspond to the minimum χ^2
5. For each pair of parameters in your grid of *m* and *b* determine the 68% ($\alpha = 0.32$ as I defined it) and 95% ($\alpha = 0.05$) confidence limits for your parameters from $\Delta\chi^2$
   - remember that you're fitting $\Delta\chi^2$ for 2 parameters
6. Plot the data with standard deviations (not variances!) as error bars. Add your best-fit model and the 68% and 95% confidence limits as lines on the plot
   - *np.std* may be useful (don't forget to pass *ddof=1*)