Lecture 7 outline

The method of maximum likelihood (II)

1. **Example of ML with 2 parameters**
2. **Numerical minimization with MINUIT**
3. **Extended maximum likelihood**
4. **ML with binned data**
5. **Testing goodness-of-fit with ML**
6. **Relationship to Bayesian parameter estimation**

Example of ML with 2 parameters

Consider a scattering angle distribution with $x = \cos\theta$,

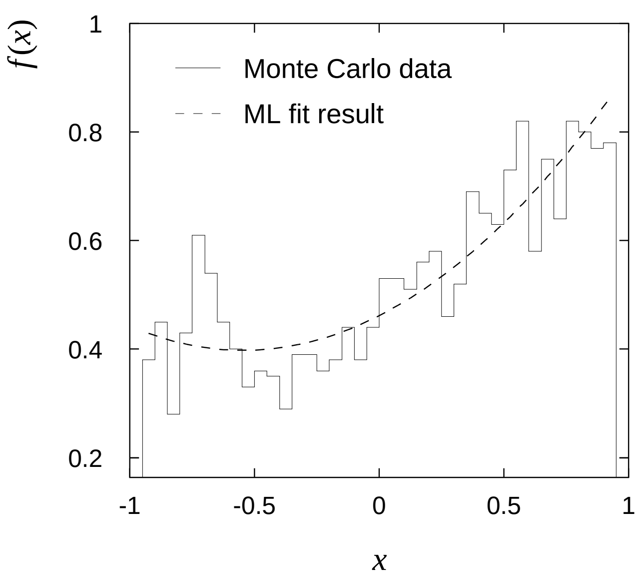
$$f(x;\alpha,\beta) = \frac{1+\alpha x+\beta x^2}{2+2\beta/3}$$

or if $x_{\rm min} < x < x_{\rm max}$, need always to normalize so that

$$\int_{x_{\rm min}}^{x_{\rm max}} f(x;\alpha,\beta)\,dx = 1.$$

Example: $\alpha = 0.5$, $\beta = 0.5$, $x_{\rm min} = -0.95$, $x_{\rm max} = 0.95$,

generate $n = 2000$ events with Monte Carlo:



Find maximum of $\log L(\alpha,\beta)$ numerically (`MINUIT`):

$\hat{\alpha} = 0.508 \pm 0.052$

$\hat{\beta} = 0.47 \pm 0.11$

$\widehat{\rm cov}[\hat{\alpha},\hat{\beta}] = 0.0026$

$r = 0.46$

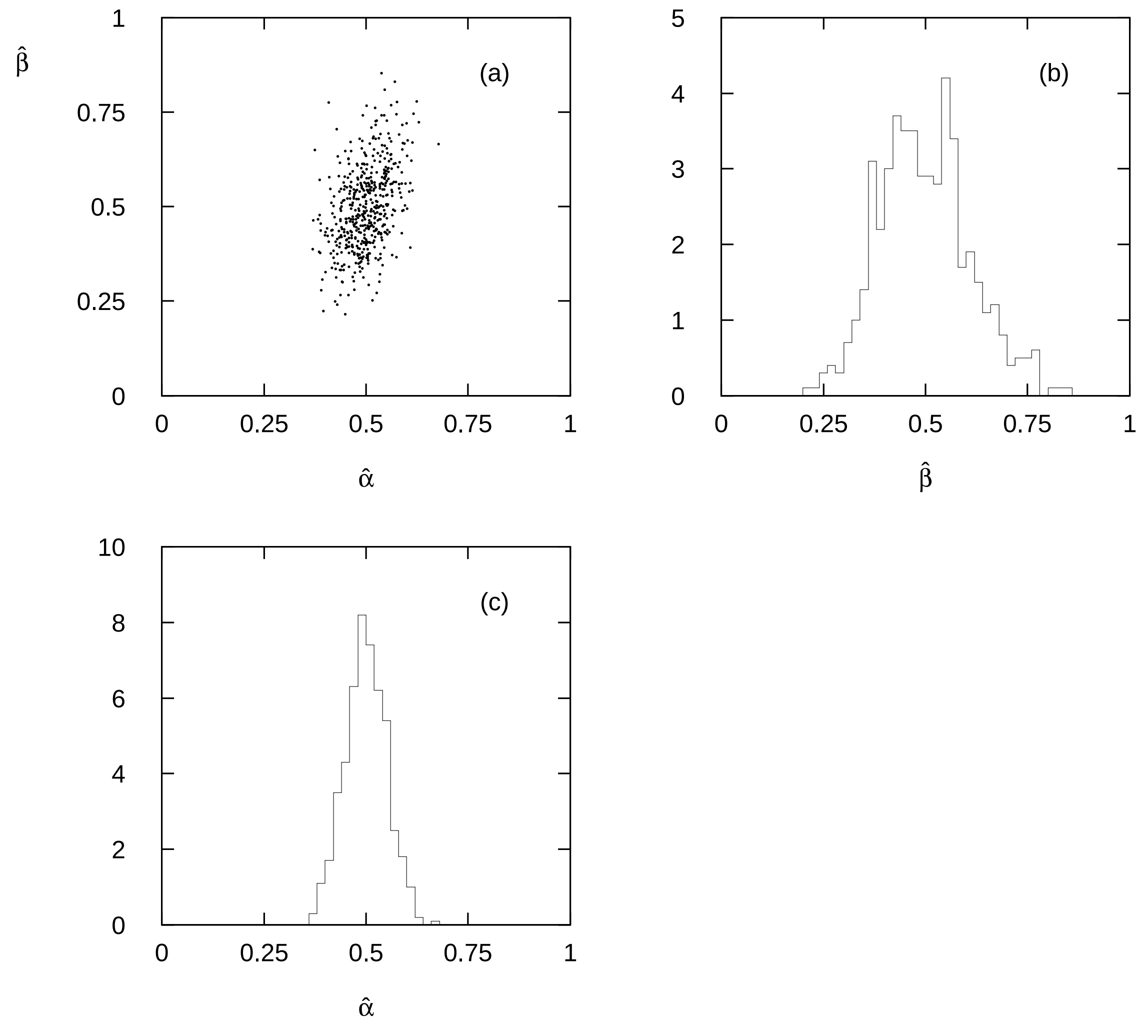
(Co)variances from $(\widehat{V^{-1}})_{ij} = -\left.\frac{\partial^2 \log L}{\partial\theta_i \partial\theta_j}\right|_{\vec{\theta}=\hat{\vec{\theta}}}$ (`MINUIT` routine `HESSE`)

The 'errors' are $\hat{\sigma}_{\hat{\alpha}}$, $\hat{\sigma}_{\hat{\beta}}$.

N.B. No binning of data for fit.

Two-parameter fit: MC study

Repeat ML fit with 500 experiments, all with $n = 2000$ events:



$$\overline{\hat{\alpha}} = 0.499 \qquad \overline{\hat{\beta}} = 0.498$$

$$s_{\hat{\alpha}} = 0.051 \qquad s_{\hat{\beta}} = 0.111$$

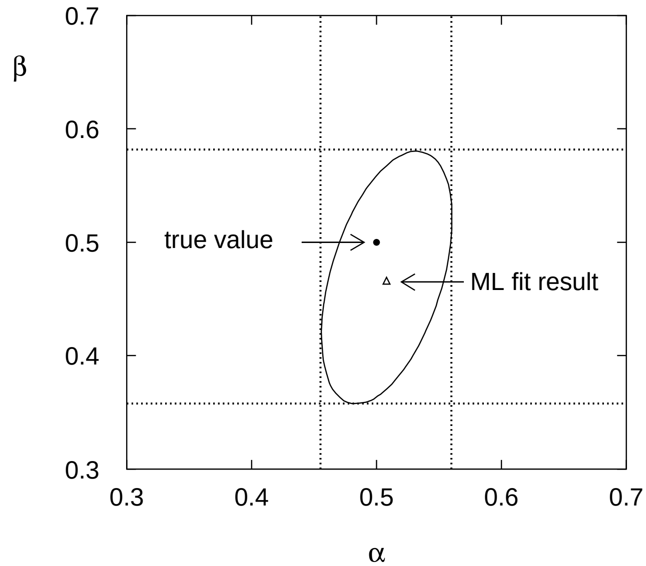
$$\widehat{\mathrm{cov}}[\hat{\alpha}, \hat{\beta}] = 0.0024 \qquad r = 0.42$$

→ $\hat{\alpha}$, $\hat{\beta}$ have positive correlation.

→ marginal pdfs approximately Gaussian.

Two-parameter fit: $\log L$ contour

The α, β plane for the first MC data set:



For large n, $\log L$ has form:

$$\log L(\alpha, \beta) = \log L_{\max}$$

$$-\frac{1}{2(1-\rho^2)}\left[\left(\frac{\alpha-\hat{\alpha}}{\sigma_{\hat{\alpha}}}\right)^2 + \left(\frac{\beta-\hat{\beta}}{\sigma_{\hat{\beta}}}\right)^2 - 2\rho\left(\frac{\alpha-\hat{\alpha}}{\sigma_{\hat{\alpha}}}\right)\left(\frac{\beta-\hat{\beta}}{\sigma_{\hat{\beta}}}\right)\right]$$

The contour $\log L(\alpha, \beta) = \log L_{\max} - 1/2$ is

$$\frac{1}{1-\rho^2}\left[\left(\frac{\alpha-\hat{\alpha}}{\sigma_{\hat{\alpha}}}\right)^2 + \left(\frac{\beta-\hat{\beta}}{\sigma_{\hat{\beta}}}\right)^2 - 2\rho\left(\frac{\alpha-\hat{\alpha}}{\sigma_{\hat{\alpha}}}\right)\left(\frac{\beta-\hat{\beta}}{\sigma_{\hat{\beta}}}\right)\right] = 1$$

→ tangent lines to contour give standard deviations $\sigma_{\hat{\alpha}}$, $\sigma_{\hat{\beta}}$.

→ angle of ellipse related to correlation: $\tan 2\phi = \dfrac{2\rho\sigma_{\hat{\alpha}}\sigma_{\hat{\beta}}}{\sigma_{\hat{\alpha}}^2 - \sigma_{\hat{\beta}}^2}$

N.B. Effect of additional correlated parameters is to increase the errors (variances of estimators).

Numerical minimization with MINUIT (1)

Often find minimum of $-\log L(\vec{\theta})$ numerically, e.g. with MINUIT:

```
      program MINUIT_FIT

      implicit        NONE

      integer            npar
      parameter          (npar = 2)

      character*10       chnam(npar)

      integer            ierr, ird, isav, istat, ivarbl, iwr
      integer            npari, nparx

      double precision   arglis (10), bnd1, bnd2, deriv(npar), dpar(npar)
      double precision   fmin, fedm, errdef, covmat (npar, npar), log_l
      double precision   FCN
      external           FCN
      double precision
      double precision   par (npar)

c  Initialize MINUIT, set print level to -1

      ird = 5          ! unit number for input to Minuit (keyboard)
      iwr = 6          ! unit number for output from Minuit (screen)
      isav = 7         ! unit number for use of the SAVE command
      call MNINIT (ird, iwr, isav)
      arglis(1) = -1.d0
      call MNEXCM (FCN, 'SET PRIN', arglis, 1, ierr, 0)

c  Define parameters alpha and beta, give initial values and step sizes.

      call MNPARM (1, 'alpha', 0.5d0, 0.1d0, 0.d0, 0.d0, ierr)
      call MNPARM (2, 'beta', 0.5d0, 0.1d0, 0.d0, 0.d0, ierr)

c  Get input data by calling FCN with iflag=1

      arglis(1) = 1.d0
      call MNEXCM (FCN, 'CALL', arglis, 1, ierr, 0)

 c  Minimize using SIMPLEX and MIGRAD, get covariance matrix with HESSE

      call MNEXCM (FCN, 'SIMPLEX', arglis, 0, ierr, 0)
      call MNEXCM (FCN, 'MIGRAD', arglis, 0, ierr, 0)
      call MNEXCM (FCN, 'HESSE', arglis, 0, ierr, 0)
```

Numerical minimization with MINUIT (2)

```
c  Get results of fit (for least squares, fmin is chi2)

      call MNSTAT (fmin, fedm, errdef, npari, nparx, istat)
      call MNPOUT (1, chnam(1), par(1), dpar(1), bnd1, bnd2, ivarbl)
      call MNPOUT (2, chnam(2), par(2), dpar(2), bnd1, bnd2, ivarbl)
      call MNEMAT (covmat, npar)

      log_l = -0.5*fmin
      write (*, *) 'Fit results:'
      write (*, *)
      write (*, *) 'alpha           = ', par(1), ' +- ',  dpar(1)
      write (*, *) 'beta            = ', par(2), ' +- ',  dpar(2)
      write (*, *) 'cov[alpha,beta] = ', covmat(1,2)
      write (*, *) 'rho[alpha,beta] = ', covmat(1,2)/(dpar(1)*dpar(2))
      write (*, *) 'log_l           = ', log_l

      stop
      END
```

User must supply subroutine FCN:

```
      subroutine FCN (npar, grad, chi2, par, iflag, futil)

c  Input:  integer           npar        number of parameters to fit
c          double precision  par(npar)   parameter vector
c          integer           iflag       select what to do
c          double precision  futil       optional external function
c
c  Output: double precision  grad(npar)  gradient vector (not filled)
c          double precision  chi2        function to be minimized

      implicit        NONE

      integer           npar,iflag
      double precision  futil, chi2, par(*), grad(*)

      integer           n_max
      parameter         (n_max = 10000)
      integer           i, n
      double precision  alpha, beta, f, log_l, x(n_max)
```

Numerical minimization with MINUIT (3)

```
c  begin

      if ( iflag .eq. 1 ) then                ! get n, array x
        call GET_INPUT_DATA (x, n, n_max)
      endif

c  calculate log-likelihood

      alpha = par(1)
      beta  = par(2)
      log_l = 0.
      do i = 1, n
        f = (1. + alpha*x(i) + beta*x(i)**2) / (2. + 2.*beta/3.)
        log_l = log_l + DLOG (f)
      end do
      chi2 = -2.*log_l                   ! 2 gets errors right

      return
      END
```

→ Must link with **MINUIT** routines in CERN library.

→ Can also be used from PAW:

interactive

various graphics options

For more info see F. James, MINUIT Reference Manual, CERN program Library Long Writeup D506, available at

http://wwwinfo.cern.ch/asdoc/

MINUIT from C++ (1)

CERNLIB provides facility to use MINUIT from C:

→ see worked example on course website

```
#include "fit.hh"    //  contains prototypes of get_data, do_fit.
const int MAX_POINTS=100;
int num_points;
double x[MAX_POINTS], y[MAX_POINTS], dy[MAX_POINTS];

 void main()
 {
   get_data (&num_points, x, y, dy);    // passed to fcn as extern
   do_fit ();
 }
```

Function **fcn** most be declared using **extern ''C''**

```
 extern "C" {void *fcn_();}
 void do_fit ()
 {
   extern int num_points;
   const int NPAR = 2;
   char chnam[10];
   int  n_dof, ierr, istat, ivarbl, npari, nparx;
   int error_flag = 0;
   double f_null = 0.;
   const int MAX_ARGS = 10;
   double chi2, arglis[MAX_ARGS], bnd1, bnd2, deriv[NPAR];
   double dpar[NPAR], fmin, fedm, errdef, par[NPAR];
   int ird = 5;
   int iwr = 6;
   int isav = 7;
   MNINIT (ird, iwr, isav);

 //  Define parameters, give initial values and step sizes.

   MNPARM (1, "alpha", 50., 1., f_null, f_null, error_flag);
   MNPARM (2, "beta", 0.5, 0.1, f_null, f_null, error_flag);

 //  Minimize chi2, get cov. mat.

   MNEXCM (fcn_, "SIMPLEX", 0, 0, error_flag, 0);
   MNEXCM (fcn_, "MIGRAD", 0, 0, error_flag, 0);
   MNEXCM (fcn_, "HESSE", 0, 0, error_flag, 0);
```

MINUIT from C++ (2)

```
//  Get results of fit

  MNSTAT (fmin, fedm, errdef, npari, nparx, istat);
  chi2 = fmin;
  n_dof = num_points - npari;
  MNPOUT (1, chnam, par[0], dpar[0], bnd1, bnd2, ivarbl);
  MNPOUT (2, chnam, par[1], dpar[1], bnd1, bnd2, ivarbl);

  cout << "Fit results:" << endl << endl;
  cout << "alpha = " << par[0] << " +- " << dpar[0] << endl;
  cout << "beta  = " << par[1] << " +- " << dpar[1] << endl;
  cout << "chi2          = " << chi2 << endl;
  cout << "n_dof         = " << n_dof << endl;
  cout << "chi2/n_dof    = " << chi2/double(n_dof) << endl;
}
```

User supplies function fcn

```
#include <iostream.h>
#include <math.h>
#include <cfortran/cfortran.h>

extern "C" { void  fcn(int npar, double grad[2], double * fcnval,
          double xval[2], int iflag, void (*Dummy)()); }

FCALLSCSUB6(fcn,FCN,fcn,INT,DOUBLEV,PDOUBLE,DOUBLEV,INT,ROUTINE)

extern "C" void fcn(int npar, double grad[2], double * fcnval,
double par[2], int iflag, void (*Dummy)())
{
  extern int num_points;
  extern double x[], y[], dy[];
  double alpha = par[0];
  double beta  = par[1];
  double chi2 = 0.;
  for(int i =0; i<num_points; i++)
  {
    double f = alpha * pow(x[i],beta);
    chi2 = chi2 + pow(y[i] - f,2) / pow(dy[i],2);
  }
  *fcnval = chi2;
}
```

Extended maximum likelihood

Up to now, sample size n considered fixed;

sometimes regard n as Poisson r.v., mean ν.

→ Result of experiment defined as $n, x_1, \ldots, x_n$.

The (extended) likelihood function is:

$$L(\nu, \vec{\theta}) = \frac{\nu^n}{n!} e^{-\nu} \prod_{i=1}^{n} f(x_i; \vec{\theta})$$

Suppose theory gives $\nu = \nu(\vec{\theta})$, drop terms not depending on θ,

$$\log L(\vec{\theta}) = n \log \nu(\vec{\theta}) - \nu(\vec{\theta}) + \sum_{i=1}^{n} \log f(x_i; \vec{\theta})$$

$$= -\nu(\vec{\theta}) + \sum_{i=1}^{n} \log(\nu(\vec{\theta})\, f(x_i; \vec{\theta}))$$

→ more info used → smaller errors for $\hat{\vec{\theta}}$.

Example: expected number of events $\nu = \sigma \int \mathcal{L}\, dt$,

cross section σ predicted as function of parameters of theory,

important e.g. for anomalous couplings in $e^+e^- \to W^+W^-$.

N.B. Before, ‘repetition of experiment’ meant with same number of events. Here, it means with same integrated luminosity (same ν).

Extended maximum likelihood: ν, $\vec{\theta}$ independent

Suppose ν, $\vec{\theta}$ are (functionally) independent:

$$L(\nu, \vec{\theta}) = \frac{\nu^n}{n!} e^{-\nu} \prod_{i=1}^{n} f(x_i; \vec{\theta})$$

$$\frac{\partial L}{\partial \nu} = 0 \quad \rightarrow \quad \hat{\nu} = n$$

$$\frac{\partial L}{\partial \theta_i} = 0 \quad \rightarrow \quad \text{same } \hat{\theta}_i \text{ as in usual ML}$$

Useful if e.g. $f(x; \vec{\theta})$ superposition of known components,

$$f(x; \vec{\theta}) = \sum_{i=1}^{m} \theta_i f_i(x)$$

Not all θ_i independent, replace e.g. θ_m by $1 - \sum_{i=1}^{m-1} \theta_i$,

→ in normal ML, θ_i not treated symmetrically.

In extended ML,

$$\log L(\nu, \vec{\theta}) = -\nu + \sum_{i=1}^{n} \log \left(\sum_{j=1}^{m} \nu \theta_j f_j(x_i) \right)$$

Define $\mu_i = \nu \theta_i$,

$$\log L(\vec{\mu}) = -\sum_{j=1}^{m} \mu_j + \sum_{i=1}^{n} \log \left(\sum_{j=1}^{m} \mu_j f_j(x_i) \right)$$

$\mu_1, \ldots, \mu_m$ = expected number of events of type i,

all parameters treated symmetrically.

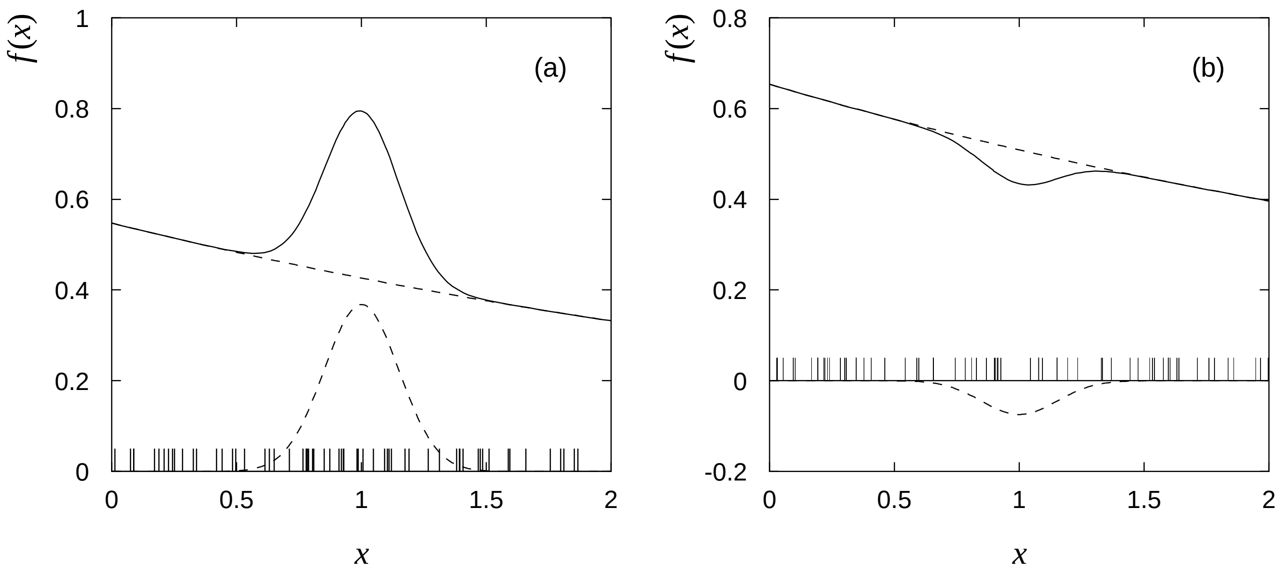
Example of extended maximum likelihood

Suppose we have 2 types of events: signal (s) and background (b):

$$f(x) = \frac{\mu_{\rm s}}{\mu_{\rm s} + \mu_{\rm b}} f_{\rm s}(x) + \frac{\mu_{\rm b}}{\mu_{\rm s} + \mu_{\rm b}} f_{\rm b}(x),$$

assume $f_{\rm s}(x)$, $f_{\rm b}(x)$ are known, we want to estimate $\mu_{\rm s}$, $\mu_{\rm b}$.

Sometimes it 'works' . . . sometimes $\hat{\mu}_{\rm s}$ negative (?!)



We can:

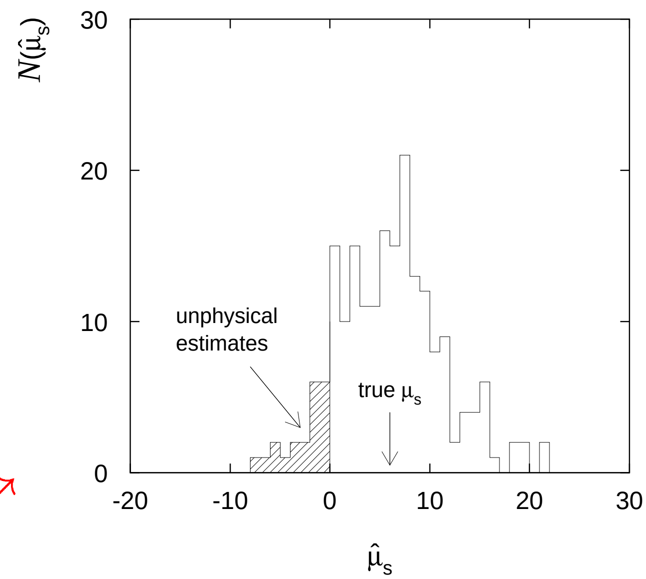
report negative $\hat{\mu}_{\rm s}$

(unphysical!),

take as estimator

$$\hat{\mu}_{\rm s}^{\rm physical} = \max(0, \hat{\mu}_{\rm s})$$

(biased!)



MC study

Maximum likelihood with binned data

Often data $\vec{x}$ in histogram: $\vec{n} = (n_1, \ldots, n_N)$, $n_{\rm tot} = \sum_{i=1}^{N} n_i$,

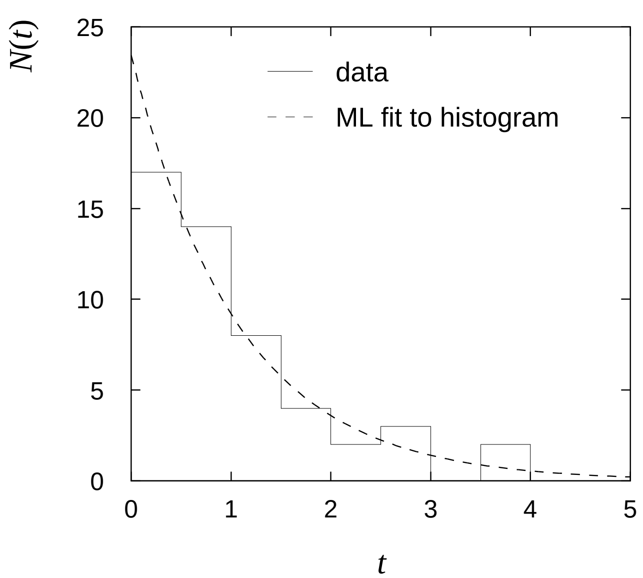
hypothesis is $\vec{\nu} = (\nu_1, \ldots, \nu_N)$, $\nu_{\rm tot} = \sum_{i=1}^{N} \nu_i$, where

$$\nu_i(\vec{\theta}) = \nu_{\rm tot} \int_{x_i^{\rm min}}^{x_i^{\rm max}} f(x; \vec{\theta}) dx$$

If joint pdf for sample multinomial ($n_{\rm tot}$ constant),

$$f_{\rm sample}(\vec{n}; \vec{\nu}) = \frac{n_{\rm tot}!}{n_1! \ldots n_N!} \left(\frac{\nu_1}{n_{\rm tot}} \right)^{n_1} \ldots \left(\frac{\nu_N}{n_{\rm tot}} \right)^{n_N}$$

$$\rightarrow \log L(\vec{\theta}) = \sum_{i=1}^{N} n_i \log \nu_i(\vec{\theta})$$



Example with exponential:

$\hat{\tau} = 1.07 \pm 0.17$

(1.06 ± 0.15 for unbinned ML with same sample)

Limit of zero bin width $\rightarrow$ usual unbinned ML.

If n_i treated as Poisson, we get extended log-likelihood,

$$\log L(\nu_{\rm tot}, \vec{\theta}) = -\nu_{\rm tot} + \sum_{i=1}^{N} n_i \log \nu_i(\nu_{\rm tot}, \vec{\theta})$$

Testing goodness-of-fit with ML

Ubinned case: difficult. Try e.g. test statistic $t = \log L_{\max}$,

OK in principle, but $g(t)$ usually not known.

$\rightarrow$ need MC study to compute P-value

Binned ML fit (or construct histogram after doing unbinned fit):

$$\hat{\nu}_i = n_{\rm tot} \int_{x_i^{\rm min}}^{x_i^{\rm max}} f(x; \hat{\vec{\theta}}) dx \qquad \text{(fit } m \text{ parameters)}$$

Consider the ratio $\lambda = \dfrac{L(\vec{n}|\vec{\nu})}{L(\vec{n}|\vec{n})} = \dfrac{f_{\rm joint}(\vec{n};\vec{\nu})}{f_{\rm joint}(\vec{n};\vec{n})}$

For e.g. multinomial or Poisson n_i, we can use test statistics

$$\chi^2_{\rm M} = -2\log\lambda_{\rm M} = 2\sum_{i=1}^{N} n_i \log\frac{n_i}{\hat{\nu}_i}$$

$$\chi^2_{\rm P} = -2\log\lambda_{\rm P} = 2\sum_{i=1}^{N} \left(n_i \log\frac{n_i}{\hat{\nu}_i} + \hat{\nu}_i - n_i \right)$$

For large data sample, both follow chi-square pdf for $N - m$ dof.

Or use Pearson's χ^2, replace ν_i by $\hat{\nu}_i = \nu_i(\hat{\vec{\theta}})$:

$$\chi^2 = \sum_{i=1}^{N} \frac{(n_i - \hat{\nu}_i)^2}{\hat{\nu}_i} \qquad \text{(Poisson: dof} = N - m)$$

$$\chi^2 = \sum_{i=1}^{N} \frac{(n_i - \hat{p}_i n_{\rm tot})^2}{\hat{p}_i n_{\rm tot}} \qquad \text{(multinomial: dof} = N - m - 1)$$

For large data sample both follow chi-square pdf (else use MC).

Relationship between ML and Bayesian parameter estimation

In Bayesian statistics, both θ and $\vec{x}$ are r.v.s:

$$L(\theta) = L(\vec{x}|\theta) = f_{\text{joint}}(\vec{x}|\theta) \quad \text{(conditional pdf for } \vec{x} \text{ given } \theta\text{)}$$

The Bayesian Method:

Use subjective probability for hypotheses (θ),

before experiment, knowledge summarized by $\pi(\theta)$ (prior pdf),

use Bayes' theorem to update prior in light of data:

$$p(\theta|\vec{x}) = \frac{L(\vec{x}|\theta)\,\pi(\theta)}{\int L(\vec{x}|\theta')\,\pi(\theta')d\theta'}$$

$p(\theta|\vec{x})$ = posterior pdf (conditional pdf for θ given $\vec{x}$)

Purist Bayesian: $p(\theta|\vec{x})$: contains all knowledge about θ.

Pragmatist Bayesian: $p(\theta|\vec{x})$ is a complicated function,

$\rightarrow$ summarize by means of estimator $\hat{\theta}_{\text{Bayes}}$

Take mode of $p(\theta|\vec{x})$, (could also use e.g. expectation value).

What do we use for $\pi(\theta)$????

No golden rule (subjective!), often represent 'prior ignorance' by

$\pi(\theta) = \text{constant} \rightarrow \hat{\theta}_{\text{Bayes}} = \hat{\theta}_{\text{ML}}$

But ... we could have used a different parameter, e.g. $\lambda = 1/\theta$.

If prior for $\pi_\theta(\theta)$ is constant, then $\pi_\lambda(\lambda)$ is not!

$\rightarrow$ 'complete prior ignorance' not well defined

The method of maximum likelihood (II)

1. **Example of ML with 2 parameters:** get (co)variance from RCF bound, contour $\log L = \log L_{\max} - 1/2$, or MC; correlated parameters increase errors.

2. **Numerical minimization with MINUIT:** use from PAW or stand alone, user writes subroutine `FCN` to compute $-\log L$ as a function of the parameters.

3. **Extended maximum likelihood:** size of data sample n treated as Poisson variable. Outcome of experiment defined to be $n, x_1, \ldots, x_n$.

4. **ML with binned data:** histogram treated as multinomial (or independent Poisson for each bin). Some information lost if bins large.

5. **Testing goodness-of-fit with ML:** difficult unless data binned (can do unbinned fit, then bin to test goodness-of-fit).

6. **Relationship to Bayesian parameter estimation:** if prior pdf constant, mode of posterior pdf is same as ML estimator.