Learning goals of the week:

[- Fitting: Least Squares method]

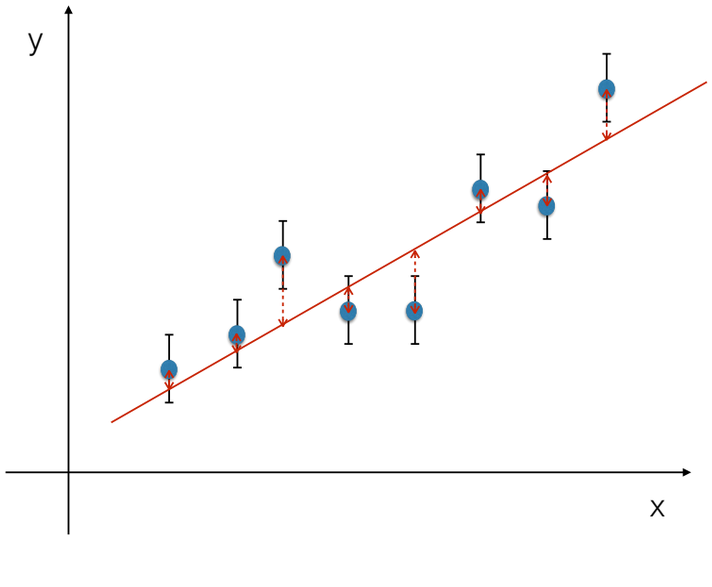
- Maximum Likelihood Fit
  - perform a likelihood fit
    - estimate the uncertainty on the best fit value
    - fit binned data
    - fit constrained parameters
- Understand similarities/differences wrt to Least Squares

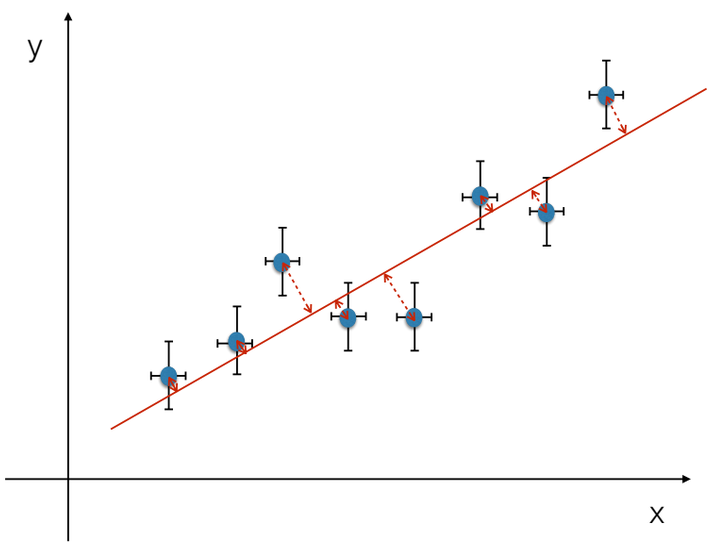
Week 5

Uncertainties on both variables: f = mx + b

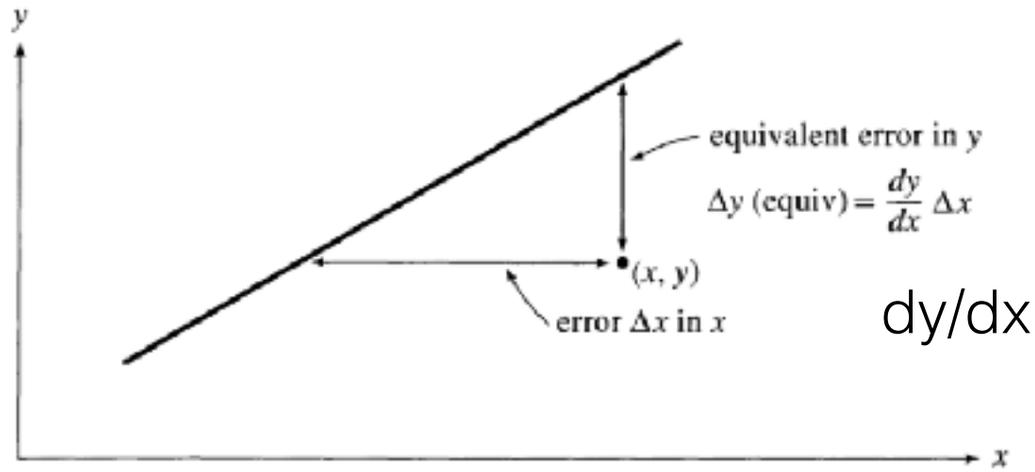
a.k.a. “total least squares”

When the uncertainty on one of the two variables cannot be neglected
($x_i \pm \sigma_{xi}$, $y_i \pm \sigma_{yi}$) the distance between the model and the points can be redefined as





The “equivalent distance/uncertainty” can be written as:



dy/dx = m

For the distance just sum in quadrature:

$$\sigma = \sqrt{\sigma_{y_i}^2 + m\sigma_{x_i}^2}$$

Uncertainties on both variables: f = mx + b

$$S(m,b) = \sum_i \frac{(y_i - mx_i - b)^2}{\sigma_{y_i}^2 + m^2\sigma_{x_i}^2}$$

Then we solve as usual $\partial S/\partial m = 0$ and $\partial S/\partial b = 0$

$$\hat{b} = \frac{\sum y_i/\kappa_i - \hat{m}\sum x_i/\kappa_i}{\sum 1/\kappa_i} \qquad \hat{m} = \frac{\sigma_x}{\sigma_y}(A \pm \sqrt{A^2+1})$$

$$\kappa_i = \sigma_{y_i}^2 + m^2\sigma_{x_i}^2. \qquad A = \frac{\sigma_x^2 V(y) - \sigma_y^2 V(x)}{2\sigma_x\sigma_y \cdot cov(x,y)}$$

$$\bar{y} = \hat{m}\bar{x} + \hat{b}$$

Notes:

- the analytical solution exists only if errors are equal at each point otherwise solve numerically
- if your model is not a simple straight line you have to compute dy/dx

Matrix notation

We can rewrite the generic chi-square definition in matrix notation as:

$$\begin{aligned}\chi^2 &= \sum_i \sum_j [y_i - f(x_i;\mathbf{a})] V_{ij}^{-1} [y_j - f(x_j;\mathbf{a})] \\ &= (\mathbf{y}-\mathbf{f})^T \mathbf{V}^{-1} (\mathbf{y}-\mathbf{f}) = \mathbf{r^T V^{-1} r}\end{aligned}$$

vector of residuals $\mathbf{r} = \mathbf{y} - \mathbf{f}$

covariance matrix $\mathbf{V}$

Which leads to a set of n-equations:

linear case : can find a closed solution

non-linear case: need to converge iteratively; may be able to linearize the problem

Practically, when working with n-equations is very convenient to “vectorize” the problem i.e. rewrite it in matrix notation and use one of the several highly optimized software-packages to solve it.

Matrix notation: linear example

Assume you want to fit a linear model: $f(x;\mathbf{a}) = \sum_r c_r(x) a_r$
where the linearity is on $\mathbf{a}$ while $c_r(x)$ can be any function

Example:
f(x|a)= $a_1+a_2x+a_3x^2+a_4\sqrt{x}$ is linear in a
f(x|a)= $a_1+\sin(x+a_2)$ is not

The chi-square can be written as:

$$\chi^2 = (\mathbf{y} - \mathbf{Ca})^T \mathbf{V}^{-1} (\mathbf{y} - \mathbf{Ca})$$

Say you have N data points and n coefficients (n≤N), then **y** and **a** are column vectors, and [**C**] = nxN and the covariance matrix [**V**] = nxn

The minimization gives:

$$\partial\chi^2/\partial\mathbf{a} = -2(\mathbf{C^T V^{-1} y} - \mathbf{C^T V^{-1} Ca}) = \mathbf{0}$$

which gives:

$$\boxed{\hat{\mathbf{a}}} = (\mathbf{C^T V^{-1} C})^{-1} \mathbf{C^T V^{-1} y} := \boxed{\mathbf{By}}$$

i.e. the solution is a linear function of the measurements **y.**

The solution is found by a simple matrix inversion ! (normal equations)

Uncertainties

You can get the covariance matrix directly as:

$$U = BVB^T = (C^T V^{-1} C)^{-1}$$

If the function is linear, then the chi2 is quadratic in a:

$$\chi^2(\mathbf{a}) = \chi^2(\hat{\mathbf{a}}) + \frac{1}{2}\sum_{i,j=1}\left[\frac{\partial^2\chi^2}{\partial a_i \partial a_j}\right]_{a=\hat{a}}(a_i - \hat{a}_i)(a_j - \hat{a}_j)$$

$$(U^{-1})_{ij} = \frac{1}{2}\left[\frac{\partial^2\chi^2}{\partial a_i \partial a_j}\right]_{\mathbf{a}=\hat{\mathbf{a}}}$$

Which in one dimension reduces to:

$$\chi^2(\mathbf{a} \pm \hat{\sigma}) = \chi^2(\hat{\mathbf{a}}) + 1 = \chi^2_{min} + 1$$

$$(a_i = \hat{a}_i \pm \hat{\sigma}_i)$$

and, as done before, reinterpret this as the uncertainty ellipse.
(in 1D it corresponds to the usual: best estimate ±1σ)

The chi-square is a paraboloid with a minimum at the best estimate of the parameters and which provides the uncertainty defined by the contour: $\chi^2_{min} + 1$

If the function is not linear the uncertainty will not be a simple ellipsoid.

Correlated measurements

Any time we perform a measurement we should use the full covariance matrix to take into account “bin to bin migrations”, sometimes we can avoid it!

Examples:

- you fill an histogram with measurements obtained with an instrument with a resolution much smaller than the bin size (diagonal covariance matrix)
- you fill an histogram with measurements obtained with an instrument with a resolution larger/worse than the bin size (non diagonal covariance matrix)

In these cases we need to take into account the off-diagonal elements of the covariance matrix.

The covariance/correlation matrix can be computed directly from the definition for each pair of bins i,j:

$$\rho_{i,j} = \frac{\sum_k (x_k^i - \bar{x}^i)(x_k^j - \bar{x}^j)}{\sqrt{\sum_k (x_k^i - \bar{x}^i)^2 \sum_k (x_k^j - \bar{x}^j)^2}}$$

(the central values come from the minimization and the variances are taken from the hessian at the minimum)

Binned X^2 fit

This is the typical application of chi-square fits to histograms (binned data).

Here interpret "f" as the pdf you want to fit. It needs to be "binned" too: integrate the pdf over the bin range and interpret it as the "expected number of events" in that bin.

$$f_i(\mathbf{a}) = n \int_{x_i^{min}}^{x_i^{max}} f(x;\mathbf{a})dx = np_i(\mathbf{a}) = E[y_i]$$

$p_i(\mathbf{a})$ is the probability for an event to be in bin "i" given the parameters **a**

n is the normalization (total number of events)

The method then applies as before:

minimize the chi-square wrt the parameters **a**;

σ_i is the uncertainty on the number of entries in bin i. *(see discussion on uncertainties Lyons 4.5)*

$$\chi^2(y_i|\mathbf{a}) = \sum_i \frac{(y_i - f_i(\mathbf{a}))^2}{\sigma_i^2}$$

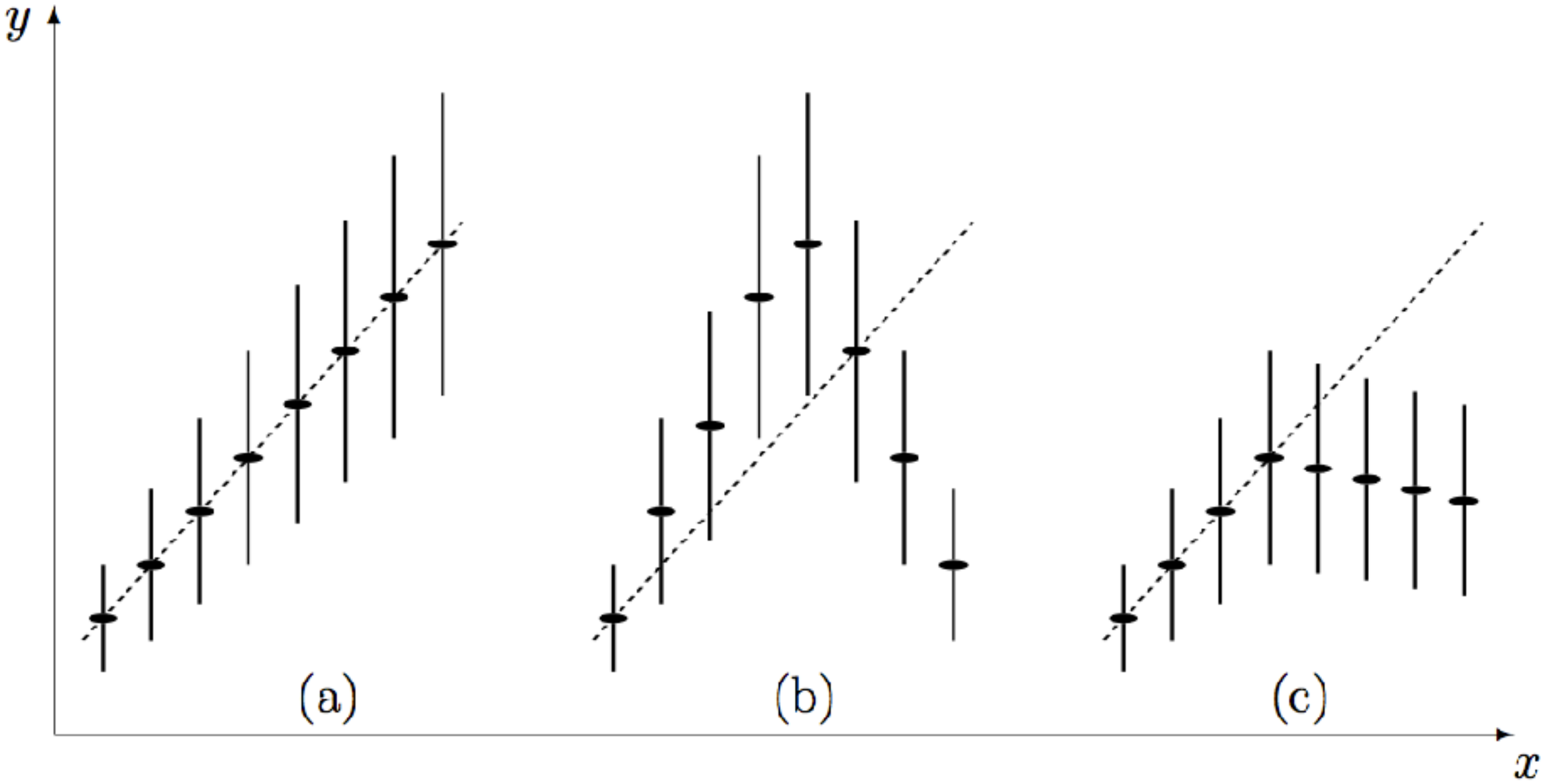
poisson uncertainty = $\sqrt{(\text{number of entries})^2}$

Careful when you have few or zero entries per bin (rule of thumb always have ≥ 5 entries/bin)

$$\chi^2(y_i|\mathbf{a}) = \sum_i \frac{(y_i - f_i(\mathbf{a}))^2}{y_i}$$

Rebin (also see later the maximum likelihood approach)

Goodness of fit



χ^2 (chi-squared) distribution

The chi-squared pdf is the joint probability of the product of n gaussians

$$f(\mathbf{x};\mu,\sigma) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left[-\frac{1}{2}\left(\frac{x_i-\mu_i}{\sigma_i}\right)^2\right]$$

$$= \exp\left[-\frac{1}{2}\sum_{i=1}^{n}\left(\frac{x_i-\mu_i}{\sigma_i}\right)^2\right] \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_i}$$

We define $\chi^2(n) = \sum_{i=1}^{n}\left(\frac{x_i-\mu_i}{\sigma_i}\right)^2$ as the chi-squared variable with **n-degrees of freedom** = number of points - parameters of the fit

The pdf is:

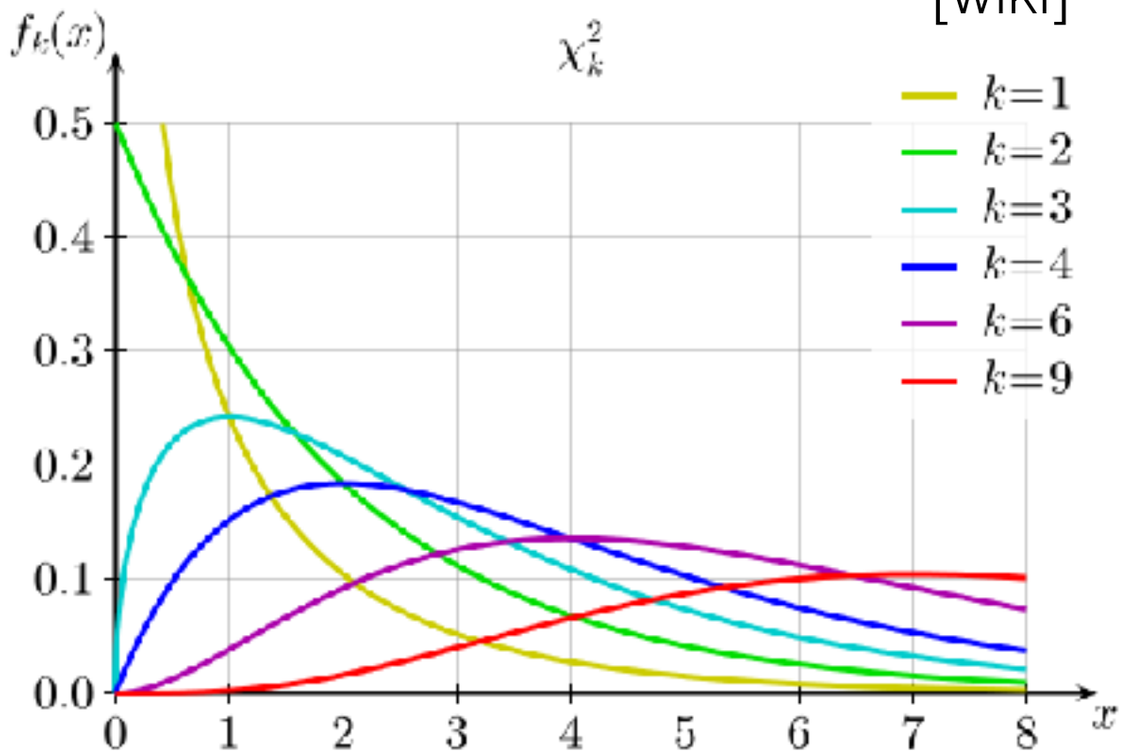
$$\chi^2(n) = f(\chi^2;n) = \frac{(\chi^2)^{n/2-1}e^{-\chi^2/2}}{\Gamma(n/2)2^{n/2}}$$

χ^2 (chi-squared) distribution

Properties:

- mean=n
- variance = 2n
- mode=n-2 for n≥2 and 0 for n≤2

[wiki]



We define: $\chi^2(n)/n$ as reduced χ^2.

For $n \to \infty$, $\chi^2(n) \to$ Gaussian(χ^2;n,2n)

Typically we can approximate the χ^2 distribution to a gaussian for $n \geq 30$.

X^2 goodness of fit

Quantitatively

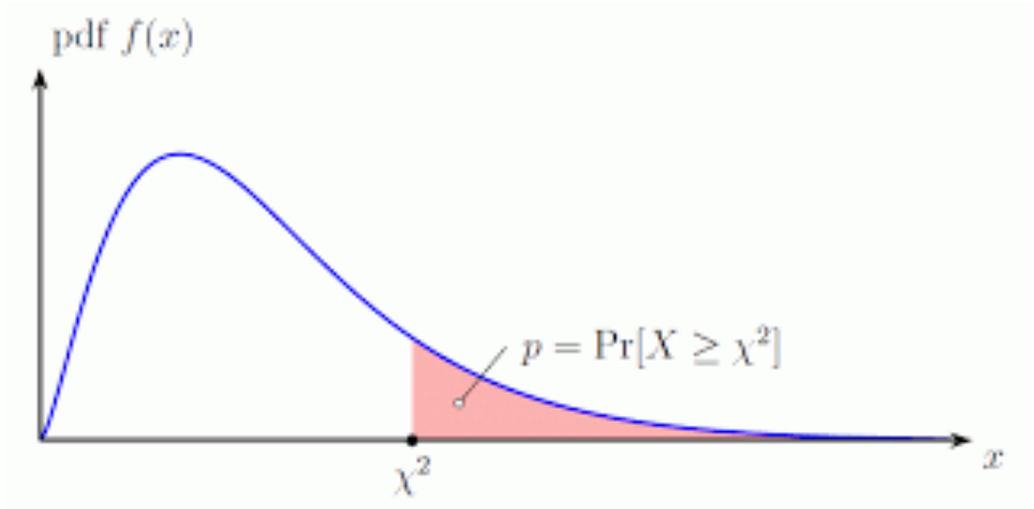
We've see the X^2 distribution before: $$f(\chi^2; n) = \frac{2^{-n/2}}{\Gamma(n/2)} \chi^{n-2} e^{-\chi^2/2}$$

The expectation value of the X^2 is n, so the number $X^2/n \sim 1$.

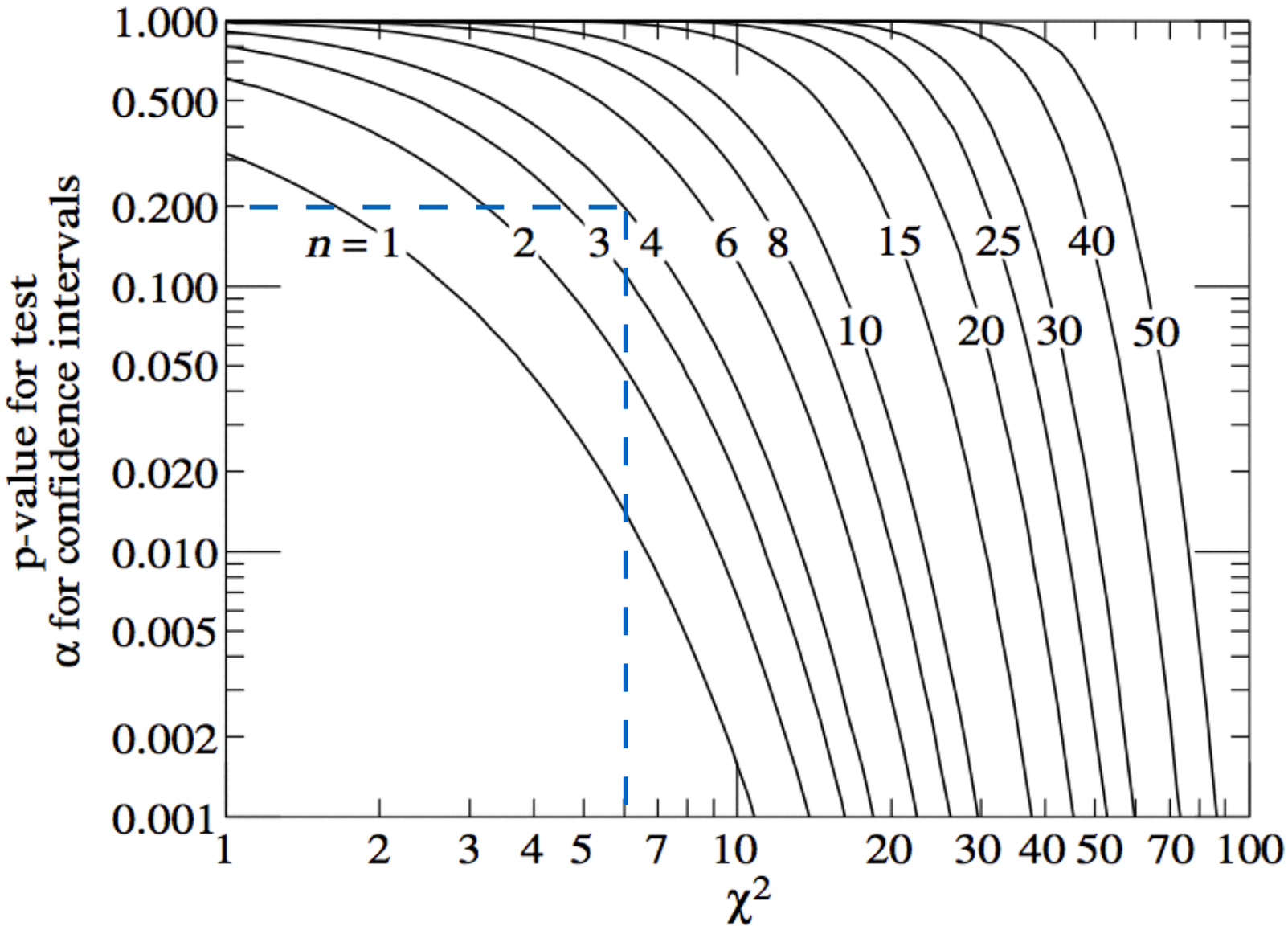
A way to quantify the goodness of fit is to see "how probable is the X^2/n we obtain", in other terms of far off in the tails of the X^2 distribution we are.

Definition: *p*-value = the probability, under the hypothesis f, of obtaining a result as incompatible with f or worse (i.e. X^2 equal or larger) than the one actually observed.

$$p = \int_{\chi^2}^{\infty} f(x'; n) dx'$$



X^2 goodness of fit



Example: n = 4, a X^2> 6 will be observed in 20% of the cases

In python:

```
>>> from scipy.stats import chi2
>>> 1 - stats.chi2.cdf (value, dof)
```

Numerical minimization: gradient descent

The problem is to find the minimum of a function $F(x|\vartheta)$ (here the X^2) in the (generally) multi-dimensional space of the parameters ϑ.

The most widely used algorithm is called "gradient descent": from a given point it computes the derivatives of F in the n-dimensional space and moves down the steepest slope towards the minimum (opposite to gradient direction)

More precisely:

- take an initial guess ϑ_0 of the minimum (typically the closer to the minimum the better)

- compute

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} F(\theta)$$

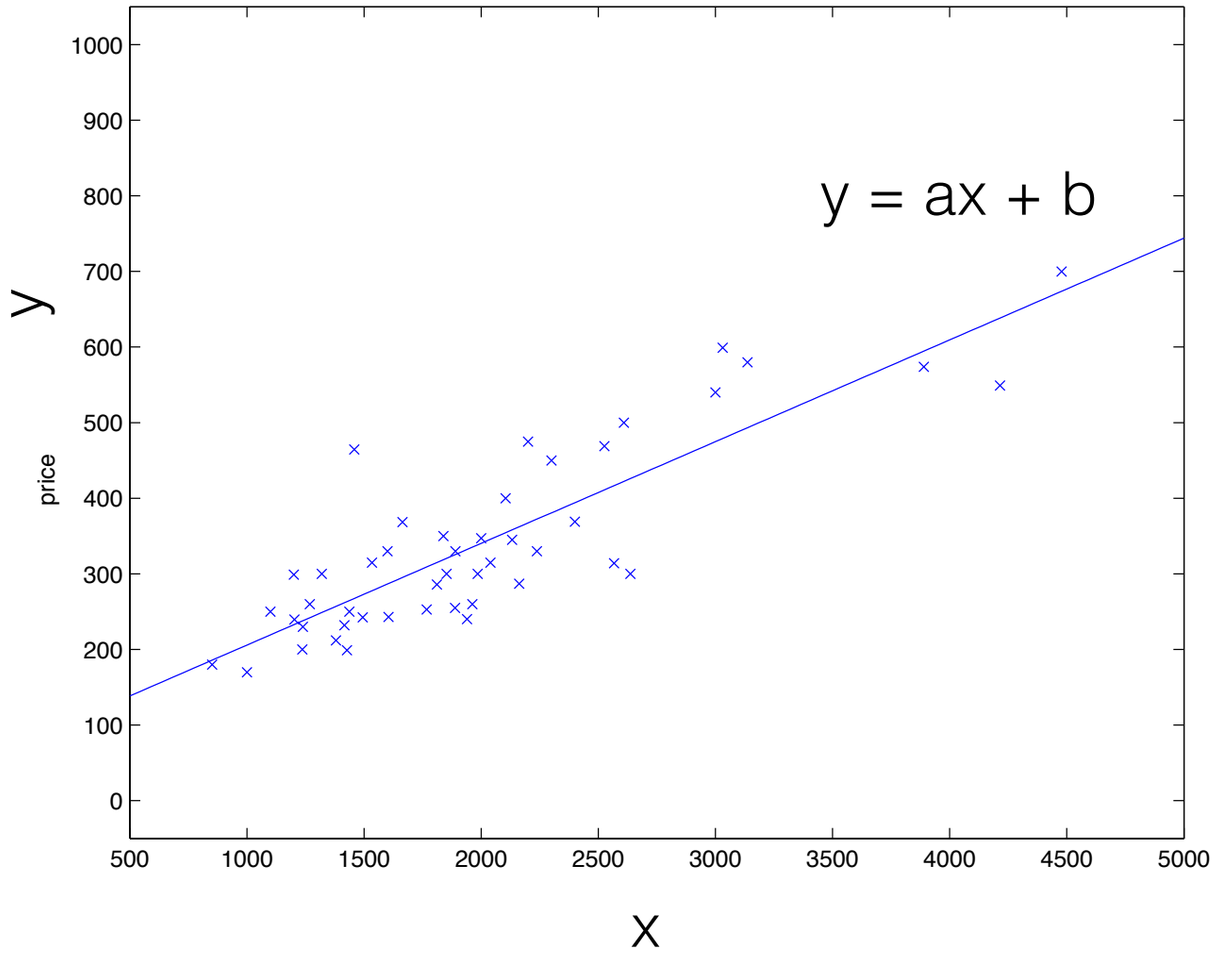
where α is a parameter called "learning rate"

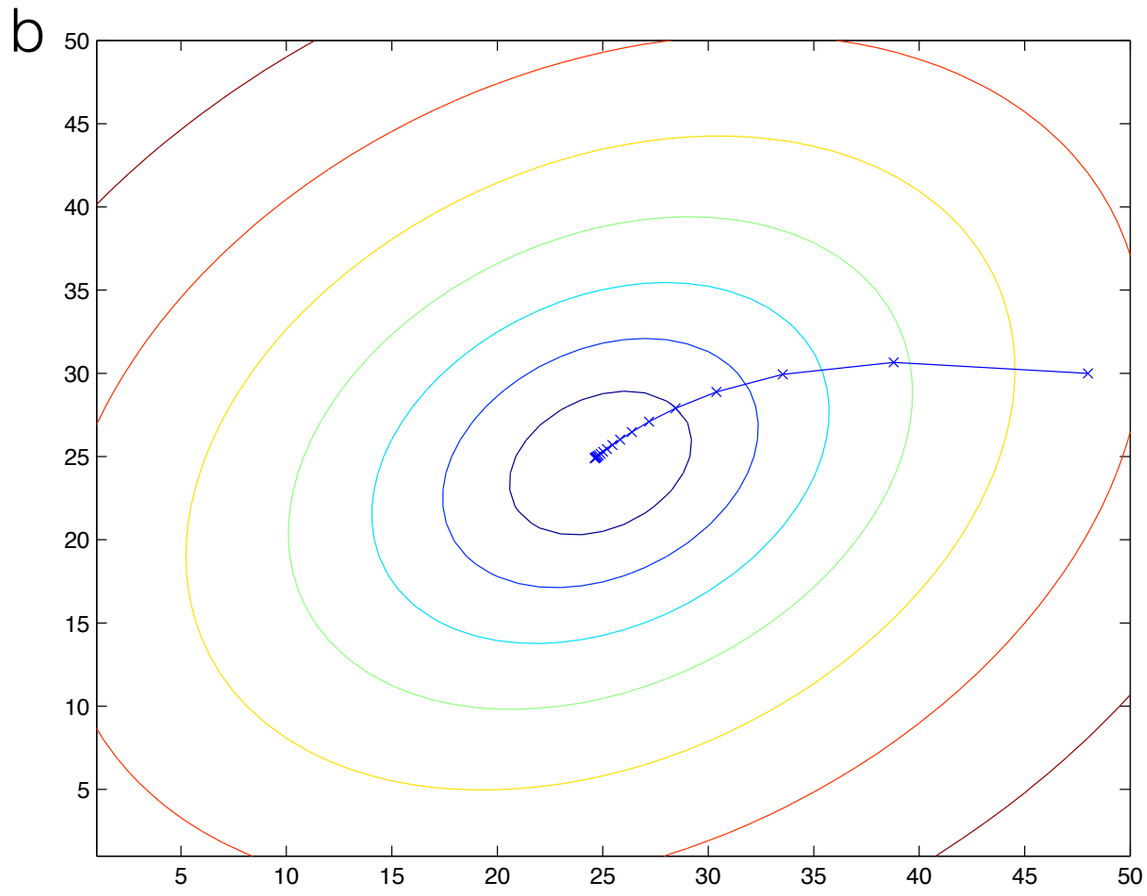
NB: here the variable is ϑ. "Computing" $F(\vartheta)$ means compute the X^2 over all points in the dataset

- until it converges: descent smaller than a tolerance value

Numerical minimization: gradient descent

[A. Ng CS229]





The size of the step is the product of the learning rate times the gradient. So “automatically” the steps taken are shorter and shorted the closer we are to the minimum

Bibliography

Error Matrix:
- Lyons Ch.3

General concepts of parameters estimation:
- Cowan Ch.5

Least Squares Fit :
- Taylor Ch. 8
- Lyons Sec 4.5

Chi2 test:
- Lyons Sec 4.6
- Taylor Ch. 12