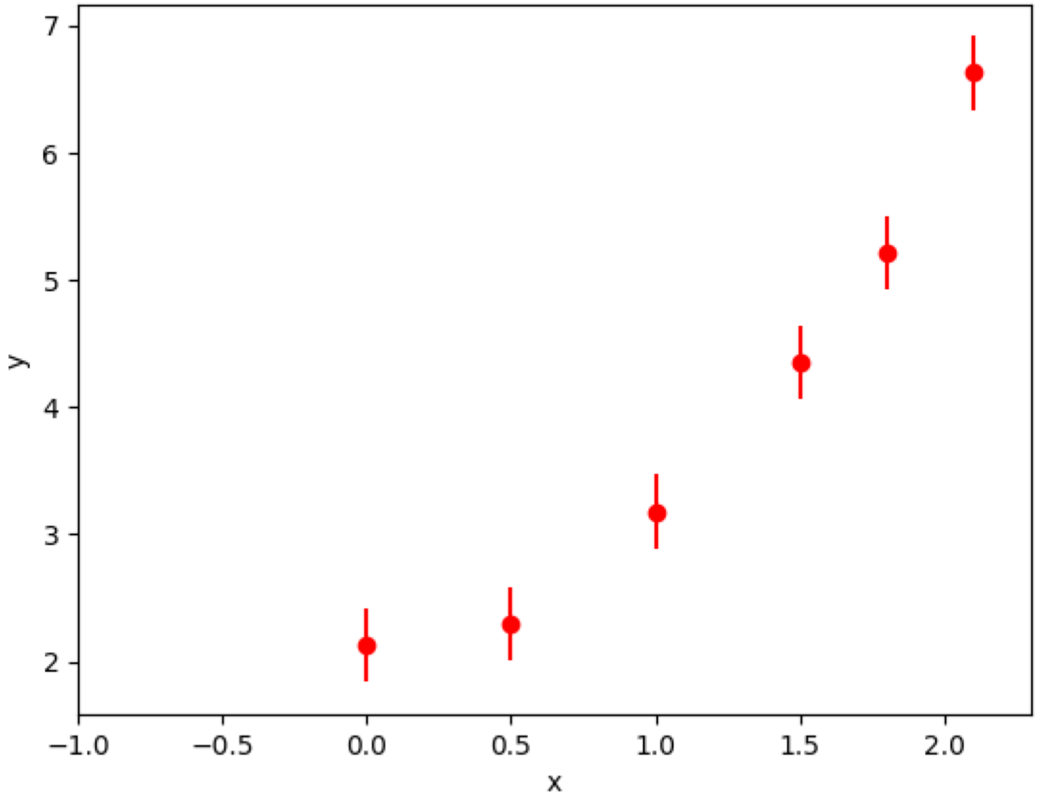
# Fitting to data

- Have N data point $\{d_i\}$ with uncertainties $\{\sigma_i\}$
- Have a model with M parameters $\{\alpha_i\}$ that can predict $\{\mu_i\}$
  - $\mu_i$ are predictions for $d_i$ and are a function of all the $\alpha$'s
- Goal: how do I estimate $\{\alpha_i\}$?

**Example**

- $d_i = y_i(x_i)$
- Looks like parabola?
  - Fit to $\mu_i = \alpha_0 + \alpha_1 x_i + \alpha_1 x_i^2$



# Fitting to data

- Have N data point $\{d_i\}$ with uncertainties $\{\sigma_i\}$
- Have a model with M parameters $\{\alpha_i\}$ that can predict $\{\mu_i\}$
  - $\mu_i$ are predictions for $d_i$ and are a function of all the $\alpha$'s
- Goal: how do I estimate $\{\alpha_i\}$?
- Must have M<N  (M=N is a special case…no fitting needed)
- Number of degrees of freedom:  *ndof = N-M*
- Find $\{\alpha_i\}$ that minimize chi-squared

$$\chi^2 = \sum \frac{(d_i - \mu_i(\vec{\alpha}))^2}{\sigma_i^2}$$

- Formula makes some sense.
- Want to see "small deviations"
- Want to give more importance to more precise measurements (smaller $\sigma$)
- But why the square?
    - Good reason for it, we may get to it later
- Note: this assumes that the $d_i$'s are not correlated

$$\chi^2 = \sum \frac{(d_i - \mu_i(\vec{\alpha}))^2}{\sigma_i^2}$$

- Without the $\sigma_i$ in the denominator (or $\sigma_i$ constant) this would be a *least square fit*

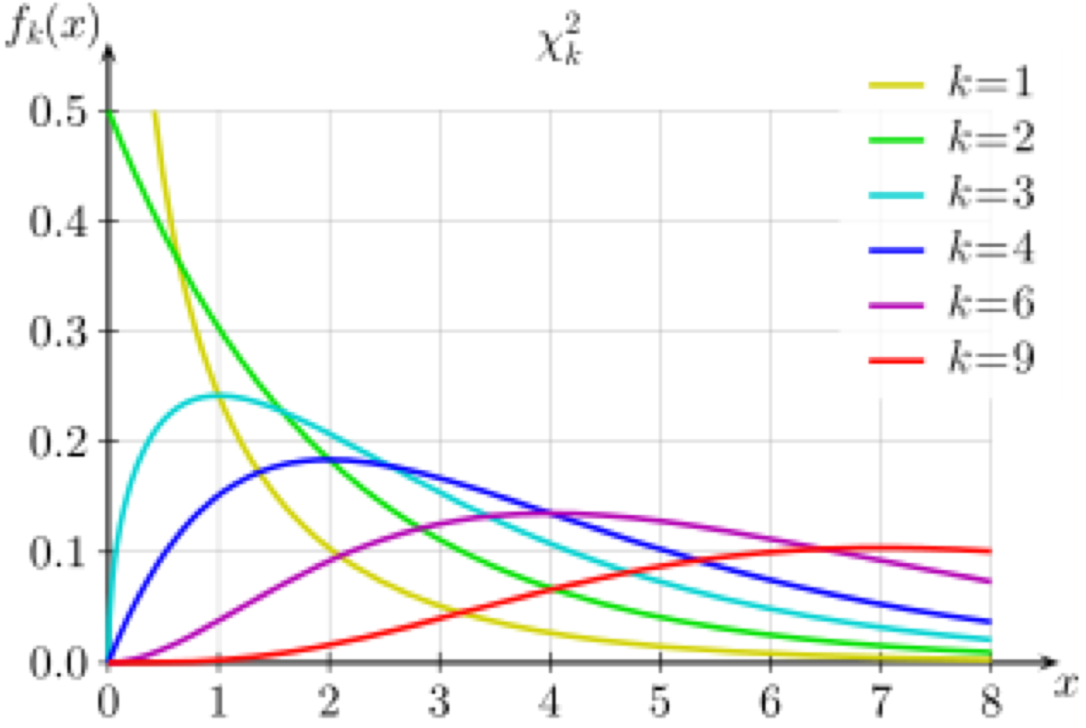
# Is the value of $\chi^2$ at min. meaningful?

$$\chi^2 = \sum \frac{(d_i - \mu_i(\vec{\alpha}))^2}{\sigma_i^2}$$

- Yes.
  - Too large: bad hypothesis (or "unlucky")
  - Too small: too good a fit (overestimated uncertainties, or got "lucky")
- Rule of thumb: expect $\chi^2 \sim$ ndof
  - (for well-behaved problem)
- Python function

```
scipy.stats.chi2.cdf(chi,ndof)
```

returns the integral from `0` to `chi` of the expected pdf for a $\chi^2$ with `ndof`




$\chi^2$ for k degrees of freedom

# Uncertainty on the $\{\alpha_i\}$

Inverse covariance matrix

$$V^{-1}(\alpha_i \alpha_j) = \frac{1}{2} \frac{\partial^2 \chi^2}{\partial \alpha_i \partial \alpha_j}$$

Where the derivatives are taken at the best fit values (Approximate, large statistics)

# Uncertainty on the $\{\alpha_i\}$

$\Delta\chi^2 = \chi^2 - \chi^2_{min}$ can be used to define contours of probability for the parameters

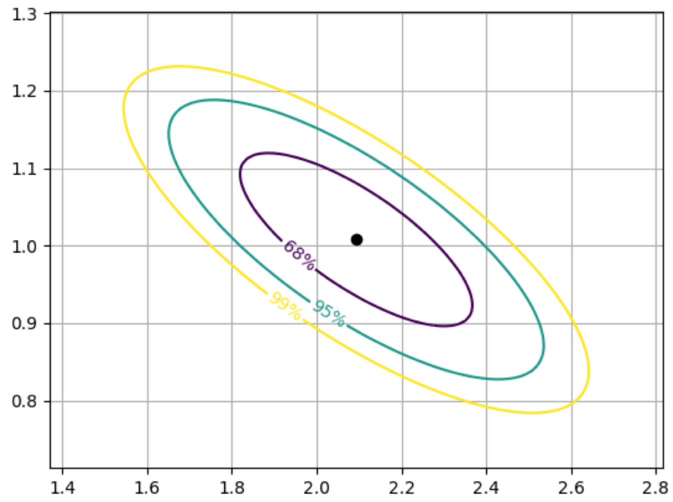
Table 38.2: Values of $\Delta\chi^2$ or $2\Delta \ln L$ corresponding to a coverage probability $1-\alpha$ in the large data sample limit, for joint estimation of $m$ parameters.

| $(1-\alpha)$ (%) | $m=1$ | $m=2$ | $m=3$ |
|---|---|---|---|
| 68.27 | 1.00 | 2.30 | 3.53 |
| 90. | 2.71 | 4.61 | 6.25 |
| 95. | 3.84 | 5.99 | 7.82 |
| 95.45 | 4.00 | 6.18 | 8.03 |
| 99. | 6.63 | 9.21 | 11.34 |
| 99.73 | 9.00 | 11.83 | 14.16 |

Inverse covariance matrix

$$V^{-1}(\alpha_i\alpha_j) = \frac{1}{2}\frac{\partial^2\chi^2}{\partial\alpha_i\partial\alpha_j}$$

Where the derivatives are taken at the best fit values (Approximate, large statistics)




Example contour for fit to 2 parameters
From one of our examples

# Fitting tools

1. `numpy.polyfit`
   - Fitting to polynomials only
   - The covariance calculation is broken in numpy 1.12.1 which is what is installed on the rpi.
   - See https://mail.scipy.org/pipermail/numpy-discussion/2013-February/065649.html
   - Newer versions are OK (with the "right" calling sequence)
2. `scipy.optimize.curve_fit`
3. `iminuit`
   - Python port of Minuit package used for the last 40+ years in HEP (!)
4. Will write our own for a simple case, to see how it works
   - Also best for special cases where speed matters