EPJ Web of Conferences

IN2P3
Institut National de Physique Nucléaire et de Physique des Particules

SOS School Of Statistics

3^{rd} edition of the IN2P3 School Of Statistics

Autrans, 28th May - 1st June 2012

Scientific program

Probability and Statistics	Benoit CLEMENT
Introduction to Bayesian Data Analysis	Devinder S. SIVIA
Method of χ^2 and Maximum Likelihood fit	Jerome BAUDOT
Introduction to Multivariate discriminant	Balázs KÉGL
Numerical Bayesian methods	Rémi BARDENET
Boosted Decision Trees & application	Yann COADOU
Neural Networks	Jan THERHAAG
Statistics for trajectometry	Pierre BILLOIR
Unfolding: general & bayesian approaches	Francesco SPANO
Settings limits for the Higgs boson search	Luca LISTA

Organizing Committee

Jerome Baudot (IPHC)	Corinne Bérat (LPSC)
Emmanuel Busato (LPC)	Julien Donini (LPC)
Balsz Kegl (LAL)	Imad Laktineh (IPNL)
Olivier Leroy (CPPM)	Arnaud Lucotte (LPSC)

Editors: T. Delemontex and A. Lucotte

Preface

The third edition of the IN2P3/CEA School Of Statistics gives an overview of the concepts and tools used in particle physics, astro-particle physics and cosmology when probabilities and statistics come to play. In particular, this first edition was dedicated to the use of multivariate discriminants in the framework of data analyses performed in collider physics. This school was targeted towards PhD students and towards more senior physicists, aiming at extending their knowledge and skills in the field of statistical tools and frameworks developed for their fields.

The school took place in Autrans from 28th of May to 1st of June 2012. The lectures were subdivided into three parts: a part reminding the fundamental concepts used in Probabilities, Statistics and Hypothesis testing applied to physics analysis; a part focusing on the presentation of the most popular multivariate techniques and used in data analyses; and a part dedicated to frameworks devoted to the fitting and the interpretation of the results in the context of heavy flavor physics and cosmology theories.

The organizing committee would like to thank all the lecturers for their interventions and their availability during the whole week. We are also very grateful to them for accepting to produce a full article corresponding to their topics, which will serve as a reference document for future editions of the school.

The organizing committee would also like to thank Francois Le Diberder for his inspiration and his continuous support in the organization of this school. We thank our funding agencies the Institut de Physique Nucléaire et de Physique des Particules (IN2P3/CNRS), the Labex ENIGMASS and the French Society of Physics (SFP).

The organizing committee,
Jerome Baudot (IPHC, IN2P3)
Corinne Berat (LPSC, IN2P3)
Emmanuel Busato (LPC Clermont, IN2P3)
Julien Donini (LPC Clermont, IN2P3)
Balázs Kégl (LAL, IN2P3)
Imad Laktineh (IPNL, IN2P3)
Olivier Leroy (CPPM, IN2P3)
Arnaud Lucotte (LPSC, IN2P3)

Contents

Part I

Probability and statistics & Bayesian approach

Probability and statistics: a reminder

Benoit Clément
LPSC, grenoble (IN2P3)

Slides

1.1 Probability and statistics: basic concepts

Probability and Statistics

Basic concepts I

(from a physicist point of view)

Benoit CLEMENT – Université J. Fourier / LPSC

bclement@lpsc.in2p3.fr

Bibliography

Kendall's Advanced theory of statistics, Hodder Arnold Pub.

volume 1 : Distribution theory, A. Stuart et K. Ord

volume 2a : Classical Inference and and the Linear Model, A. Stuart, K. Ord, S. Arnold

volume 2b : Bayesian inference, A. O'Hagan, J. Forster

The Review of Particle Physics, K. Nakamura et al., J. Phys. G 37, 075021 (2010) (+Booklet)

Data Analysis: A Bayesian Tutorial, D. Sivia and J. Skilling, Oxford Science Publication

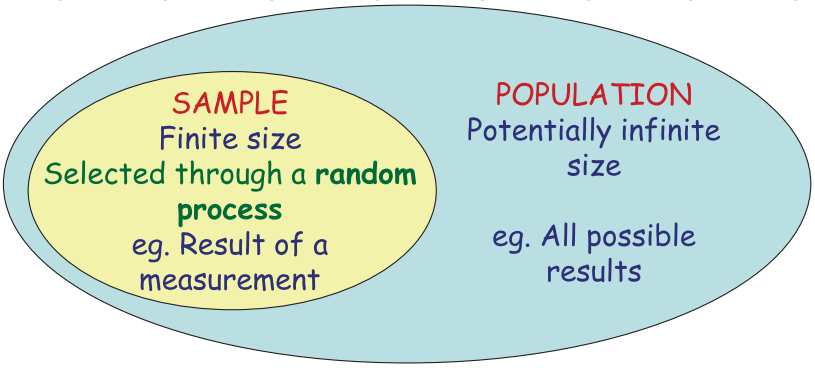
Statistical Data Analysis, Glen Cowan, Oxford Science Publication

Analyse statistique des données expérimentales, K. Protassov, EDP sciences

Probabilités, analyse des données et statistiques, G. Saporta, Technip

Analyse de données en sciences expérimentales, B.C., Dunod

Sample and population



Characterizing the sample, the population and the drawing procedure :

-> **Probability theory** (today's lecture)

Using the sample to estimate the characteristics of the population

-> **Statistical inference** (tomorrow's lecture)

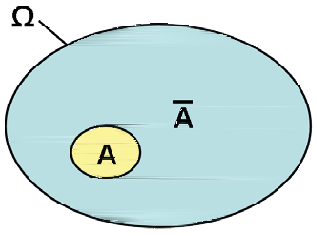
Random process

A random process (« measurement » or « experiment ») is a process whose **outcome cannot be predicted with certainty.**

It will be described by :

Universe: Ω = set of all possible **outcomes**.

Event : **logical condition on an outcome**. It can either be true or false; an event splits the universe in 2 subsets.



An event $\mathcal{A}$ will be identified by the subset **A** for which $\mathcal{A}$ is true.

Probability

A **probability function** P is defined by : (Kolmogorov, 1933)

$$P : \{Events\} \rightarrow [0:1]$$
$$A \rightarrow P(A)$$

satisfying :

$$P(\Omega)=1$$
$$P(A \cup B) = P(A) + P(B) \quad \text{if } A \cap B = \varnothing$$

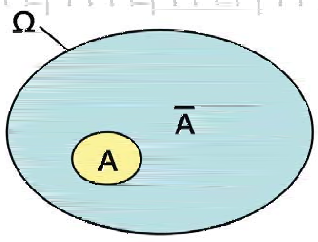
Interpretation of this number :

- **Frequentist approach** : if we repeat the random process a great number of times n , and count the number of times the outcome satisfy event A, n_A then the ratio :

$$\lim_{n \to +\infty} \frac{n_A}{n} = P(A)$$ defines a probability

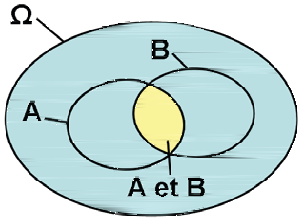
- **Bayesian interpretation** : a probability is a measure of the credibility associated to the event.

Simple logic

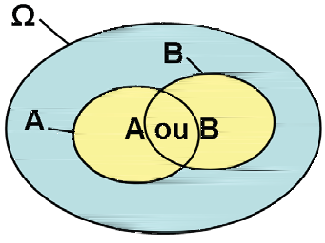


Event « not $\mathcal{A}$ » is associated with the complement $\bar{A}$.

$$P(\bar{A}) = 1-P(A)$$
$$P(\varnothing) = 1-P(\Omega) = 0$$



Event « $\mathcal{A}$ and $\mathcal{B}$ » is associated with the ensemble $A \cap B$.



Event « $\mathcal{A}$ or $\mathcal{B}$ » is associated with the ensemble $A \cup B$.

$$P(A \cup B) = P(A)+P(B)-P(A \cap B)$$

We will use and mix notations : $\mathcal{A} \leftrightarrow A$, or $\leftrightarrow \cup$, and $\leftrightarrow \cap$

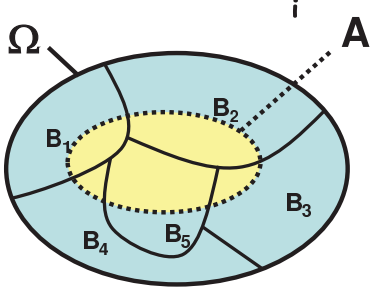
Incompatibility and partition

Two **incompatible** events cannot be true simultaneously, then :

$P(A \text{ and } B) = 0$ and $P(A \text{ or } B) = P(A)+P(B)$

A **partition** is a set of incompatible events that cover the full universe :

$$\Omega = \bigcup_i B_i, \quad B_i \cap B_j = \varnothing \quad (i \neq j)$$



Then, for any event A :

$$P(A) = \sum_i P(A \text{ and } B_i)$$

Similar to a basis in linear algebra.

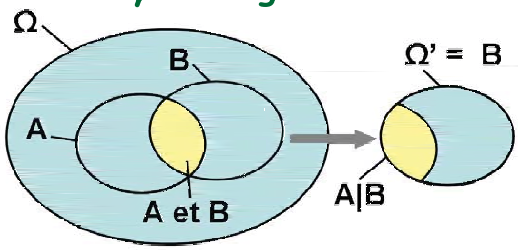
Conditional probability and independence

If an event **B** is **known to be true**, one can restrain the universe to $\Omega'=B$ and define a new probability function on this universe, the **conditional probability**.

$P(A|B)$ = « probability of A given B »

From Venn diagram :

$$P(A|B) = \frac{P(A \text{ and } B)}{P(B)}$$



Two events are **independent**, if the realization of one is not linked in any way to the realization of the other :

$P(A|B)=P(A)$ and $P(B|A) = P(B)$

From the previous relations: $P(A \text{ and } B) = P(A).P(B)$

Bayes theorem

The definition of conditional probability leads to :

$P(A \text{ and } B) = P(A|B).P(B) = P(B|A).P(A)$

Hence relating $P(A|B)$ to $P(B|A)$ by the **Bayes theorem** :

$$P(B|A) = \frac{P(A|B).P(B)}{P(A)}$$

Or, using a partition $\{B_i\}$:

$$P(B_i|A) = \frac{P(A|B_i).P(B_i)}{\sum_i P(A \text{ and } B_i)} = \frac{P(A|B_i).P(B_i)}{\sum_i P(A|B_i).P(B_i)}$$

This theorem will play a major role in Bayesian inference : given **data** and a **set of models**, it translates into :

$$P(\text{model}_i|\text{data}) = \frac{P(\text{data}|\text{model}_i).P(\text{model}_i)}{\sum_i P(\text{data}|\text{model}_i).P(\text{model}_i)}$$

Application of Bayes

100 dices in a box :

70 are equiprobable (A) 30 have a probability 1/3 to get 6 (B)

You pick one dice, throw it until you reach 6 and count the number of try. Repeating the process thrice, you get 2, 4 and 1.

What's the probability that the dice is equilibrated ?

For one throw : $P(n|A) = (1-p_6)^{n-1}p_6 = \frac{5^{n-1}}{6^n}$ $P(n|B) = \frac{2^{n-1}}{3^n}$

Combining several throw: (for one dice, throws are independents)

$$P(n_1 \text{ and } n_2 \text{ and } n_3|A) = P(n_1|A)P(n_2|A)P(n_3|A) = \frac{5^{n_1+n_2+n_3-3}}{6^{n_1+n_2+n_3}}$$

$$P(n_1 \text{ and } n_2 \text{ and } n_3|B) = \frac{2^{n_1+n_2+n_3-3}}{3^{n_1+n_2+n_3}}$$

$$P(A|n_1, n_2, n_3) = \frac{P(n_1, n_2, n_3|A)P(A)}{P(n_1, n_2, n_3|B)P(B) + P(n_1, n_2, n_3|A)P(A)}$$

$$= \frac{\frac{5^{n_1+n_2+n_3-3}}{6^{n_1+n_2+n_3}} \times 0.7}{\frac{2^{n_1+n_2+n_3-3}}{3^{n_1+n_2+n_3}} \times 0.3 + \frac{5^{n_1+n_2+n_3-3}}{6^{n_1+n_2+n_3}} \times 0.7} = \frac{\frac{5^4}{6^7} \times 0.7}{\frac{2^4}{3^7} \times 0.3 + \frac{5^4}{6^7} \times 0.7} \approx 0.42$$

Random variable

When the outcome of the random process is a **number** (real or integer), we associate to the random process, a **random variable X.**
Each realization of the process leads to a particular result : **X=x.**
x is a realization of **X.**

For a **discrete variable** :

Probability law : $p(x) = P(X=x)$

For a **real variable** : $P(X=x)=0$,

Cumulative density function : $F(x) = P(X<x)$

$dF = F(x+dx)-F(x) = P(X < x+dx) - P(X < x)$

$= P(X < x \text{ or } x < X < x+dx) - P(X < x)$

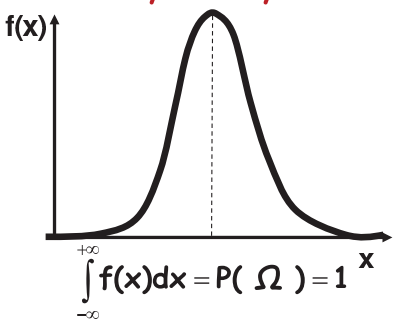
$= P(X < x) + P(x < X < x+dx) - P(X < x)$

$= P(x < X < x+dx) = f(x)dx$

Probability density function (pdf) : $f(x) = \frac{dF}{dx}$

Density function

Probability density function



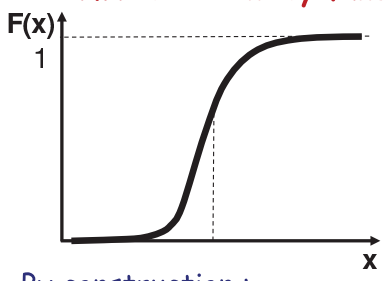
$\int_{-\infty}^{+\infty} f(x)dx = P(\Omega) = 1$

Note : discrete variables can also be described by a probability density function using Dirac distributions:

$f(x) = \sum_i p(i)\delta(i-x)$

$\sum_i p(i) = 1$

Cumulative density function



By construction :

$F(-\infty) = P(\varnothing) = 0$

$F(+\infty) = P(\Omega) = 1$

$F(a) = \int_{-\infty}^{a} f(x)dx$

$P(a < X < b) = F(b) - F(a) = \int_a^b f(x)dx$

Change of variable

Probability density function of Y = φ(X)

For φ bijective

·φ increasing : X<x ⇔ Y<y

$$P(X<x)=F_X(x)=P(Y<y)=F_Y(Y)=F_Y(\varphi(x)) \Rightarrow f_Y(y)=\frac{dF(x)}{dy}=\frac{f(x)}{\varphi'(x)}$$

·φ decreasing : X<x ⇔ Y>y

$$P(X<x)=F_X(x)=P(Y>y)=1-F_Y(Y)=1-F_Y(\varphi(x)) \Rightarrow f_Y(y)=-\frac{dF(x)}{dy}=\frac{f(x)}{-\varphi'(x)}$$

in both case $$f_Y(y)=\frac{f(x)}{|\varphi'(x)|}=\frac{f(\varphi^{-1}(y))}{|\varphi'(\varphi^{-1}(y))|}$$

If φ not bijective : split into several bijective parts φ_i

$$f_Y(y)=\sum_i \frac{f(x)}{|\varphi_i'(x)|}=\sum_i \frac{f(\varphi_i^{-1}(y))}{|\varphi_i'(\varphi_i^{-1}(y))|}$$

Very useful for Monte-Carlo : if X is uniformly distributed between 0 and 1 then $Y=F^{-1}(X)$ has F for cumulative density

Multidimensional PDF (1)

Random variables can be generalized to random vectors :

$$\vec{X}=(X_1,X_2,\ldots,X_n)$$

the **probability density function** becomes :

$$f(\vec{x})d\vec{x} = f(x_1,x_2,\ldots,x_n)dx_1dx_2\ldots dx_n$$
$$= P(x_1<X_1<x_1+dx_1 \text{ and } x_2<X_2<x_2+dx_2\ldots$$
$$\ldots \text{and } x_n<X_n<x_n+dx_n)$$

and $$P(a<X<b \text{ and } c<Y<d)=\int_a^b dx\int_c^d dy\ f(x,y)$$

Marginal density : probability of only one of the component

$$f_X(x)dx=P(x<X<x+dx \text{ and } -\infty<Y<+\infty)=\int(f(x,y)dx)dy$$
$$\Rightarrow f_X(x)=\int f(x,y)dy$$

Multidimensional PDF (2)

For a fixed value of $Y=y_0$:

$f(x|y_0)dx$ = « Probability of $x<X<x+dx$ knowing that $Y=y0$ »

is , a **conditional density** for X. It is proportional to f(x,y), so

$$f(x|y) \propto f(x,y) \qquad \int f(x|y)dx = 1$$

$$\Rightarrow f(x|y) = \frac{f(x,y)}{\int f(x,y)dx} = \frac{f(x,y)}{f_y(y)}$$

The two random variables X and Y are **independent** if all events of the form $x<X<x+dx$ are independent from $y<Y<y+dy$

$f(x|y)=f_X(x)$ and $f(y|x)=f_Y(y)$ hence $f(x,y)= f_X(x).f_Y(y)$

Translated in term of pdf's, Bayes' theorem becomes:

$$f(y|x) = \frac{f(x|y)f_Y(y)}{f_X(x)} = \frac{f(x|y)f_Y(y)}{\int f(x|y)f_Y(y)dy}$$

D.Sivia's lecture will detail the use of this formula for statistical inference

Sample PDF

A **sample** is obtained from a **random drawing** within a **population**, described by a probability density function.

We're going to discuss how to **characterize, independently from one another:**

- a **population**
- a **sample**

To this end, it is useful, to consider a sample as a finite set from which one can randomly draw elements, with equipropability

We can the associate to this process a probability density, the **empirical density** or **sample density**

$$f_{sample}(x) = \frac{1}{n}\sum_i \delta(x-i)$$

This density will be useful to translate properties of distribution to a finite sample.

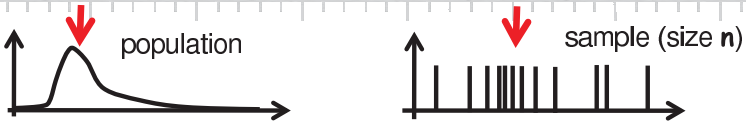
Characterizing a distribution

How to reduce a distribution/sample to a finite number of values ?

- **Measure of location**:
  Reducing the distribution to **one central value**
  -> **Result**
- **Measure of dispersion**:
  Spread of the distribution around the central value
  -> **Uncertainty/Error**
- **Higher order measure of shape**

- **Frequency table/histogram** (for a finite sample)

Measure of location



Mean value : Sum (integral) of all possible values weighted by the probability of occurrence:

$$\mu = \bar{x} = \int_{-\infty}^{+\infty} x f(x) dx \qquad \mu = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Median : Value that split the distribution in 2 equiprobable parts

$$x_1 \le x_2 \le \ldots \le x_n$$

$$\int_{-\infty}^{med(x)} f(x)dx = \int_{med(x)}^{+\infty} f(x)dx = \frac{1}{2} \quad med(x) = \begin{cases} x_{(n+1)/2} & , \text{ odd } n \\ \frac{1}{2}(x_{n/2} + x_{1+n/2}), & \text{even } n \end{cases}$$

Mode : The most probable value = maximum of pdf

$$\left.\frac{df}{dx}\right|_{x=mod(x)} = 0, \quad \left.\frac{d^2f}{dx^2}\right|_{x=mod(x)} < 0 \qquad ?$$

Measure of dispersion

Standard deviation (σ) and variance (v= σ²) : Mean value of the squared deviation to the mean :

$$v = \sigma^2 = \int (x-\mu)^2 f(x)dx \qquad v = \sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2$$

Koenig's theorem :

$$\sigma^2 = \int x^2 f(x)dx + \mu^2 \int f(x)dx - 2\mu \int x f(x)dx$$

$$\boxed{\sigma^2 = \overline{x^2} - \mu^2 = \overline{x^2} - \overline{x}^2}$$

Interquartile difference : generalize the median by splitting the distribution in 4 :

$$\int_{-\infty}^{q_1} f(x)dx = \int_{q_1}^{q_2} f(x)dx = \int_{q_2}^{q_3} f(x)dx = \int_{q_3}^{+\infty} f(x)dx = \frac{1}{4} \qquad \text{med}(x) = q_2 \qquad \delta = q_3 - q_1$$

Others...

Bienaymé-Chebyshev

Consider the interval : $\Delta =]-\infty, \mu-a[\cup]\mu+a, +\infty[$

Then for $x \in \Delta$:

$$\left(\frac{x-\mu}{a}\right)^2 > 1 \Rightarrow \left(\frac{x-\mu}{a}\right)^2 f(x) > f(x)$$

$$\Rightarrow \int_\Delta \left(\frac{x-\mu}{a}\right)^2 f(x)dx > \int_\Delta f(x)dx$$

$$\Rightarrow \int_{-\infty}^{+\infty} \left(\frac{x-\mu}{a}\right)^2 f(x)dx > \int_\Delta f(x)$$

$$\Rightarrow \frac{\sigma^2}{a^2} > P(|X-\mu| > a)$$

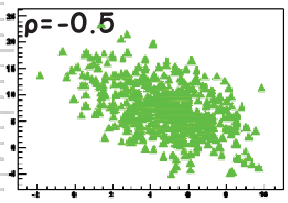
Finally **Bienaymé-Chebyshev's inequality** $\boxed{P(|X-\mu| \le a\sigma) > 1 - \frac{1}{a^2}}$

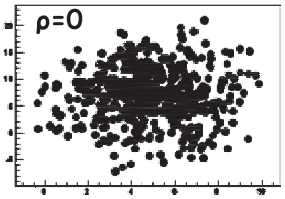
It gives a bound on the **confidence level** if the interval $\mu \pm a\sigma$

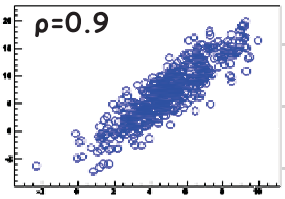
a	1	2	3	4	5
Chebyshev's bound	0	0.75	0.889	0.938	0.96
Normal distribution	0.683	0.954	0.997	0.99996	0.9999994

Multidimensional case

A **random vector (X,Y)** can be treated as **2 separate variables**
marginal densities : mean and variance for each variable : μ_X μ_Y σ_X σ_Y
Doesn't take into account correlations between the variables







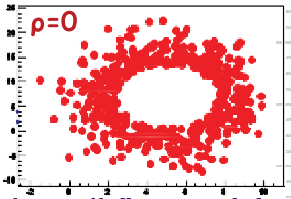
Generalized measure of dispersion : **Covariance of X and Y**

$$Cov(X,Y) = \iint (x-\mu_X)(y-\mu_Y) f(x,y)\,dx\,dy = \rho\sigma_X\sigma_Y = \mu_{XY} - \mu_X\mu_Y$$

$$Cov(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu_X)(y_i-\mu_Y)$$

Correlation : $\rho = \dfrac{Cov(X,Y)}{\sigma_X\sigma_Y}$ **Uncorrelated** : $\rho=0$

Independent ⇒ **Uncorrelated**



only quantify linear correlation

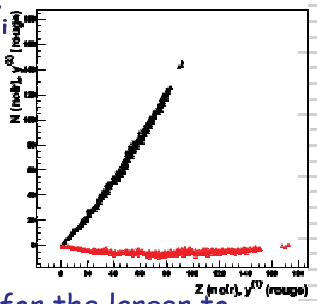
Decorrelation

Covariance matrix for **n** variables X_i:

$$\Sigma_{ij} = Cov(X_i,X_j) \Rightarrow \Sigma = \begin{pmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 & \cdots & \rho_{1n}\sigma_1\sigma_n \\ \rho_{12}\sigma_1\sigma_2 & \sigma_2^2 & \cdots & \rho_{2n}\sigma_2\sigma_n \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{1n}\sigma_1\sigma_n & \rho_{2n}\sigma_2\sigma_n & \cdots & \sigma_n^2 \end{pmatrix}$$

For **uncorrelated variables** Σ **is diagonal**
Matrix **real** and **symmetric** : can de diagonalized
On can define **n** new uncorrelated variables Y_i

$$\Sigma' = \begin{pmatrix} \sigma'^2_1 & 0 & \cdots & 0 \\ 0 & \sigma'^2_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma'^2_n \end{pmatrix} = B^{-1}\Sigma B, \qquad Y = BX$$



σ'^2_i are the **eigenvalues** of Σ,
B contains the **orthonormal eigenvectors.**
The Y_i are the **principal components**. Sorted for the larger to the smaller σ' they allow **dimensional reduction**

Regression

Measure of location:

- a point : (μ_X , μ_Y)
- a curve : line closest to the points -> **linear regression**

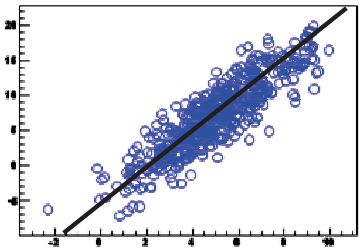
Minimizing the dispersion between the curve « y=ax+b » and the distribution :

$$w(a,b) = \iint (y-ax-b)^2 f(x,y)dxdy \left(= \frac{1}{n}\sum_i (y_i - ax_i - b)^2 \right)$$

$$\begin{cases} \frac{\partial w}{\partial a} = 0 = \iint x(y-ax-b)f(x,y)dxdy \\ \frac{\partial w}{\partial b} = 0 = \iint (y-ax-b)f(x,y)dxdy \end{cases}$$

$$\Leftrightarrow \begin{cases} a(\sigma_X^2 - \mu_X^2) + b\mu_X = \rho\sigma_X\sigma_Y + \mu_X\mu_Y \\ a\mu_X + b = \mu_Y \end{cases}$$

$$\Leftrightarrow \begin{cases} a = \rho\frac{\sigma_Y}{\sigma_X} \\ b = \mu_Y - \rho\frac{\sigma_Y}{\sigma_X}\mu_X \end{cases}$$



Fully correlated ρ=1
Fully anti-correlated ρ=-1
Then Y = aX+b

Moments

For any function **g(x)**, the **expectation** of **g** is :

$$E[g(X)] = \int g(x)f(x)dx$$ It's the mean value of g

Moments μ_k are the expectation of X^k.

0^{th} moment : $\mu_0 = 1$ (pdf normalization)

1^{st} moment : $\mu_1 = \mu$ (mean)

X' = X-μ_1 is a **central variable**

2^{nd} central moment : $\mu'_2 = \sigma^2$ (variance)

Characteristic function $\varphi(t) = E[e^{ixt}] = \int f(x)e^{ixt}dx = FT^{-1}[f]$

From Taylor expansion : $\varphi(t) = \int \sum_k \frac{(itx)^k}{k!} f(x)dx = \sum_k \frac{(it)^k}{k!}\mu_k$

$$\mu_k = -i^k \left.\frac{d^k\varphi}{dt^k}\right|_{t=0}$$

Pdf entirely defined by its moments
CF : useful tool for demonstrations

Skewness and kurtosis

Reduced variable : $X'' = (X-\mu)/\sigma = X'/\sigma$

Measure of asymmetry :

3rd reduced moment : $\mu''_3 = \sqrt{\beta_1} = \gamma_1$: **skewness**

$\gamma_1=0$ for symmetric distribution. Then **mean = median**

Measure of shape :

4th reduced moment : $\mu''_4 = \beta_2 = \gamma_2 + 3$: **kurtosis**

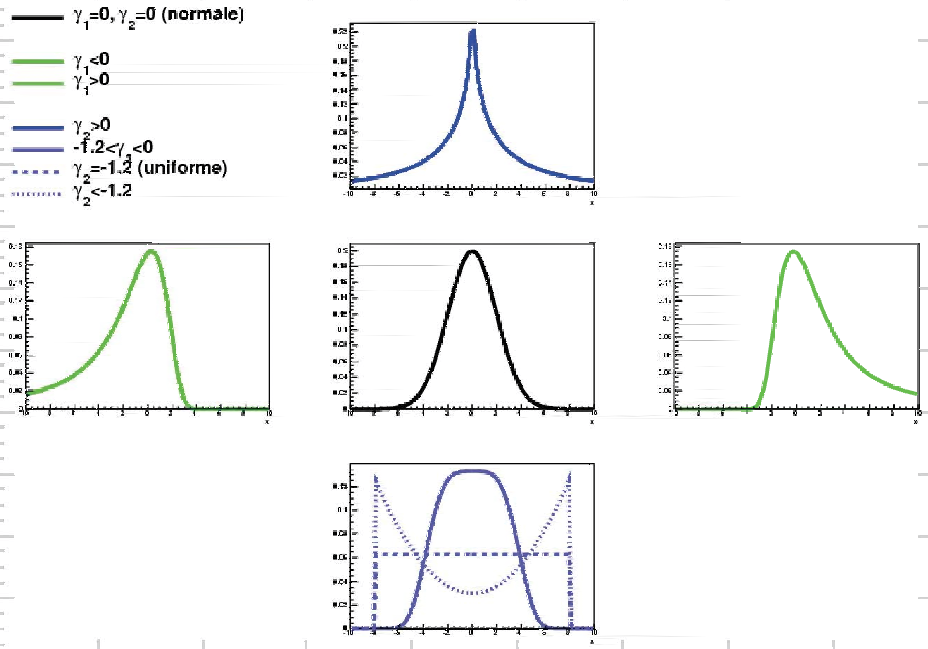
For the normal distribution $\beta_2 = 3$ and $\gamma_2 = 0$

Generalized Koenig's theorem

$$\mu'_n = (-1)^n(1-n)\mu_1^n + \sum_{k=2}^{n} \frac{n!}{k!(n-k)!}(-\mu_1)^{n-k}\mu_k$$

$$\mu''_n = \left(\frac{1}{\mu'_2}\right)^{n-2} \mu'_n$$

Skewness and kurtosis (2)



Discrete distributions

Binomial distribution: randomly choosing **K** objects within a finite set of **n**, with a fixed drawing probability of **p**

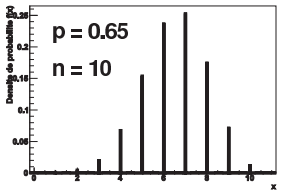
Variable : K

Parameters : n, p

Law : $P(k;n,p) = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}$

Mean : np

Variance : $np(1-p)$



Poisson distribution : limit of the binomial when $n \to +\infty, p \to 0, np = \lambda$

Counting events with fixed probability per time/space unit.

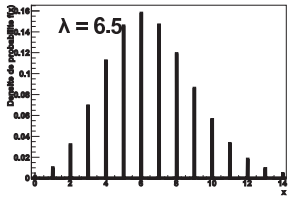
Variable : K

Parameters : λ

Law : $P(k;\lambda) = \frac{e^{-\lambda}\lambda^k}{k!}$

Mean : λ

Variance : λ



Real distributions

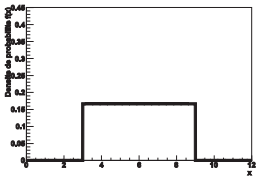
Uniform distribution : equiprobability over a finite range [a,b]

Parameters : a, b

Law : $f(x;a,b) = \frac{1}{b-a}$ if $a < x < b$

Mean : $\mu = (a+b)/2$

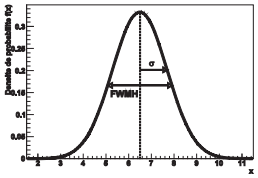
Variance : $v = \sigma^2 = (b-a)^2/12$



Normal distribution (Gaussian) : limit of many processes

Parameters : μ, σ

Law : $f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$

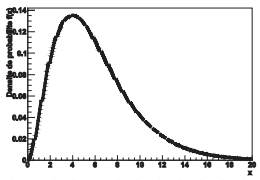


Chi-square distribution : sum of the square of **n** normal reduced variables

Variable : $C = \sum_{k=1}^{n} \left(\frac{X_k - \mu_{X_k}}{\sigma_{X_k}} \right)^2$

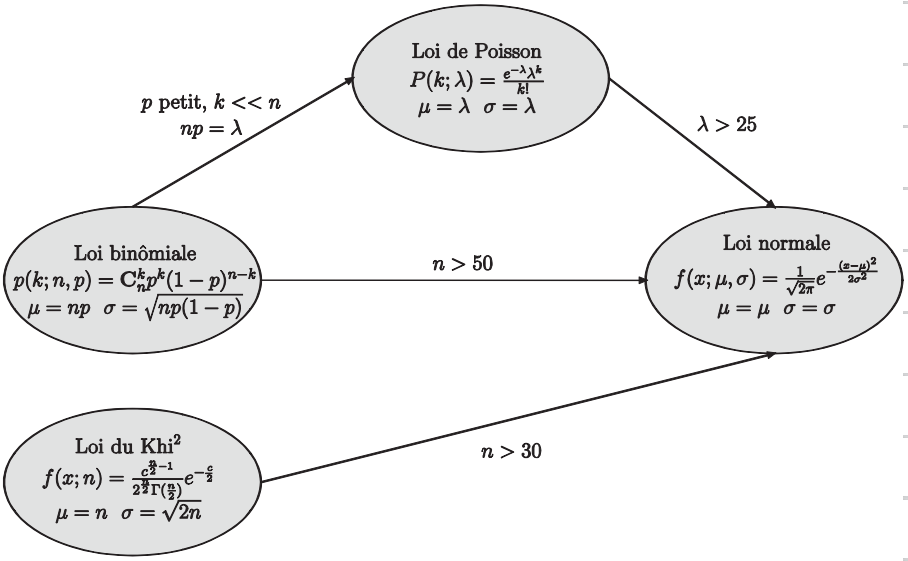
Parameters : n

Law : $f(c;n) = c^{\frac{n}{2}-1} e^{\frac{-c}{2}} \Big/ 2^{\frac{n}{2}-1} \Gamma\left(\frac{n}{2}\right)$



Mean : n Variance : $2n$

Convergence



Multidimensional Pdfs

Multinomial distribution : randomly choosing K_1 , K_2 ,... K_s objects within a finite set of n, with a fixed drawing probability for each category p_1, p_2,... p_s with $\Sigma K_i=n$ and $\Sigma p_i=1$

Parameters : $n,\ p_1,\ p_2,\ldots\ p_s$

Law : $P(\vec{k};n,\vec{p})=\dfrac{n!}{k_1!k_2!\ldots k_s!}p_1^{k_1}p_2^{k_2}\ldots p_s^{k_s}$

Mean : $\mu_i=np_i$

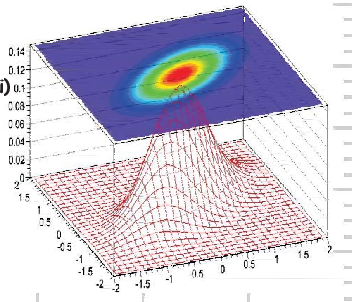
Variance : $\sigma_i^2=np_i(1-p_i)$ $\quad Cov(K_i,K_j)=-np_ip_j$

Rem : **variables are not independent**. The binomial, correspond to s=2, but has only one independent variable.

Multinormal distribution :

Parameters : $\vec{\mu},\Sigma$

Law : $f(\vec{x};\vec{\mu},\Sigma)=\dfrac{1}{\sqrt{2\pi|\Sigma|}}e^{-\frac{1}{2}(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})}$



if **uncorrelated** $f(\vec{x};\vec{\mu},\Sigma)=\prod\dfrac{1}{\sigma_i\sqrt{2\pi}}e^{\frac{(x_i-\mu_i)^2}{2\sigma_i^2}}$

Independent ⇔ **Uncorrelated**

Sum of random variables

The sum of several random variable is a new random variable S

$$S = \sum_{i=1}^{n} X_i$$

Assuming the mean and variance of each variable exists,

Mean value of S :

$$\mu_S = \int \left(\sum_{i=1}^{n} x_i \right) f(x_1, \ldots, x_n) dx_1 \ldots dx_n = \sum_{i=1}^{n} \int x_i f_{X_i}(x_i) dx_i = \sum_{i=1}^{n} \mu_i$$

The mean is an additive quantity

Variance of S :

$$\sigma_S^2 = \int \left(\sum_{i=1}^{n} x_i - \mu_{X_i} \right)^2 f(x_1, \ldots, x_n) dx_1 \ldots dx_n$$

$$= \sum_{i=1}^{n} \sigma_{X_i}^2 + 2 \sum_i \sum_{j<i} Cov(X_i, X_j)$$

For uncorrelated variables, $\sigma_S^2 = \sum_{k=1}^{n} \sigma_{X_i}^2$

the variance is additive -> used for error combinations

Sum of random variables

Probability density function of S : $f_S(s)$

Using the characteristic function :

$$\varphi_S(t) = \int f_S(s) e^{ist} ds = \int f_{\vec{X}}(\vec{x}) e^{it \sum x_i} d\vec{x}$$

For **independent variables**

$$\varphi_S(t) = \prod \int f_{X_k}(x_k) e^{itx_k} dx_k = \prod \varphi_{X_i}(t)$$

The characteristic function factorizes.

Finally the pdf is the **Fourier transform** of the cf, so :

$$f_S = f_{X_1} * f_{X_2} * \ldots * f_{X_n}$$

The pdfs of the sum is a **convolution**.

Sum of Normal variables -> Normal

Sum of Poisson variables (λ_1 and λ_2) -> Poisson, $\lambda = \lambda_1 + \lambda_2$

Sum of Khi-2 variables (n_1 and n_2) -> Khi-2, $n = n_1 + n_2$

Sum of independent variables

Weak law of large numbers

Sample of size **n** = realization of **n** independent variables, with the **same distribution** (mean **μ**, variance **σ²**).

The **sample mean** is a realization of $M = \frac{S}{n} = \frac{1}{n}\sum X_i$

Mean value of **M** : $\mu_M = \mu$ **Variance** of **M** : $\sigma_M^2 = \sigma^2/n$

From Bienaymé-Chebyshev : $P(|M-\mu| > a) < \frac{\sigma^2}{na^2} \xrightarrow[n\to+\infty]{} 0 \ (\forall a)$

Central-Limit theorem

n independent random variables of mean μ_i and variance σ_i^2

Sum of the reduced variables : $C = \frac{1}{\sqrt{n}}\sum \frac{X_i - \mu_i}{\sigma_i}$

The pdfs of **C** converge to a reduced normal distribution :

$$f_C(c) \xrightarrow[n\to+\infty]{} \frac{1}{\sqrt{2\pi}} e^{-\frac{c^2}{2}}$$

Central limit theorem

Naive demonstration:

For each X_i : X''_i has mean 0 and variance 1. So its characteristic function is :

$$\varphi_{X_i''}(t) = 1 - \frac{t^2}{2} + o(t^2)$$

Hence the characteristic function of **C** :

$$\varphi_C(t) = \varphi_{X_i''}\left(\frac{t}{\sqrt{n}}\right)^n = \left(1 - \frac{t^2}{2n} + o\left(\frac{t^2}{n}\right)\right)^n$$

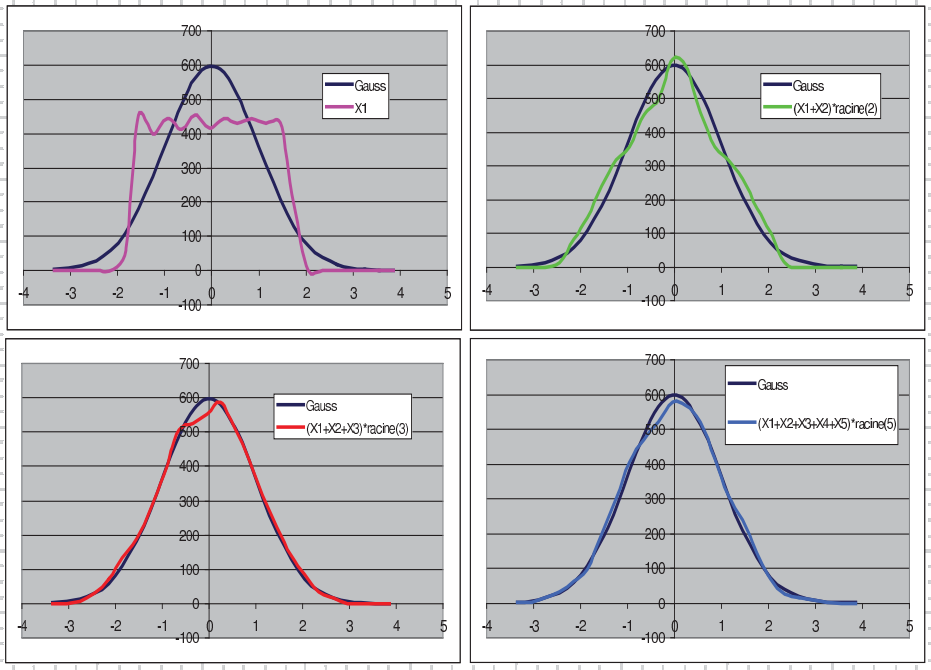
For **n** large :

$$\lim_{n\to+\infty} \varphi_C(t) = \lim_{n\to+\infty}\left(1 - \frac{t^2}{2n}\right)^n = e^{-\frac{t^2}{2}} = FT^{-1}[f_C]$$

This is a naive demonstration, because we assumed that the moments were defined.

For CLT, only mean and variance are required (much more complex)

Central limit theorem



Dispersion and uncertainty

Any measure (or combination of measure) is a realization of a random variable.

- Measured value : θ
- True value : θ_0

Uncertainty = quantifying the difference between θ and θ_0 :

->measure of dispersion

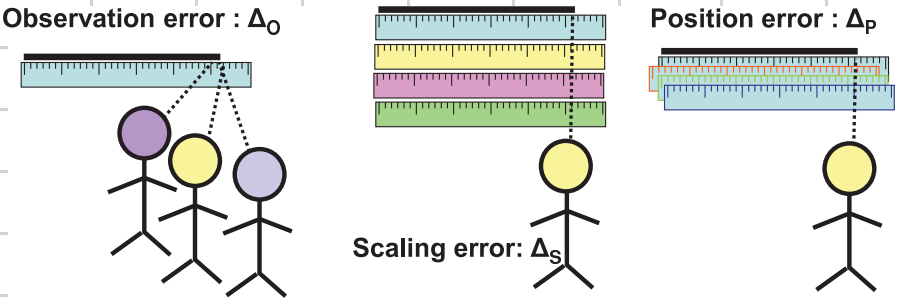
We will postulate : $\Delta\theta = \alpha\sigma_\theta$ Absolute error, always positive

Usually one separate

- **Statistical error** : due to the measurement Pdf.
- **Systematic errors** or bias -> fixed but unknown deviation (equipment, assumptions,…)

Systematic errors can be seen as statistical error in a set a similar experiences.

Error sources



$$\theta = \theta_0 + \delta_O + \delta_S + \delta_P$$

Each δ_i is a realization of a random variable : mean 0 (negligible) and variance σ_i^2. For **uncorrelated error sources** :

$$\left.\begin{aligned}\Delta_O &= a\sigma_O \\ \Delta_S &= a\sigma_S \\ \Delta_P &= a\sigma_P\end{aligned}\right\} \Delta_{tot}^2 = (a\,\sigma_{tot})^2 = a^2(\sigma_O^2 + \sigma_S^2 + \sigma_P^2) = \Delta_O^2 + \Delta_S^2 + \Delta_P^2$$

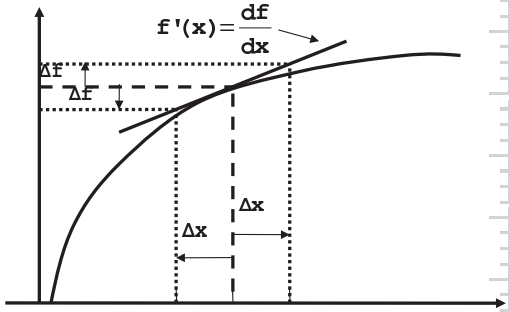
Choice of a ?

If many sources, from central-limit -> normal distribution
a=1 gives (approximately) a 68% confidence interval
a=2 gives 95% CL (and at least 75% from Bienaymé-Chebyshev)

Error propagation

Measure : $x \pm \Delta x$

Compute : f(x) -> Δf ?



Assuming **small errors**, using Taylor expansion :

$$f(x+\Delta x) = f(x) + \frac{df}{dx}\Delta x + \frac{1}{2}\frac{d^2f}{dx^2}\Delta x^2 \left(+ \frac{1}{6}\frac{d^3f}{dx^3}\Delta x^3 + \frac{1}{24}\frac{d^4f}{dx^4}\Delta x^4 \right)$$

$$f(x-\Delta x) = f(x) - \frac{df}{dx}\Delta x + \frac{1}{2}\frac{d^2f}{dx^2}\Delta x^2 \left(- \frac{1}{6}\frac{d^3f}{dx^3}\Delta x^3 + \frac{1}{24}\frac{d^4f}{dx^4}\Delta x^4 \right)$$

$$\Rightarrow \Delta f = \frac{1}{2}\left| f(x+\Delta x) - f(x-\Delta x) \right| = \frac{df}{dx}\Delta x \left(+ \frac{1}{6}\frac{d^3f}{dx^3}\Delta x^3 \right)$$

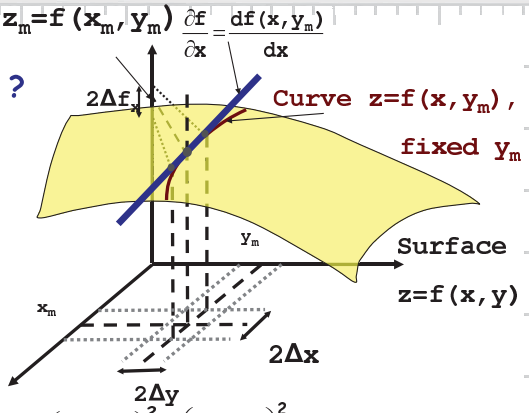
Error propagation

Measure : $x \pm \Delta x,\ y \pm \Delta y, \ldots$

Compute : $f(x,y,\ldots) \rightarrow \Delta f$?

Idea : treat the effect of each variable as separate **error sources**

$$\Delta_x f = \left|\frac{\partial f}{\partial x}\right| \Delta x \ , \ \Delta_y f = \left|\frac{\partial f}{\partial y}\right| \Delta y$$



Then

$$\Delta f^2 = \Delta_x f^2 + \Delta_y f^2 + \rho_{xy}\Delta_x f \Delta_y f = \left(\frac{\partial f}{\partial x}\Delta x\right)^2 + \left(\frac{\partial f}{\partial y}\Delta y\right)^2 + \rho_{xy}\frac{\partial f}{\partial x}\frac{\partial f}{\partial y}\Delta x \Delta y$$

$$\Delta f^2 = \sum_i \left(\frac{\partial f}{\partial x_i}\Delta x_i\right)^2 + \sum_{i,j<i} \rho_{x_i x_j} \left|\frac{\partial f}{\partial x_i}\frac{\partial f}{\partial x_j}\right| \Delta x_i \Delta x_j$$

uncorrelated	correlated	anticorrelated
$\Delta f^2 = \sum_i \left(\frac{\partial f}{\partial x_i}\Delta x_i\right)^2$	$\Delta f = \left\|\frac{\partial f}{\partial x}\right\| \Delta x + \left\|\frac{\partial f}{\partial y}\right\| \Delta y$	$\Delta f = \left\|\left\|\frac{\partial f}{\partial x}\right\| \Delta x - \left\|\frac{\partial f}{\partial y}\right\| \Delta y\right\|$

Probability and Statistics

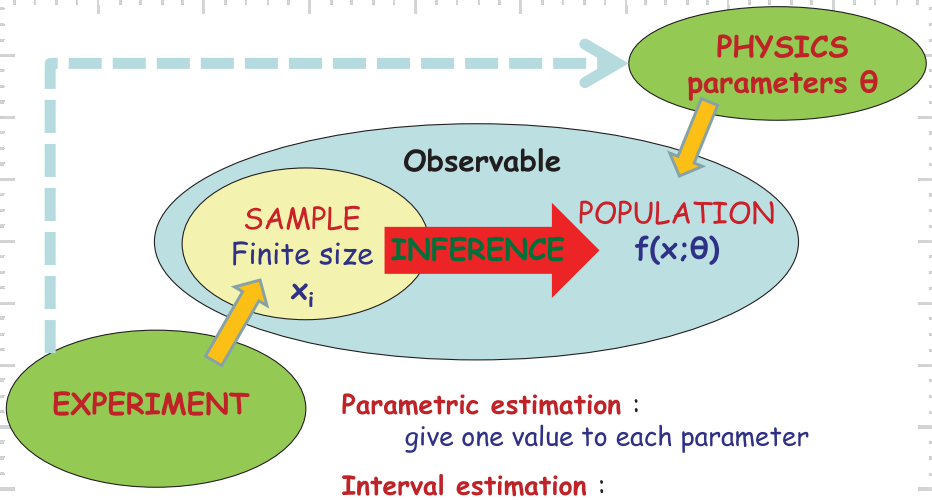
Basic concepts II

(from a physicist point of view)

Benoit CLEMENT – Université J. Fourier / LPSC

bclement@lpsc.in2p3.fr

Statistics



Parametric estimation :
give one value to each parameter

Interval estimation :
derive a interval that probably contains the true value

Non-parametric estimation :
estimate the full pdf of the population

Parametric estimation

From a finite sample $\{x_i\}$ -> estimating a parameter θ

Statistic = a function $S = f(\{x_i\})$

Any statistic can be considered as an **estimator** of θ

To be a good estimator it needs to satisfy :

- **Consistency** : limit of the estimator for a infinite sample.
- **Bias** : difference between the estimator and the true value
- **Efficiency** : speed of convergence
- **Robustness** : sensitivity to statistical fluctuations

A **good estimator** should at least be **consistent** and **asymptotically unbiased**

Efficient / Unbiased / Robust often contradict each other

⇒**different choices for different applications**

Bias and consistency

As the sample is a set of realization of random variables (or one vector variable), so is the estimator :

$\hat{\theta}$ **is a realization of** $\hat{\Theta}$

it has a mean, a variance,... and a probability density function

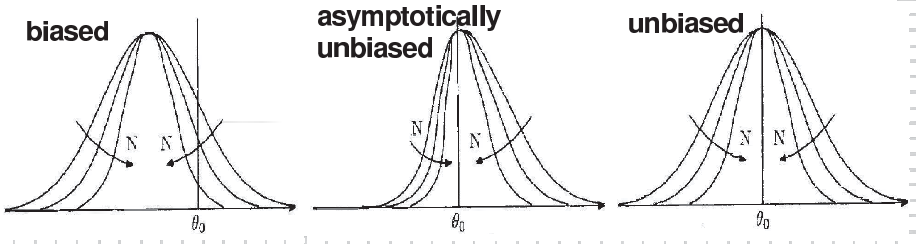
Bias : Mean value of the estimator $b(\hat{\theta}) = E[\hat{\Theta} - \theta_0] = \mu_{\hat{\Theta}} - \theta_0$

unbiased estimator : $b(\hat{\theta}) = 0$

asymptotically unbiased : $b(\hat{\theta}) \xrightarrow[n\to+\infty]{} 0$

Consistency: formally $P(|\hat{\theta} - \theta| > \varepsilon) \xrightarrow[n\to+\infty]{} 0, \forall\varepsilon$

in practice, if **asymptotically unbiased** $\sigma_{\hat{\Theta}} \xrightarrow[n\to+\infty]{} 0$



Empirical estimator

Sample mean is a good estimator of the **population mean**

-> **weak law of large numbers** : convergent, unbiased

$$\hat{\mu} = \frac{1}{n}\sum x_i, \qquad \mu_{\hat{\mu}} = E[\hat{\mu}] = \mu, \qquad \sigma^2_{\hat{\mu}} = E[(\hat{\mu}-\mu)^2] = \frac{\sigma^2}{n}$$

Sample variance as an estimator of the population variance :

$$\hat{s}^2 = \frac{1}{n}\sum_i (x_i - \hat{\mu})^2 = \left(\frac{1}{n}\sum_i (x_i-\mu)^2\right) - (\mu-\hat{\mu})^2$$

$$E[\hat{s}^2] = \left(\frac{1}{n}\sum_i \sigma^2\right) - \sigma^2_{\hat{\mu}} = \sigma^2 - \frac{\sigma^2}{n} = \frac{n-1}{n}\sigma^2$$

biased, asymptotically unbiased

unbiased variance estimator : $\hat{\sigma}^2 = \frac{1}{n-1}\sum_i (x_i - \hat{\mu})^2$

variance of the estimator (convergence) $\sigma^2_{\hat{\sigma}^2} = \frac{\sigma^4}{n-1}\left(\frac{n-1}{n}\gamma_2 + 2\right) \to \frac{2\sigma^4}{n}$

Errors on these estimator

Uncertainty ⇔ Estimator standard deviation

Use an estimator of standard deviation : $\hat{\sigma} = \sqrt{\hat{\sigma}^2}$ (!!! Biased)

Mean : $\hat{\mu} = \frac{1}{n}\sum x_i, \qquad \sigma^2_{\hat{\mu}} = \frac{\sigma^2}{n} \Rightarrow \boxed{\Delta\hat{\mu} = \sqrt{\frac{\hat{\sigma}^2}{n}}}$

Variance : $\hat{\sigma}^2 = \frac{1}{n-1}\sum_i (x_i - \hat{\mu})^2, \ \sigma^2_{\hat{\sigma}^2} \approx \frac{2\sigma^4}{n} \Rightarrow \boxed{\Delta\hat{\sigma}^2 = \sqrt{\frac{2}{n}}\hat{\sigma}^2}$

Central-Limit theorem -> empirical estimators of mean and variance are **normally distributed**, for **large enough samples**

$\hat{\mu} \pm \Delta\hat{\mu}$; $\hat{\sigma}^2 \pm \Delta\hat{\sigma}^2$ define 68% confidence intervals

Likelihood function

Generic function $k(x,\theta)$

x : random variable(s)

θ : parameter(s)

fix $\theta=\theta_0$ (true value) → **Probability density function**

$$f(x;\theta) = k(x,\theta_0)$$

$$\int f(x;\theta)\,dx = 1$$

for Bayesian $f(x|\theta) = f(x;\theta)$

fix $x=u$ (one realization of the random variable) → **Likelihood function**

$$\mathcal{L}(\theta) = k(u,\theta)$$

$$\int \mathcal{L}(\theta)\,d\theta = ???$$

for Bayesian $f(\theta|x) = \mathcal{L}(\theta)/\int \mathcal{L}(\theta)d\theta$

For a **sample** : **n** independent realizations of the same variable **X**

$$\mathcal{L}(\theta) = \prod_i k(x_i,\theta) = \prod_i f(x_i;\theta)$$

Estimator variance

Start from the generic k function, differentiate twice, with respect to θ, the pdf normalization condition: $1 = \int k(x,\theta)dx$

$$0 = \int \frac{\partial k}{\partial \theta}dx = \int k\frac{\partial \ln k}{\partial \theta}dx = E\left[\frac{\partial \ln k}{\partial \theta}\right] \Rightarrow (b+\theta)E\left[\frac{\partial \ln k}{\partial \theta}\right] = 0$$

$$0 = \int \frac{\partial^2 k}{\partial \theta^2}dx = \int k\frac{\partial^2 \ln k}{\partial \theta^2}dx + \int k\left(\frac{\partial \ln k}{\partial \theta}\right)^2 dx \Rightarrow E\left[\frac{\partial^2 \ln k}{\partial \theta^2}\right] = -E\left[\left(\frac{\partial \ln k}{\partial \theta}\right)^2\right]$$

Now differentiating the estimator bias : $\theta + b = \int \hat{\theta}(x)k(x,\theta)dx$

$$1+\frac{\partial b}{\partial \theta} = \frac{\partial}{\partial \theta}\int \hat{\theta}(x)k(x,\theta)dx = \int \hat{\theta}\frac{\partial k}{\partial \theta}dx = \int \hat{\theta}k\frac{\partial \ln k}{\partial \theta}dx = \int (\hat{\theta}-b-\theta)k\frac{\partial \ln k}{\partial \theta}dx$$

Finally, using Cauchy-Schwartz inequality

$$\left(1+\frac{\partial b}{\partial \theta}\right)^2 \le \int (\hat{\theta}-b-\theta)^2 k dx \int k\left(\frac{\partial \ln k}{\partial \theta}\right)^2 dx \Rightarrow \sigma^2_{\hat{\theta}} \ge \frac{(1+b')^2}{E\left[\left(\frac{\partial \ln k}{\partial \theta}\right)^2\right]}$$

Cramer-Rao bound

Efficiency

For **any unbiased estimator of θ**, the variance cannot exceed :

$$\sigma^2_{\hat{\Theta}} \geq \frac{1}{E\left[\left(\frac{\partial \ln \mathcal{L}}{\partial \Theta}\right)^2\right]} \left(= \frac{-1}{E\left[\frac{\partial^2 \ln \mathcal{L}}{\partial \Theta^2}\right]} \right)$$

The **efficiency** of a convergent estimator, is given by its **variance**.

An **efficient estimator** reaches the Cramer-Rao bound (at least asymptotically) : Minimal variance estimator

MVE will often be biased, asymptotically unbiased

Maximum likelihood

For a sample of measurements, $\{x_i\}$

The **analytical form of the density is known**

It depends on **several unknown parameters θ**

eg. event counting : Follow a Poisson distribution, with a parameter that depends on the physics : $\lambda_i(\theta)$

$$\mathcal{L}(\theta) = \prod_i \frac{e^{\lambda_i(\theta)} \lambda_i(\theta)^{x_i}}{x_i!}$$

An estimator of the parameters of θ, are the ones that **maximize of observing the observed result.**

-> Maximum of the likelihood function

$$\left.\frac{\partial \mathcal{L}}{\partial \Theta}\right|_{\Theta=\hat{\Theta}} = 0$$

rem : system of equations for several parameters

rem : **often minimize** $-\ln\mathcal{L}$: simplify expressions

Properties of MLE

Mostly **asymptotic properties** : valid for large sample, often assumed in any case for lack of better information

Asymptotically **unbiased**

Asymptotically **efficient** (reaches the CR bound)

Asymptotically **normally distributed**

-> Multinormal law, with covariance given by generalization of CR Bound :

$$f(\hat{\vec{\theta}};\vec{\theta},\Sigma)=\frac{1}{\sqrt{2\pi|\Sigma|}}e^{-\frac{1}{2}(\hat{\vec{\theta}}-\vec{\theta})^{T}\Sigma^{-1}(\hat{\vec{\theta}}-\vec{\theta})} \qquad \Sigma^{-1}_{ij}=-E\left[\frac{\partial \ln\mathcal{L}}{\partial\theta_i}\frac{\partial \ln\mathcal{L}}{\partial\theta_j}\right]$$

Goodness of fit = The value of $-2\ln L(\hat{\theta})$ is Khi-2 distributed, with

ndf = sample size – number of parameters

$$p-value=\int_{-2\ln\mathcal{L}(\hat{\theta})}^{+\infty} f_{\chi^2}(x;ndf)dx$$

Probability of getting a worse agreement

Errors on MLE

$$f(\hat{\vec{\theta}};\vec{\theta},\Sigma)=\frac{1}{\sqrt{2\pi|\Sigma|}}e^{-\frac{1}{2}(\hat{\vec{\theta}}-\vec{\theta})^{T}\Sigma^{-1}(\hat{\vec{\theta}}-\vec{\theta})} \qquad \Sigma^{-1}_{ij}=-E\left[\frac{\partial \ln\mathcal{L}}{\partial\theta_i}\frac{\partial \ln\mathcal{L}}{\partial\theta_j}\right]$$

Errors on parameter -> from the covariance matrix

For **one parameter**, 68% interval $\Delta\theta=\hat{\sigma}_{\hat{\theta}}=\sqrt{\frac{-1}{\frac{\partial^2 \ln\mathcal{L}}{\partial\theta^2}}}$

only one realization of the estimator -> empirical mean of 1 value…

More generally :

$$\Delta\ln\mathcal{L}=\ln\mathcal{L}(\hat{\theta})-\ln\mathcal{L}(\theta)=\frac{1}{2}\sum_{i,j}\Sigma^{-1}_{ij}(\theta_i-\hat{\theta}_i)(\theta_j-\hat{\theta}_j)+O(\theta^3)$$

Confidence contour are defined by the equation : $\Delta\ln\mathcal{L}=\beta(n_\theta,\alpha)$ with $\alpha=\int_0^{2\beta} f_{\chi^2}(x;n_\theta)dx$

Values of β for different **number of parameters n_θ** and **confidence levels α**

$n_\theta\rightarrow$ / $\alpha\downarrow$	1 (0. 5*n_θ^2)	2	3
68.3	0.5	1.15	1.76
95.4	2	3.09	4.01
99.7	4.5	5.92	7.08

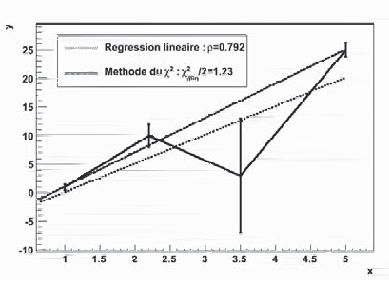
Least squares

Set of measurements (x_i, y_i) with uncertainties on y_i

Theoretical law : $y = f(x,\theta)$

Naïve approach : use **regression**

$$w(\theta) = \sum_i (y_i - f(x_i,\theta))^2, \quad \frac{\partial w}{\partial \theta_i} = 0$$



Reweight each term by the error

$$K^2(\theta) = \sum_i \left(\frac{y_i - f(x_i,\theta)}{\Delta y_i} \right)^2, \quad \frac{\partial K^2}{\partial \theta_i} = 0$$

Maximum likelihood : assume each y_i is normally distributed with a mean equal to $f(x_i,\theta)$ and a variance equal to Δy_i

Then the **likelihood** is : $\mathcal{L}(\theta) = \prod_i \frac{1}{\sqrt{2\pi}\Delta y_i} e^{-\frac{1}{2}\left(\frac{y_i - f(x_i,\theta)}{\Delta_{y_i}}\right)^2}$

$$\frac{\partial \mathcal{L}}{\partial \theta} = 0 \Leftrightarrow -2\frac{\partial \ln \mathcal{L}}{\partial \theta} = \frac{\partial K^2}{\partial \theta} = 0$$

Least squares or **Khi-2 fit** is the MLE, for Gaussian errors

Generic case with correlations: $K^2(\vec{\theta}) = \frac{1}{2}(\vec{y} - \vec{f}(x,\vec{\theta})^T \Sigma^{-1} (\vec{y} - \vec{f}(x,\vec{\theta})$

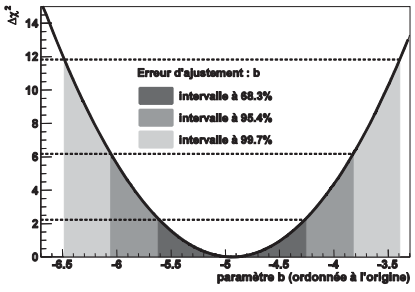
Example : fitting a line

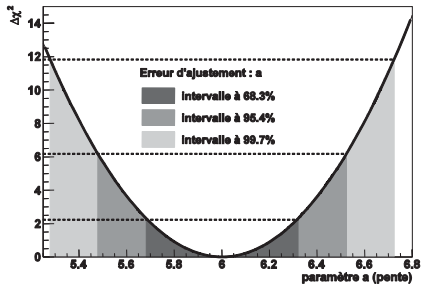
For $f(x)=ax+b$

$$A = \sum_i \frac{x_i y_i}{\Delta y_i^2}, \; B = \sum_i \frac{x_i^2}{\Delta y_i^2}, \; C = \sum_i \frac{x_i}{\Delta y_i^2}, \; D = \sum_i \frac{y_i}{\Delta y_i^2}, \; E = \sum_i \frac{1}{\Delta y_i^2}$$

$$\hat{a} = \frac{AE - DC}{BE - C^2}, \qquad \hat{b} = \frac{DB - AC}{BE - C^2}$$

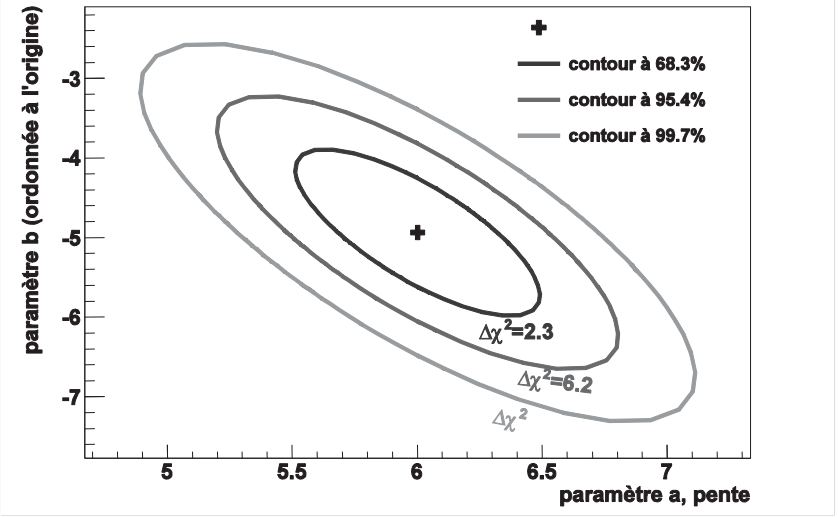
$$\Delta\hat{a} = \frac{1.52}{\sqrt{B}}, \qquad \Delta\hat{b} = \frac{1.52}{\sqrt{E}}$$





Example : fitting a line

2 dimensional error contours on **a** and **b**



Confidence interval

For a random variable, a **confidence interval** with **confidence level** **α**, is any interval **[a,b]** such as :

$$P(X \in [a,b]) = \int_a^b f_X(x)dx = \alpha$$

Probability of finding a realization inside the interval

Generalization of the concept of uncertainty:

interval that contains the **true value** with a given probability

-> slightly different concepts

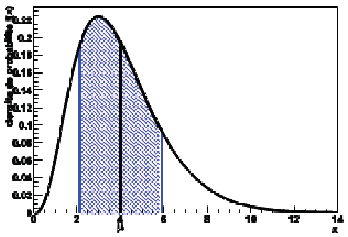
For **Bayesians** : the posterior density is the probability density of the true value. It can be used to derive interval :

$$P(\theta \in [a,b]) = \alpha$$

No such thing for a **Frequentist** : The interval itself becomes the random variable **[a,b]** is a realization of **[A,B]**

$$P(A < \theta \text{ and } B > \theta) = \alpha$$ Independently of **θ**

Confidence interval



Mean centered, symetric interval

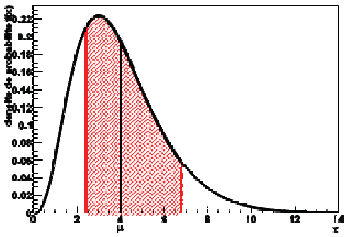
$[\mu-\alpha,\ \mu+\alpha]$

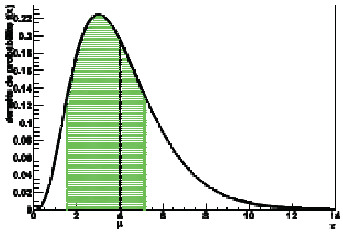
$$\int_{\mu-\alpha}^{\mu+\alpha} f(x)dx = \alpha$$

Mean centered, probability symetric interval : $[a,\ b]$

$$\int_{a}^{\mu} f(x)dx = \int_{\mu}^{b} f(x)dx = \frac{\alpha}{2}$$

,





Highest Probability Density (HDP)

$[a,\ b]\quad \int_{a}^{b} f(x)dx = \alpha$

$f(x) > f(y)$ for $x \in [a,b]$ and $y \notin [a,b]$

Confidence Belt

To build a frequentist interval for an estimator $\hat{\theta}$ of θ

1. Make pseudo-experiments for several values of θ and compute he estimator $\hat{\theta}$ for each (MC sampling of the estimator pdf)
2. For each θ, determine $A(\theta)$ and $B(\theta)$ such as :

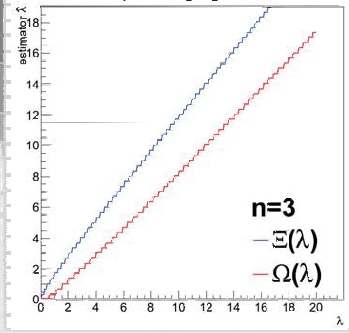
   $\hat{\theta} < \Xi(\theta)$ for a fraction $(1-\alpha)/2$ of the pseudo-experiments

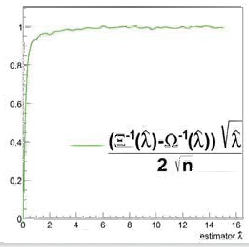
   $\hat{\theta} > \Omega(\theta)$ for a fraction $(1-\alpha)/2$ of the pseudo-experiments

   These 2 curves are the **confidence belt**, for a **CL** α.
3. Inverse these functions. The interval $[\Omega^{-1}(\hat{\theta}),\ \Xi^{-1}(\hat{\theta})]$ satisfy:

$$P\left(\Omega^{-1}(\hat{\theta}) < \theta < \Xi^{-1}(\hat{\theta})\right) = 1 - P\left(\Xi^{-1}(\hat{\theta}) < \theta\right) - P\left(\Omega^{-1}(\hat{\theta}) > \theta\right)$$

$$= 1 - P\left(\hat{\theta} < \Xi(\theta)\right) - P\left(\hat{\theta} > \Omega(\theta)\right) = \alpha$$





Confidence Belt for Poisson parameter λ estimated with the empirical mean of 3 realizations (68%CL)

Dealing with systematics

The variance of the estimator only measure the statistical uncertainty.

Often, we will have to deal with some **parameters** whose **values are known with limited precision.**

Systematic uncertainties

The likelihood function becomes :

$$\mathcal{L}(\theta, \nu) \qquad \nu = \nu_0 \pm \Delta\nu \text{ or } \nu_{0}{}^{+\Delta\nu_+}_{-\Delta\nu_-}$$

The known parameters ν are **nuisance parameters**

Bayesian inference

In **Bayesian statistics**, nuisance parameters are dealt with by assigning them a prior $\pi(\nu)$.

Usually a multinormal law is used with mean ν_0 and covariance matrix estimated from $\Delta\nu_0$ (+correlation, if needed)

$$f(\theta, \nu \mid x) = \frac{f(x \mid \theta, \nu)\pi(\theta)\pi(\nu)}{\iint f(x \mid \theta, \nu)\pi(\theta)\pi(\nu) d\theta\, d\nu}$$

The final prior is obtained by **marginalization** over the nuisance parameters

$$f(\theta \mid x) = \int f(\theta, \nu \mid x) d\nu = \frac{\int f(x \mid \theta, \nu)\pi(\theta)\pi(\nu) d\nu}{\iint f(x \mid \theta, \nu)\pi(\theta)\pi(\nu) d\theta\, d\nu}$$

Profile Likelihood

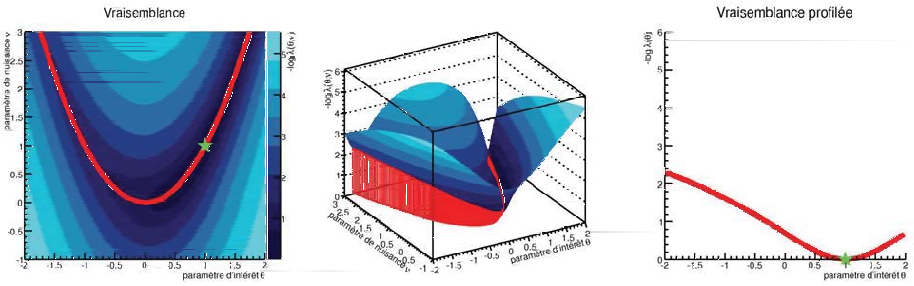
No true frequentist way to add systematic effects. Popular method of the day : **profiling**

Deal with nuisance parameters as realization if random variables : extend the likelihood : $\mathcal{L}(\theta, \nu) \rightarrow \mathcal{L}'(\theta, \nu)\mathcal{G}(\nu)$

$\mathcal{G}(\nu)$ is the **likelihood of the new parameters** (identical to prior)

For each value of θ, maximize the likelihood with respect to nuisance : **profile likelihood** $\mathcal{PL}(\theta)$.

$\mathcal{PL}(\theta)$ has the same statistical asymptotical properties than the regular likelihood



Non parametric estimation

Directly estimating the probability density function

- Likelihood ratio discriminant
- Separating power of variables
- Data/MC agreement
- ...

Frequency Table : For a sample $\{x_i\}$, $i=1..n$

1.Define successive intervals (bins) $C_k=[a_k, a_{k+1}[$

2.Count the number of events n_k in C_k

Histogram : Graphical representation of the frequency table

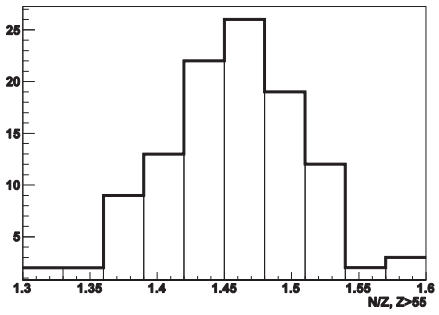
$$h(x) = n_k \text{ if } x \in C_k$$

Histogram

Classe	Nombre de N/Z	Fréquence	Classe	Nombre de N/Z	Fréquence
< 1.30	0	0	1.45 - 1.48	26	0.2363
1.30 - 1.33	2	0.0182	1.48 - 1.51	19	0.1727
1.33 - 1.36	2	0.0182	1.51 - 1.54	12	0.1091
1.36 - 1.39	9	0.0818	1.54 - 1.57	2	0.0182
1.39 - 1.42	13	0.1182	1.57 - 1.60	3	0.0273
1.42 - 1.45	22	0.2	≥ 1.60	0	0

N/Z for stable heavy nuclei

1.321, 1.357, 1.392, 1.410, 1.428, 1.446,
1.464, 1.421, 1.438, 1.344, 1.379, 1.413,
1.448, 1.389, 1.366, 1.383, 1.400, 1.416,
1.433, 1.466, 1.500, 1.322, 1.370, 1.387,
1.403, 1.419, 1.451, 1.483, 1.396, 1.428,
1.375, 1.406, 1.421, 1.437, 1.453, 1.468,
1.500, 1.446, 1.363, 1.393, 1.424, 1.439,
1.454, 1.469, 1.484, 1.462, 1.382, 1.411,
1.441, 1.455, 1.470, 1.500, 1.449, 1.400,
1.428, 1.442, 1.457, 1.471, 1.485, 1.514,
1.464, 1.478, 1.416, 1.444, 1.458, 1.472,
1.486, 1.500, 1.465, 1.479, 1.432, 1.459,
1.472, 1.486, 1.513, 1.466, 1.493, 1.421,
1.447, 1.460, 1.473, 1.486, 1.500, 1.526,
1.480, 1.506, 1.435, 1.461, 1.487, 1.500,
1.512, 1.538, 1.493, 1.450, 1.475, 1.500,
1.512, 1.525, 1.550, 1.506, 1.530, 1.487,
1.512, 1.524, 1.536, 1.518, 1.577, 1.554,
1.586, 1.586



Histogram

Statistical description : n_k are **multinomial** random variables.
with parameters :

$$n = \sum_k n_k \qquad p_k = P(x \in C_k) = \int_{C_k} f_X(x)dx$$

$$\mu_{n_k} = np_k \qquad \sigma^2_{n_k} = np_k(1-p_k) \underset{p_k<<1}{\approx} \mu_{n_k} \qquad Cov(n_k, n_r) = -np_kp_r \underset{p_k<<1}{\approx} 0$$

For a large sample :

$$\lim_{n\to+\infty} \frac{n_k}{n} = \frac{\mu_k}{n} = p_k$$

For small classes (width δ):

$$p_k = \int_{C_k} f_X(x)dx \approx \delta f(x_c) \Rightarrow \lim_{\delta\to 0} \frac{p_k}{\delta} = f(x)$$

So finally :

$$f(x) = \lim_{\substack{n\to+\infty \\ \delta\to 0}} \frac{1}{n\delta} h(x)$$

The histogram is an estimator of the probability density

Each bin can be described by a **Poisson density**.
The 1σ error on n_k is then : $\Delta n_k = \sqrt{\hat{\sigma}^2_{n_k}} = \sqrt{\hat{\mu}_{n_k}} = \sqrt{n_k}$

Kernel density estimators

Histogram is a step function -> sometime need smoother estimator

On possible solution : **Kernel Density Estimator**

Attribute to each point of the sample a "kernel" function **k(u)**

$$u = \frac{x - x_i}{w}, \qquad k(u) = k(-u), \qquad \int k(u)du = 1$$

Triangle kernel : $k(u) = 1 - |u|$, for $-1 < u < 1$

Parabolic kernel : $k(u) = \frac{3}{4}(1 - u^2)$, for $-1 < u < 1$

Gaussian kernel : $k(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}$

...

w = **kernel width**, similar to bin width of the histogram

The a pdf estimator is :

$$K(x) = \frac{1}{n}\sum_i k(u_i) = \frac{1}{n}\sum_i k\left(\frac{x - x_i}{w}\right)$$

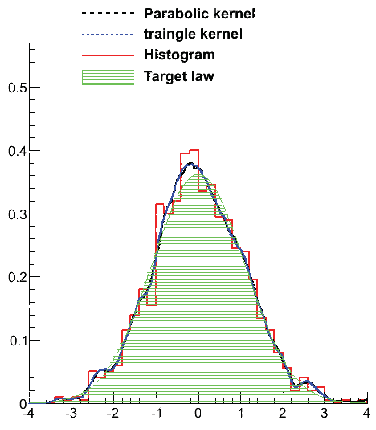
Rem : for multidimensional pdf : $u^2 = \sum_k \left(\frac{x^{(k)} - x_i^{(k)}}{w^{(k)}}\right)^2$

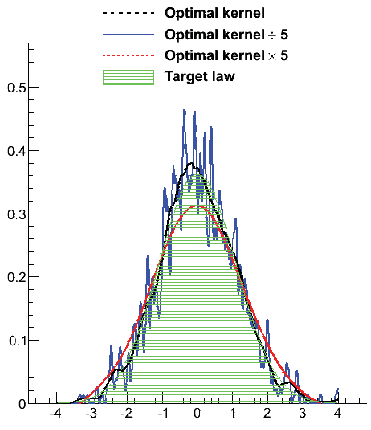
Kernel density estimators

If the estimated density is normal, the **optimal width** is :

$$w = \sigma\left(\frac{3}{(d+2)n}\right)^{\frac{1}{d+4}}$$ with **n** that sample size and **d** the dimension

As for the histogram binning, no generic result : try and see





Statistical Tests

Statistical tests aim at:

- Checking the **compatibility** of a dataset $\{x_i\}$ **with a given distribution**
- Checking the **compatibility of two datasets** $\{x_i\}$, $\{y_i\}$: are they issued from the same distribution.
- **Comparing different hypothesis** : background vs signal+background

In every case :

- build a statistic that quantify the agreement with the hypothesis
- **convert it into a probability** of compatibility/incompatibility : **p-value**

Pearson test

Test for **binned data** : use the Poisson limit of the histogram

- Sort the sample into **k** bins C_i : n_i
- Compute the probability of this class : $p_i = \int_{C_i} f(x)dx$
- The test statistics compare, for each bin **the deviation of the observation from the expected mean** to the **theoretical standard deviation.**

Data, Poisson mean

$$\chi^2 = \sum_{\text{bins } i} \frac{(n_i - np_i)^2}{np_i}$$

Poisson variance

Then χ^2 follow (asymptotically) a Khi-2 law with **k-1** degrees of freedom (1 constraint $\sum n_i = n$)

p-value : probability of doing worse, $p - \text{value} = \int_{\chi^2}^{+\infty} f_{\chi^2}(x;k-1)dx$

For a "good" agreement $\chi^2/(k-1) \sim 1$,

More precisely $\chi^2 \in (k-1) \pm \sqrt{2(k-1)}$ (1σ interval ~ 68%CL)

Kolmogorov-Smirnov test

Test for **unbinned data** : compare the sample cumulative density function to the tested one

Sample Pdf (ordered sample)

$$f_s(x) = \frac{1}{n}\sum_i \delta(x-i) \Rightarrow F_s(x) = \begin{cases} 0 & x < x_0 \\ \frac{k}{n} & x_k \le x < x_{k+1} \\ 1 & x > x_n \end{cases}$$

The the **Kolmogorov statistic** is the largest deviation :

$$D_n = \sup_x |F_S(x) - F(x)|$$

The test distribution has been computed by Kolmogorov:

$$P(D_n > \beta\sqrt{n}) = 2\sum_r (-1)^{r-1} e^{-2r^2z^2}$$

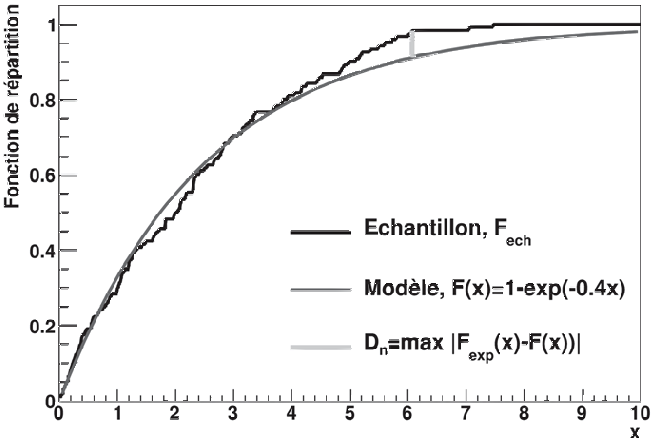
[0;β] define a confidence interval for D_n

β=0.9584/√n for 68.3% CL β=1.3754/√n for 95.4% CL

Example

Test compatibility with an **exponential law** : $f(x) = \lambda e^{-\lambda x}$, $\lambda = 0.4$

0.008, 0.036, 0.112, 0.115, 0.133, 0.178, 0.189, 0.238, 0.274, 0.323, 0.364, 0.386, 0.406, 0.409, 0.418, 0.421, 0.423, 0.455, 0.459, 0.496, 0.519, 0.522, 0.534, 0.582, 0.606, 0.624, 0.649, 0.687, 0.689, 0.764, 0.768, 0.774, 0.825, 0.843, 0.921, 0.987, 0.992, 1.003, 1.004, 1.015, 1.034, 1.064, 1.112, 1.159, 1.163, 1.208, 1.253, 1.287, 1.317, 1.320, 1.333, 1.412, 1.421, 1.438, 1.574, 1.719, 1.769, 1.830, 1.853, 1.930, 2.041, 2.053, 2.119, 2.146, 2.167, 2.237, 2.243, 2.249, 2.318, 2.325, 2.349, 2.372, 2.465, 2.497, 2.553, 2.562, 2.616, 2.739, 2.851, 3.029, 3.327, 3.335, 3.390, 3.447, 3.473, 3.568, 3.627, 3.718, 3.720, 3.814, 3.854, 3.929, 4.038, 4.065, 4.089, 4.177, 4.357, 4.403, 4.514, 4.771, 4.809, 4.827, 5.086, 5.191, 5.928, 5.952, 5.968, 6.222, 6.556, 6.670, 7.673, 8.071, 8.165, 8.181, 8.383, 8.557, 8.606, 9.032, 10.482, 14.174



D_n = 0.069

p-value = 0.0617

1σ : [0, 0.0875]

Hypothesis testing

Two exclusive hypotheses H_0 and H_1

-which one is the most compatible with data

- how incompatible is the other one

P(data|H0) vs P(data|H1)

Build a **statistic**, define an interval **w**

- if the observation falls into **w** : accept H_1
- else accept H_0

Size of the test : how often did you get it right

$$\alpha = \int_w \mathcal{L}(x|H_0)dx$$

Power of the test : how often do you get it wrong !

$$1-\beta = \int_w \mathcal{L}(x|H_1)dx$$

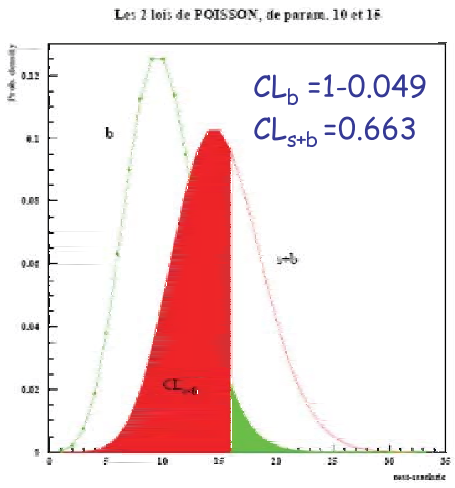
Neyman-Pearson lemma : optimal statistic for testing hypothesis is the **Likelihood ratio** $\lambda = \frac{\mathcal{L}(x|H_0)}{\mathcal{L}(x|H_1)} < k_\alpha$

CL_b and CL_s

Two hypothesis, for counting experiment

- background only : expect 10 events
- signal+ background : expect 15 events

You observe 16 events



CL_b = confidence in the background hypothesis (power of the test)

Discovery : $1 - CL_b < 5.7\times10^{-7}$

CL_{s+b} = confidence in the signal+background hypothesis (size of the test)

Rejection : $CL_{s+b} < 5\times10^{-2}$

Test for signal (non standard)

$CL_s = CL_{s+b}/CL_b$

Dealing with systematics and setting limits

D. S. Sivia
St. John's College, Oxford, UK

Abstract

The analysis of data in particle physics often involves systematic uncertainties, and the results entail the setting of limits on inferred parameters. This paper illustrates the former with a simple example, building on the introduction to the Bayesian approach given in Sivia [1], and offers a novel suggestion for the latter.

2.1 Introduction

The training in data analysis that most of us are given as undergraduates consists of being taught a collection of disjointed statistical recipes. This is generally unsatisfactory because the prescriptions appear *ad hoc* by lacking a unifying rationale. While the various tests might individually seem sensible at an intuitive level, the underlying assumptions and approximations are not obvious. It is far from clear, therefore, exactly what question is being addressed by their use.

Although attempts to give guidelines on 'best practice' are laudable, the shortcomings above will not be remedied without a programme of education on the fundamental principles of data analysis. To this end, scientists and engineers are increasingly finding that the Bayesian approach to probability theory advocated by mathematical physicists such as Laplace [2], Jeffreys [3] and Jaynes [4] provides the most suitable framework. This viewpoint is outlined in Section 2, an instructive example on dealing with systematic uncertainties given in Section 3 and a novel suggestion for setting limits offered in Section 4; we conclude with Section 5.

2.2 Bayesian Probability Theory

The origins of the Bayesian approach to probability theory dates back over three hundred years, to people such as the Bernoullis, Bayes and Laplace, and was developed as a tool for reasoning in situations where it is not possible to argue with certainty. This subject is relevant to all of us because it pertains to what we have to do everyday of our lives, both professionally and generally: namely, make inferences based on incomplete and/or unreliable data. In this context, a probability is seen as representing a *degree of belief*, or a *state of knowledge*, about something given the information available. For example, the probability of rain in the afternoon, given that there are dark clouds in the morning, is denoted by a number between zero and one, where the two extremes correspond to certainty about the outcome. Since the assessment of rain could easily be very different with additional access to the current weather maps, it means that probabilities are always *conditional* and that the associated information, assumptions and approximations must be stated clearly.

2.2.1 Manipulating Probabilities

In addition to the convention that probabilities should lie between 0 and 1, there are just two basic rules that they must satisfy:

$$\text{prob}(X|I) + \text{prob}\big(\overline{X}|I\big) = 1\,, \tag{2.1}$$

$$\text{prob}(X,Y|I) = \text{prob}(X|Y,I) \times \text{prob}(Y|I)\,. \tag{2.2}$$

Here X and Y are two specific propositions, $\overline{X}$ denotes that X is false, the vertical bar '$|$' means 'given' (so that all items to the right of this conditioning symbol are taken as being true) and the comma is read as the conjunction 'and'; I subsumes all the pertinent background information, assumptions and approximations. Equations (2.1) and (2.2), known as the *sum* and *product* rule respectively, are the same as those found in orthodox or conventional statistics; this later school of thought differs from the Bayesian one in its interpretation of probability, restricting it to apply only to frequencies, which limits its sphere of direct application.

Many other relationships can be derived from Eqs. (2.1) and (2.2). Among the most useful are:

$$\text{prob}(X|Y,I) = \frac{\text{prob}(Y|X,I) \times \text{prob}(X|I)}{\text{prob}(Y|I)}\,, \tag{2.3}$$

$$\text{prob}(X|I) = \text{prob}(X,Y|I) + \text{prob}\big(X,\overline{Y}|I\big)\,. \tag{2.4}$$

Equation (2.3) is called *Bayes' theorem*. Its power lies in the fact that it turns things around with respect to the conditioning symbol: it relates $\text{prob}(X|Y,I)$ to $\text{prob}(Y|X,I)$. Equation (2.4) is the simplest form of *marginalisation*. Its generalisations provide procedures for dealing with *nuisance* parameters and hypothesis uncertainties.

2.2.2 Assigning Probabilities

While Eqs. (2.1) and (2.2), and their corollaries, specify how probabilities are to be manipulated, the rules for their assignment are less well defined. This is inevitable to some extent as 'states of knowledge' can take a myriad different forms, often rather vague. Nevertheless, there are some simple but powerful ideas on the issue based on arguments of self-consistency: if two people have the same information then they should assign the same probability. We refer the reader to some recent textbooks for a good discussion of this topic, and for examples of Bayesian analyses in general: Jaynes [4], Sivia [1], MacKay [5] and Gregory [6].

2.3 Dealing with systematic uncertainties

Scientists often talk about statistical and systematic errors, and accuracy versus precision. From our point-of-view, however, these distinctions seem unnecessary. While the use of the term 'systematic' suggests that it's a type of error that could be reduced through calibration, it is simply a source of uncertainty that has be taken into account just like any other. Rather than discussing it in the abstract, let's consider a specific example.

Suppose we have N measurements of a quantity μ, $\{d_k\}$; what can be said about the value of μ? The information given is insufficient to address the problem unambiguously and so, as usual, the answer will depend on both the data and the assumptions made in the analysis. With the latter denoted by I, as earlier, the inference is encapsulated by $\text{prob}(\mu|\{d_k\},I)$, or the *posterior* probability distribution function (PDF) for μ.

2.3.1 The simplest case

Let us begin by making some common assumptions: suppose that the measurements are subject to *independent*, additive, *Gaussian* noise of constant magnitude σ. Then, following Section 2.3 of Sivia [1], the chance of collecting the observed data, when given the values of μ and σ, is specified by the *likelihood* function

$$\text{prob}(\{d_k\}|\mu,\sigma,I) \;=\; \prod_{k=1}^{N} \text{prob}(d_k|\mu,\sigma,I) \;=\; \left(\sigma\sqrt{2\pi}\right)^{-N} \exp\left[-\frac{Q}{2\sigma^2}\right] , \tag{2.5}$$

where Q is the sum of the squares of the mismatch between μ and the measurements:

$$Q \;=\sum_{k=1}^{N} (d_k-\mu)^2 \,. \tag{2.6}$$

For a known σ, this is related to the posterior PDF for μ through Bayes' theorem:

$$\text{prob}(\mu|\{d_k\},\sigma,I) \;\propto\; \text{prob}(\{d_k\}|\mu,\sigma,I) \times \text{prob}(\mu|\sigma,I) \,, \tag{2.7}$$

where the equality has been replaced by a proportionality due to the omission of the denominator,

$$\text{prob}(\{d_k\}|\sigma,I) \;=\; \int \text{prob}(\{d_k\}|\mu,\sigma,I)\;\text{prob}(\mu|\sigma,I)\;\mathrm{d}\mu \,, \tag{2.8}$$

which is simply a *normalisation* term. With a uniform assignment for $\text{prob}(\mu|\sigma,I)$ over a suitably large range, $\mu_{\min} \leqslant \mu \leqslant \mu_{\max}$, to express gross *prior* ignorance about the value of μ, Sivia [1] shows that the posterior PDF of Eq. (2.7) can be summarized by a Gaussian *mean* and *standard deviation*

$$\mu \;=\; \frac{1}{N}\sum_{k=1}^{N} d_k \;\pm\; \frac{\sigma}{\sqrt{N}} \,. \tag{2.9}$$

This is a familiar result, with the best estimate of μ being the average of the measurements and its reliability improving with the square-root of the number of data.

2.3.2 Unknown noise level

Since the value of σ was not specified in the statement of the problem, the posterior PDF for μ of relevance is $\text{prob}(\mu|\{d_k\},I)$ rather than the one in Eq. (2.7). Indeed, it's important to remember that even the notion of uniform noise is merely part of our simplifying assumptions. Within this context, the appropriate likelihood function for the analysis is related to that of Eqs. (2.5) and (2.6) through marginalisation and the product rule:

$$\text{prob}(\{d_k\}|\mu,I) = \int \text{prob}(\{d_k\},\sigma|\mu,I)\ \mathrm{d}\sigma = \int \text{prob}(\{d_k\}|\mu,\sigma,I)\ \text{prob}(\sigma|\mu,I)\ \mathrm{d}\sigma\,. \tag{2.10}$$

With a Jeffreys' prior for σ, $\text{prob}(\sigma|\mu,I) \propto 1/\sigma$, which is uniform with respect to $\log\sigma$ to express ignorance about the 'scale' parameter, Section 8.2 of Sivia [1] shows that Eq. (2.10) reduces to

$$\text{prob}(\{d_k\}|\mu,I) \propto Q^{-N/2}\,. \tag{2.11}$$

The resultant posterior PDF for μ,

$$\text{prob}(\mu|\{d_k\},I) \propto \text{prob}(\{d_k\}|\mu,I) \times \text{prob}(\mu|I)\ , \tag{2.12}$$

with a uniform assignment for $\text{prob}(\mu|I)$ as before, is found to be approximated well by a Gaussian for moderately large N (>10 say):

$$\mu \approx \mu_\mathrm{o} \pm \frac{S}{\sqrt{N}} \qquad \text{where} \quad \mu_\mathrm{o} = \frac{1}{N}\sum_{k=1}^{N} d_k \quad \text{and} \quad S^2 = \frac{1}{N-1}\sum_{k=1}^{N} (d_k - \mu_\mathrm{o})^2 \tag{2.13}$$

This is just like Eq. (2.9), except that the previously given value of σ has been replaced with an estimate obtained from the scatter of the data.

2.3.3 Partially known offset

Now suppose that all the measurements are subject to an offset β, known from calibration experiments to an extent characterised by $\text{prob}(\beta|I)$, so that the empirical estimates of μ are $\hat{d}_k$ where

$$\hat{d}_k = d_k - \beta\,. \tag{2.14}$$

We are still after $\text{prob}(\mu|\{d_k\},I)$, of course, but the related likelihood function entails another nuisance parameter, β, that needs to be marginalised out:

$$\text{prob}(\{d_k\}|\mu,I) = \int \text{prob}(\{d_k\},\beta|\mu,I)\ \mathrm{d}\beta = \int \text{prob}(\{d_k\}|\mu,\beta,I)\ \text{prob}(\beta|\mu,I)\ \mathrm{d}\beta\,, \tag{2.15}$$

where the likelihood of the data conditional on being given β, $\text{prob}(\{d_k\}|\mu,\beta,I)$, is the same as in Eq. (2.11) but the formula for Q is modified slightly to

$$Q = \sum_{k=1}^{N} (d_k - \beta - \mu)^2\,. \tag{2.16}$$

If β is known to be β_o to within a standard deviation accuracy of ϵ, so that

$$\text{prob}(\beta|\mu,I) = \text{prob}(\beta|I) = \left(\epsilon\sqrt{2\pi}\right)^{-1} \exp\left[-\frac{(\beta-\beta_\mathrm{o})^2}{2\epsilon^2}\right], \tag{2.17}$$

and N is sufficiently large for the Gaussian approximation of Eq. (2.13) to be valid, then the best estimate of μ and its reliability yielded by the integral of Eq. (2.15) is found to be

$$\mu \approx \mu_{\rm o} - \beta_{\rm o} \pm \sqrt{\frac{S^2}{N} + \epsilon^2} \; . \tag{2.18}$$

The calibration of the systematic offset can be considered adequate if $\epsilon^2 \ll S^2/N$, but would benefit from improvement otherwise.

2.4 Summarizing the inference

2.4.1 Gaussian-like posterior PDFs

Although the (marginal) posterior PDF for a parameter, x say, encodes our inference about its value, given the data $\{d_k\}$ and the relevant background information I, we often wish to summarize it with just two numbers: the best estimate of x, $x_{\rm o}$, and a measure of its reliability, σ_x. The usefulness of such a summary, $x = x_{\rm o} \pm \sigma_x$, relies on the posterior PDF being approximated well by a Gaussian:

$$\mathrm{prob}(x|\{d_k\},I) \approx \left(\sigma_x\sqrt{2\pi}\right)^{-1} \exp\left[-\frac{(x-x_{\rm o})^2}{2{\sigma_x}^2}\right] . \tag{2.19}$$

If that is the case, then $x_{\rm o}$ and σ_x can either be estimated from the mean and standard deviation of the posterior PDF,

$$x_{\rm o} = \int x \; \mathrm{prob}(x|\{d_k\},I) \; \mathrm{d}x \qquad \text{and} \qquad {\sigma_x}^2 = \int (x-x_{\rm o})^2 \; \mathrm{prob}(x|\{d_k\},I) \; \mathrm{d}x \; , \tag{2.20}$$

or from the location of its maximum and the curvature at that point,

$$\left.\frac{\mathrm{d}L}{\mathrm{d}x}\right|_{x_{\rm o}} = 0 \qquad \text{and} \qquad {\sigma_x}^{-2} = -\left.\frac{\mathrm{d}^2L}{\mathrm{d}x^2}\right|_{x_{\rm o}} \tag{2.21}$$

where $L = \log_{\rm e}\big[\mathrm{prob}(x|\{d_k\},I)\big]$. The two estimates will converge if Eq. (2.19) holds exactly, but should be fairly close as along as the approximation is reasonable.

2.4.2 Asymmetric posterior PDFs

The above prescription provides a poor summary of posterior PDFs that are highly skew with respect to their modes, because the concept of an error-bar implicitly entails the idea of symmetry: $x = x_{\rm o} \pm \sigma_x$. A good way of expressing the reliability with which a parameter can be inferred in such instances is through a *confidence*, or 'credible', *interval*. Since the area under the posterior PDF between x_1 and x_2 is proportional to how much we believe that x lies in that range, the *shortest* interval that encloses 95% of the area represents a sensible measure of the uncertainty of the estimate:

$$\mathrm{prob}(x_1 \leqslant x < x_2 \,|\{data\},I) = \int_{x_1}^{x_2} \mathrm{prob}(x|\{data\},I) \; \mathrm{d}x \approx 0.95 \tag{2.22}$$

where the difference $x_2 - x_1$ is as small as possible. The region $x_1 \leqslant x < x_2$ is called the shortest 95% confidence interval.

There is no reason why we cannot give the shortest 70%, 99% or any other confidence interval, of course, instead of the conventional 95% one. Indeed, there is something to be said for listing a set of nested intervals since this provides a more complete picture of the reliability analysis; by doing so, however, we are merely reconstructing the posterior PDF!

Figure 2.1: An idealised limit-type posterior PDF. (a) The upper bound is traditionally set through the use a shortest confidence interval; (b) the new suggestion is based on the value of the probability relative to its maximum.

2.4.3 Limit-type posterior PDFs

If the posterior PDF is of the form shown in the idealised Fig. 1(a), then we are only able to infer an upper bound for x, $x_{\rm b}$. Its value is often specified by using the concept of the shortest 95% confidence interval:

$$\mathrm{prob}(x \leqslant x_{\rm b}\,|\{data\},I) = \int_{x_{\rm min}}^{x_{\rm b}} \mathrm{prob}(x|\{data\},I)\;\mathrm{d}x \;\approx\; 0.95 \tag{2.23}$$

where $x_{\rm min}$ is set by the lower bound of the prior PDF for x. The problem with this prescription is that the resultant $x_{\rm b}$ depends on $x_{\rm min}$ whereas the inference about the upper bound is actually defined by the transition from high to low probability. If the posterior PDF was a step function, for example, then $x_{\rm b}$ should be at the location of the discontinuity and independent of $x_{\rm min}$; neither holds from Eq. (2.23). A better way of capturing the nature of the limit-type posterior PDF would be through the specification of $x_{\rm b}$ as the point at which the probability of x drops to a certain fraction of the maximum value:

$$\frac{\mathrm{prob}(x|\{data\},I)}{[\mathrm{prob}(x|\{data\},I)]_{\rm max}} \begin{cases} > 1-\beta & \text{for } x < x_{\rm b} \\ < 1-\beta & \text{for } x > x_{\rm b} \end{cases} \tag{2.24}$$

where $0 < \beta < 1$ defines the level of the significance. That is to say, assuming ideal monotonic decay, $x_{\rm b} \to x_{\rm min}$ as $\beta \to 0$ and $x_{\rm b} \to x_{\rm max}$ as $\beta \to 100\%$; $x_{\rm b}$ would also converge on the discontinuity of a sharp transition, for all β, as seems desirable.

2.4.4 Multimodal posterior PDFs

Depending on the nature of the experimental data to be analysed, posterior PDFs can be *multimodal*. If one of the maxima is very much larger than the others, the prescriptions of Sections 2.4.1 and 2.4.2 may be employed in the neighbourhood of the global maximum. Otherwise, the posterior PDF, and hence our inference about the parameter x, simply cannot be summarized in a meaningful way with just a couple of numbers.

2.5 Conclusions

The Bayesian viewpoint expounded here follows the approach of mathematical physicists such as Laplace[2], Jeffreys[3] and Jaynes[4], and is still not widely taught to science and engineering undergraduates today. It differs markedly in its accessibility for scientists from the works of many statisticians engaged in the Bayesian field; the latter carry over much of the vocabulary and mind-set of their classical frequentist training, which we believe to be neither necessary nor helpful. We refer the reader to some recent text-books, such as Jaynes[4], Sivia[1], MacKay[5] and Gregory[6], for a good introduction to our viewpoint.

To conclude, a black-box approach to the subject of data analysis, even with useful guidelines, is best avoided because it can be both limiting and misleading. All analyses are conditional on assumptions and approximations, and it's important to understand and state them clearly. While the evaluation of an arithmetic mean might seem objective and incontrovertible, for example, its status as a crucial number

requires some qualified justification. We believe that an understanding of the principles underlying data analysis, along the lines outlined here, is at least as important as formal prescriptions of best practice.

References

[1] D.S. Sivia, *Data Analysis — a Bayesian tutorial* (Oxford University Press, 1996); 2nd edition with J. Skilling (2006).

[2] P. S. de Laplace, *Théorie analytique des probabilités*, Courcier Imprimeur, Paris (1812).

[3] H. Jeffreys, *Theory of probability* (Clarendon Press, Oxford, 1939).

[4] E. T. Jaynes, *Probability theory: the logic of science*, edited by G. L. Bretthorst (Cambridge University Press, 2003).

[5] D. J. C. MacKay *Information Theory, Inference, and Learning Algorithms* (Cambridge University Press, 2003).

[6] P. Gregory, *Bayesian Logical Data Analysis for the Physical Sciences* (Cambridge University Press, 2005).

Part II

Multivariate analysis tool

Introduction to multivariate discrimination

Balázs Kégl
LAL, Orsay (IN2P3)

Abstract

Multivariate discrimination or classification is one of the best-studied problem in machine learning, with a plethora of well-tested and well-performing algorithms. There are also several good general textbooks [1, 2, 3, 4, 5, 6, 7, 8, 9] on the subject written to an average engineering, computer science, or statistics graduate student; most of them are also accessible for an average physics student with some background on computer science and statistics. Hence, instead of writing a generic introduction, we concentrate here on relating the subject to a practitioner experimental physicist. After a short introduction on the basic setup (Section 3.1) we delve into the practical issues of complexity regularization, model selection, and hyperparameter optimization (Section 3.2), since it is this step that makes high-complexity non-parametric fitting so different from low-dimensional parametric fitting. To emphasize that this issue is not restricted to classification, we illustrate the concept on a low-dimensional but non-parametric *regression* example (Section 3.2.1). Section 3.3 describes the common algorithmic-statistical formal framework that unifies the main families of multivariate classification algorithms. We explain here the large-margin principle that partly explains why these algorithms work. Section 3.4 is devoted to the description of the three main (families of) classification algorithms, neural networks, the support vector machine, and AdaBoost. We do not go into the algorithmic details; the goal is to give an overview on the form of the functions these methods learn and on the objective functions they optimize. Besides their technical description, we also make an attempt to put these algorithm into a socio-historical context. We then briefly describe some rather heterogeneous applications to illustrate the pattern recognition pipeline and to show how widespread the use of these methods is (Section 3.5). We conclude the chapter with three essentially open research problems that are either relevant to or even motivated by certain unorthodox applications of multivariate discrimination in experimental physics.

3.1 The supervised learning setup

The goal of supervised learning is to infer a function $g : \mathcal{X} \to \mathcal{C}$ from a data set

$$\mathcal{D} = \big\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\big\} \in (\mathcal{X} \times \mathcal{C})^n$$

comprised of pairs of *observations* $\mathbf{x}_i$ and *labels* y_i. The components $x^{(j)}$ of the d-dimensional *feature* or *observation* vector $\mathbf{x}$ are either real numbers or they come from an unordered set of cardinality $M^{(j)}$. In this latter case, without loss of generality, we will assume that $x^{(j)} \in \mathcal{I}^{(j)} = \{1, \ldots, M^{(j)}\}$.

When the label space $\mathcal{C}$ is real-valued (for example, the label represents the mass of a particle), the problem is known as *regression*, whereas when the labels come from a finite set of classes (for example, $\mathcal{C} = \{\text{signal}, \text{background}\}$), we are talking about *classification*. The quality of g on an arbitrary pair $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{C}$ is measured by an *error* or *loss* function $L\big(g, (\mathbf{x}, y)\big)$ that depends on the type of problem. In regression, the goal is to get $g(\mathbf{x})$ as close to y as possible, so the loss grows with the difference between $g(\mathbf{x})$ and y. The most widely used loss function in this case is the *quadratic* or *squared* loss

$$L_2\big(g, (\mathbf{x}, y)\big) = \big(g(\mathbf{x}) - y\big)^2.$$

In classification, typically there is no distance or similarity defined between the classes, so all we can measure is whether or not g predicts correctly the class y. The usual loss in this case is the *one-loss* or *zero-one loss*

$$L_{\mathbb{I}}\big(g, (\mathbf{x}, y)\big) = \mathbb{I}\{g(\mathbf{x}) \neq y\}, \tag{3.1}$$

where the indicator function $\mathbb{I}\{A\}$ is 1 if its argument A is true and 0 otherwise.

The goal of learning algorithms is not to *memorize* $\mathcal{D}$, rather to *generalize* from it. Indeed, it is rather trivial to construct a function g that has zero loss on every sample point $(\mathbf{x}_i, y_i)$ by explicitly setting $g(\mathbf{x}_i) \triangleq y_i$ on all sample points from $\mathcal{D}$, and, say, $g(\mathbf{x}_i) \triangleq 0$ everywhere else. We obviously have not *learned* anything from $\mathcal{D}$, we have simply memorized it. It is also clear that this function has very little chance to perform well on points not in the set $\mathcal{D}$ (unless 0 is a good prediction everywhere in which case the problem itself is trivial). To formalize this intuitive notion of generalization, it is usually assumed that all sample points (including those in $\mathcal{D}$) are independent samples from a fixed but unknown distribution $\mathfrak{D}$, and the goal is to minimize the *expected loss* (a.k.a., *risk* or *error*)

$$R(g) = \mathbb{E}_{(\mathbf{x},y)\sim\mathfrak{D}}\big\{L\big(g, (\mathbf{x}, y)\big)\big\},$$

where $\mathbb{E}_{(\mathbf{x},y)\sim\mathfrak{D}}\{L\}$ denotes the expectation of L with respect to the random variable $(\mathbf{x}, y)$ drawn from the distribution $\mathfrak{D}$. When we *know* $\mathfrak{D}$, the *optimal* function is the one that minimizes the risk, that is,

$$g^* = \arg\min_g R(g).$$

With this notation, since the expectation of the indicator of an event is the probability of the event, the misclassification probability $\mathbb{P}\{g(\mathbf{x}) \neq y\}$ is the risk generated by the one-loss

$$R_{\mathbb{I}}(g) = \mathbb{E}_{(\mathbf{x},y)\sim\mathfrak{D}}\big\{L_{\mathbb{I}}\big(g, (\mathbf{x}, y)\big)\big\} = \mathbb{P}\{g(\mathbf{x}) \neq y\},^1 \tag{3.2}$$

and so it can be shown that the optimal classifier (the so-called *Bayes classifier*) is

$$g^*_{\mathbb{I}}(\mathbf{x}) = \arg\max_y \mathbb{P}\{y \mid \mathbf{x}\}.$$

In regression, the risk is the expectation of the squared error

$$R_2(g) = \mathbb{E}_{(\mathbf{x},y)\sim\mathfrak{D}}\big\{(g(\mathbf{x}) - y)^2\big\},$$

[1] For denoting the risk generated by a particular loss, we will use the same index: the risk generated by the loss $L_\bullet$ will be denoted by $R_\bullet$.

and it can be shown easily that the optimal regressor is the conditional expectation of y given $\mathbf{x}$, that is,

$$g_2^*(\mathbf{x}) = \mathbb{E}\{y \mid \mathbf{x}\}.$$

In practice, of course, the data distribution $\mathfrak{D}$ is unknown, and the goal is to get close to the performance of g^* by only using the sample $\mathcal{D}$. As our imaginary example demonstrates it, finding a function g that minimizes the *empirical risk* (or *training error*)

$$\widehat{R}(g) = \frac{1}{n}\sum_{i=1}^{n} L\big(g, (\mathbf{x}_i, y_i)\big) \tag{3.3}$$

is not necessarily a good idea. Nevertheless, it turns out that the estimator

$$\widehat{g}^* = \arg\min_{g\in\mathcal{G}} \widehat{R}(g) \tag{3.4}$$

can have good properties both in theory and in practice if the function class $\mathcal{G}$ is not too "rich" so the functions in $\mathcal{G}$ are not too complex. In fact, the *real* error $R(\widehat{g}^*)$ of the empirically best classifier $\widehat{g}^*$ is usually underestimated by the *training* error $\widehat{R}(\widehat{g}^*)$, but the bias can be controlled, and the minimization (3.4) can lead to a good solution if the complexity of the function class $\mathcal{G}$ is "small" compared to the number of data points n.

How to control the complexity of $\mathcal{G}$ and how to find an optimal function in $\mathcal{G}$ computationally efficiently are the two main subjects of the design of supervised learning algorithms. Finding $\widehat{g}^*$ in $\mathcal{G}$ is the subject of algorithmic design. The process is called *training* or *learning*, and the data set $\mathcal{D}$ on which $\widehat{R}(\widehat{g})$ is minimized is the *training set*. Of course, once $\mathcal{D}$ is used for finding $\widehat{g}^*$, it is tainted from a statistical point of view: $\widehat{R}(\widehat{g}^*)$ is no longer an unbiased estimator of $R(\widehat{g}^*)$. *Overfitting* is the term that describes the situation when the risk $R(\widehat{g}^*)$ is significantly larger than the training error $\widehat{R}(\widehat{g}^*)$. To detect and to assess overfitting, $\widehat{g}^*$ has to be evaluated on a *test set* $\mathcal{D}'$, independent of $\mathcal{D}$.

3.1.1 Parametric fitting and the maximum likelihood method

The main subject of the chapter is non-parametric (model-less) multi-variate classification, and the setup of Section 3.1 is designed to formalize the theoretical framework for this approach. Nevertheless, it is worthy to note that "classical" low-dimensional fitting, be it maximum likelihood or least-square, also fits into this framework. These methods are treated in other chapters, so here we only sketch them to illuminate their relationship to the setup of Section 3.1.

When we know a generative probabilistic model, one of the possibilities of fitting a model is to maximize the likelihood $p(\mathbf{z})$. This can also be cast into the framework of this section with the usual loss function

$$L_P(\mathbf{z}) = -\log p(\mathbf{z}),$$

and with the risk that L_P implies

$$R_P(\mathbf{z}) = \mathbb{E}_{\mathbf{z}\sim\mathfrak{D}}\{-\log p(\mathbf{z})\}.$$

For example, when the data points $\mathbf{z}_i$ are triplets $(\mathbf{x}_i, y_i, \sigma_i)$ with the generative model of $y_i \sim \mathcal{N}\big(g(\mathbf{x}_i), \sigma_i\big)$,[2] maximum likelihood reduces to weighted least square ("chi square"), that is,

$$\arg\min_{g\in\mathcal{G}} \sum_{i=1}^{n} -\log p(\mathbf{x}_i, y_i, \sigma_i) = \arg\min_{g\in\mathcal{G}} \sum_{i=1}^{n} \frac{\big(g(\mathbf{x}_i) - y_i\big)^2}{\sigma_i^2}.$$

Although the least-square method is mostly used in parametric fitting when $\mathcal{G}$ is a low-complexity function class parametrized with a few parameters, neural networks and other model-less approaches can be used also with a (weighted) quadratic loss, and in this case all considerations about overfitting, complexity regularization, and hyperparameter optimization, usually associated with non-parametric *classification*, are also valid.

[2] $\mathcal{N}(\mu, \sigma)$ denotes the normal distribution with mean μ and standard deviation σ.

3.2 Complexity regularization (model selection) and hyperparameter optimization

The simplest way to control the complexity of the function class $\mathcal{G}$ is to fix it either explicitly (for example, by taking the set of all linear functions) or implicitly (for example, by taking the set of all functions that a neural network with N neurons and one hidden layer can represent). In *parametric* fitting, $\mathcal{G}$ is a set of functions parametrized with a low number of parameters, and the dimensionality d of the input space is usually low. Most importantly, the number of parameters is *fixed* independently of the size n of the training data set, so the overfitting issue only shows up in goodness-of-fit tests (for example, in setting the degrees of freedom of the chi-square distribution).

The situation is different in *non-parametric* fitting. Contrary to what its name suggests, non-parametric fitting means that we have a lot of parameters. This approach is used when we have little knowledge about the model that generated the observations, so we want to avoid the bias caused by fitting a misspecified rigid model. More importantly, we do not want to decide the complexity or the smoothness of the solution beforehand; we prefer that the data speak for itself.

Formally, we can define a (possibly infinite) *set* of function classes $\mathcal{G}_\theta$, parametrized by a vector θ of the so-called *hyperparameters*. Hyperparameters can take diverse forms. Most of them are related to the complexity of the class; the number of neurons in a neural network, the depth of decision trees, or the number of boosting iterations are typical examples. Other hyperparameters are penalty coefficients in a framework called *complexity regularization* [4] or *structured risk minimization* [9]. In this setup, instead of finding the empirical minimizer in $\mathcal{G}$, we *penalize* the complexity of the functions $g \in \mathcal{G}$ by an appropriate penalty $C(g)$, and find

$$\widehat{g}^*_\beta = \arg\min_{g \in \mathcal{G}} \widehat{R}(g) + \beta C(g), \tag{3.5}$$

where β is a penalty coefficient (a hyperparameter) which has to be tuned. The classical practices of *weight decay* or *early stopping* in neural networks fall in the category of structured risk minimization. Another typical example is the penalty coefficient (usually denoted by C) in support vector machines. A third family of hyperparameters are simple "technical" parameters, such as the learning rate in neural networks, or the number of features $x^{(j)}$ tested at each cut in a decision tree.

There is a vast literature on how to tune hyperparameters on the training set $\mathcal{D}$ itself. This so-called *model selection* problem has both Bayesian and frequentist solutions, but most of the time, at least in model-less non-parametric fitting, optimality results are only valid asymptotically (as training sample size $n \to \infty$), and so they only provide some ballpark hints in practice. The solution adopted in practice is called *validation*. In the simplest setup, we choose $\widehat{g}^*_\theta$ by minimizing the training error in each set $\mathcal{G}_\theta$

$$\widehat{g}^*_\theta = \arg\min_{g \in \mathcal{G}_\theta} \widehat{R}(g; \mathcal{D}), \tag{3.6}$$

then we choose the overall best predictor $\widehat{g}^*$ by minimizing the error on a held-out *validation* set $\mathcal{D}''$

$$\widehat{g}^* = \arg\min_{\widehat{g}^*_\theta} \widehat{R}(\widehat{g}^*_\theta; \mathcal{D}''),$$

where we explicitly added $\mathcal{D}$ and $\mathcal{D}''$ as an argument of the error to emphasize that the two minimizations use two different sets. Indeed, $\mathcal{D}''$ has to be independent of both the training set $\mathcal{D}$ and the eventual test set $\mathcal{D}'$ (used for estimating the performance of the final classifier). This procedure is known as *hyperparameter optimization* or *hyperparameter tuning* in machine learning.

The particularity of hyperparameter optimization is that the evaluation of the function $f(\theta) = \widehat{R}(\widehat{g}^*_\theta; \mathcal{D}'')$ is computationally expensive. Indeed, the evaluation of $f(\theta)$ requires the optimization of $\widehat{g}_\theta \in \mathcal{G}_\theta$, that is, *training* $\widehat{g}_\theta$ on $\mathcal{D}$ and *testing* it on $\mathcal{D}''$, which can take hours or days, depending on the particular learning algorithm, the data size n, and the data dimensionality d. This feature relates hyperparameter optimization to *experimental design* where, say, we want to optimize a handful parameters of a detector by evaluating its performance using expensive simulations. When the number of hyperparameters is small (say, not more than 3), the usual procedure is simple *grid search*: we discretize the components of θ, and evaluate $f(\theta)$ for all members of the discrete set. For example, we train AdaBoost

for $T \in \{10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10\,000\}$ iterations using decision trees with a number of leaves of $N \in \{2, 3, 5, 10, 20, 50, 100\}$, and then select the best (T, N) pair out of the 70 trained functions. When the number of hyperparameters exceeds three, this procedure rapidly becomes infeasible because the number of grid points blows up exponentially. Simple heuristics, such as Latin hypercube search, simple random search [10], or gradient search [11] can be applied, but the principled solution is *surrogate optimization*: we replace f by a, usually non-parametric, smooth function which we train iteratively on a sequence of evaluation points $\big(\theta_t, f(\theta_t)\big)$. In each iteration t, a surrogate function $\widehat{f}_t(\theta)$ is regressed over the set $\big\{\theta_{t'}, f(\theta_{t'})\big\}_{t'=1}^{t}$, then a new evaluation point is selected based on $\widehat{f}_t(\theta)$. Gaussian process regression [12] is one of the most popular concrete choices for the regression method since it provides not only a regressor but also a conditional distribution $p\big(f(\theta) \mid \theta\big)$, and so it can be used with probabilistic criteria to select the next evaluation point θ_t. The best-known such criterion is *expected improvement*. The recent success of deep neural networks [13, 14, 15], with often tens of hyperparameters, generated a surge of papers on the subject [16, 10, 17, 18, 19], showing that the jury is still out on what the best strategy is.

3.2.1 An example in low-dimensional fitting

To illustrate that overfitting and model selection do not only concern multi-variate classification, we start by a simple example using Gaussian process (GP) regression [12] on a one-dimensional fitting problem. The main ingredient of a GP is the kernel function $K(x, x')$ that defines the smoothness of the regressor functions $\widehat{g}(x)$. In this example we use the squared exponential kernel

$$K(x, x') = a \exp\left(-\frac{(x - x')^2}{w^2}\right),$$

where a and w are hyperparameters to be tuned. Without going into the details, the GP regression function is given in the form of

$$\widehat{g}(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i).$$

The coefficient vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ is given by

$$\boldsymbol{\alpha} = (\mathbf{K} + \Sigma)^{-1}\mathbf{y},$$

where $\mathbf{K} = [K(x_i, x_j)]_{i,j=1}^{n}$ is the *Gram matrix* and $\mathbf{y} = (y_1, \ldots, y_n)$ is the label vector. The diagonal matrix Σ contains either the squared uncertainties σ_i^2 in the diagonal in case they are known, or a constant σ^2 which becomes a third hyperparameter that has to be tuned along with a and w.

There are both Bayesian and frequentist techniques to find the optimal hyperparameters a, w, and σ using a single training set $\mathcal{D}$. These methods work well in practice, so validation is not a common technique in GP regression. Nevertheless, for the sake of illustrating the general technique, we show on an example how validation works in this case. We generate 50 training and 50 validation points (x_i, y_i) by first drawing x_i uniformly from the interval $[3, 100]$, and then drawing $y_i \sim \mathcal{N}\big(g(x_i), 0.4\big)$, where

$$g(x) = \sin\big(\sqrt{x}\big)$$

is our target regression function. Figure 3.1 depicts $g(x)$ in blue, and the training set $\mathcal{D}$ in red (the validation set $\mathcal{D}''$ is not shown). For the sake of simplicity, we fix $a = 10$ and $\sigma = 0.4$ and only tune the kernel width w. The role of w is to set the smoothness of the regressor function. We plot three cases in Figure 3.1. When $w = 10$, the estimated regressor $\widehat{g}(x)$ clearly overfits the data: it follows the training data too closely, achieving a root mean square error (RMSE) $\sqrt{R_2(g)} = 0.33$ on the training set. At the same time, on the validation set its RMSE is 0.47, giving a qualitative indication of overfitting. When $w = 80$, we are underfitting (oversmoothing) the data. The training and validation RMSEs are close (0.47 and 0.48, respectively), but they are both suboptimal. At the optimum $w = 40$, selected by the validation procedure, we can achieve a 0.39 training RMSE and a 0.38 validation RMSE. The validation RMSE slightly underestimates the expected RMSE since w was selected based on this sample, but this

"hyper-overfitting" is negligible. In fact it is comparable to overfitting a single-parameter function in a parametric setup.

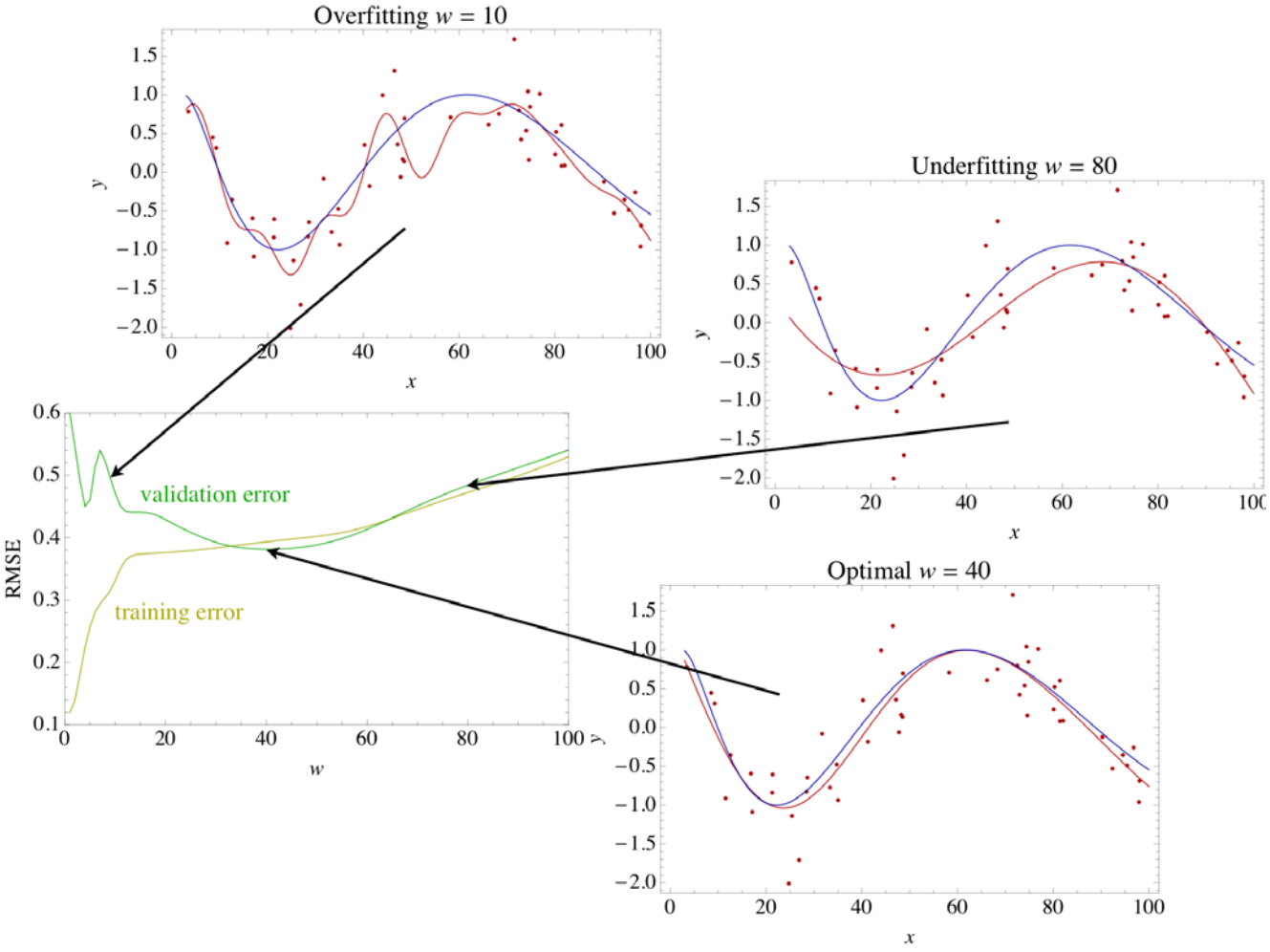


Figure 3.1: Over- and underfitting in non-parametric GP regression. The blue curve is the target regression function $g(x) = \sin(\sqrt{x})$. The red dots are the training set: x_i is drawn uniformly from $[3, 100]$, and $y_i \sim \mathcal{N}\big(g(x_i), 0.4\big)$. Two of the GP hyperparameters are fixed to $a = 10$ and $\sigma = 0.4$, and the third hyperparamater w is tuned on a validation set, yielding an optimal value of $w = 40$.

3.3 Convex losses in binary classification

Finding the empirical minimizer $\widehat{g}^*$ (3.4), $\widehat{g}^*_\theta$ (3.6), or $\widehat{g}^*_\beta$ (3.5) can be algorithmically challenging even in relatively "simple" function sets $\mathcal{G}$. For example, finding the *linear* binary classifier that minimizes the training error $\widehat{R}_{\mathbb{I}}(g)$ is NP hard [20].[3] One way to make the learning problem tractable is to relax the minimization of $\widehat{R}_{\mathbb{I}}(g)$ by defining *convex losses* that upper bound the one-loss $L_{\mathbb{I}}\big(g, (\mathbf{x}, y)\big)$ (3.1).

To formalize this, we need to introduce some notions used in binary classification. Most of the learning algorithms do not directly output a $\{\pm 1\}$-valued classifier g, rather, they learn a real-valued *discriminant function* f, and obtain g by thresholding the output of $f(\mathbf{x})$ at 0, that is,

$$g(\mathbf{x}) = \begin{cases} +1 & \text{if } f(\mathbf{x}) > 0, \\ -1 & \text{if } f(\mathbf{x}) \leq 0. \end{cases}$$

Using the real-valued output of the discriminant function, the classifier can be evaluated on a finer scale

[3] http://en.wikipedia.org/wiki/NP_(complexity)

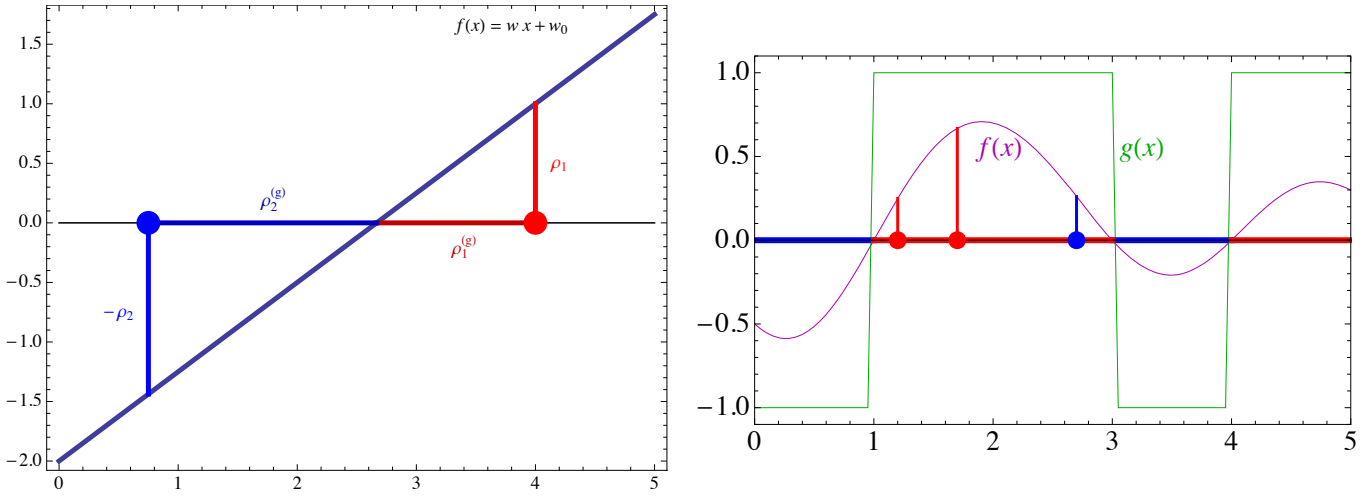
by defining the (*functional*) *margin*

$$\rho = \rho\big(f, (\mathbf{x}, y)\big) = f(\mathbf{x})y.$$

With this notation, the misclassification indicator (one-loss) of a discriminant function f on a sample point $(\mathbf{x}, y)$ can be re-defined as a function of the margin ρ by

$$L_{\mathbb{I}}\big(f, (\mathbf{x}, y)\big) = L_{\mathbb{I}}\Big(\rho\big(f, (\mathbf{x}, y)\big)\Big) = L_{\mathbb{I}}(\rho) = \mathbb{I}\{\rho < 0\}.$$

Besides the *sign* of ρ that represents the classification error, the *magnitude* of ρ is also important: it indicates the confidence or the robustness of the classification. Indeed, the combination of a large ρ with a Lipschitz-type (slope) penalty on f means that the *geometric* margin $\rho^{(\mathrm{g})}$ is also large, that is, $\mathbf{x}$ is far from the *decision border* (defined by the set of points for which $f(\mathbf{x}) = 0$, see Figure 3.2).



(a) (b)

Figure 3.2: Geometric and functional margins. The labels of the red and blue points are $y = +1$ and $y = -1$, respectively. (a) An upper bound ω on the slope w and a functional margin ρ implies a geometric margin (distance from the point $\mathbf{x} : f(\mathbf{x}) = 0$) of $\rho^{(\mathrm{g})} > \rho/\omega$. (b) The discriminant function $f(\mathbf{x})$ and the induced classifier $g(\mathbf{x})$. The blue point (with label $y = -1$) is misclassified and the two red points (with label $y = +1$) are correctly classified, but the second red point has a larger margin $f(x)y$ indicating that its classification is more robust.

The idea behind large margin classification is to design loss functions that "push" the decision border away from the training points. The goal is not just to minimize the training zero-one error $R_{\mathbb{I}}(g)$ (3.2) but also to increase the margin of the correctly classified points. The common feature of these loss functions is that 1) they penalize larger errors (negative margins) more than smaller ones, and 2) they keep penalizing even correctly classified points especially if they are close to the decision border (their margin is close to zero). Formally, one can define smooth convex upper bounds of the margin-based one-loss $L_{\mathbb{I}}(\rho)$, and minimize the corresponding empirical risks instead of the training error $\widehat{R}_{\mathbb{I}}$. The most common losses, depicted in Figure 3.3, are the *exponential loss*

$$L_{\mathrm{EXP}}(\rho) = \exp(-\rho), \tag{3.7}$$

the *hinge loss*

$$L_{1+}(\rho) = \max(0, 1 - \rho), \tag{3.8}$$

and the *logistic loss*

$$L_{\mathrm{LOG}}(\rho) = \log_2\big(\exp(-\rho) + 1\big). \tag{3.9}$$

Given the margin-based loss $L_\bullet(\rho)$, the corresponding margin-based empirical risk can be defined as

$$\widehat{R}_\bullet(f) = \frac{1}{n}\sum_{i=1}^{n} L_\bullet(f(\mathbf{x}_i)y_i) \tag{3.10}$$

Minimizing the margin-based empirical risks $\widehat{R}_{\text{EXP}}(f)$, $\widehat{R}_{1+}(f)$, or $\widehat{R}_{\text{LOG}}(f)$ have both algorithmic and statistical advantages. First, combining the minimization of these risks with Lipschitz-type penalties[4] often leads to convex optimization problems that can be solved with standard algorithms (e.g., linear, quadratic, or convex programming). Second, it can also be shown within the complexity regularization framework that the minimization of these penalized convex risks leads to large margin classifiers with good generalization properties, confirming the intuitive explanation of the previous paragraph.

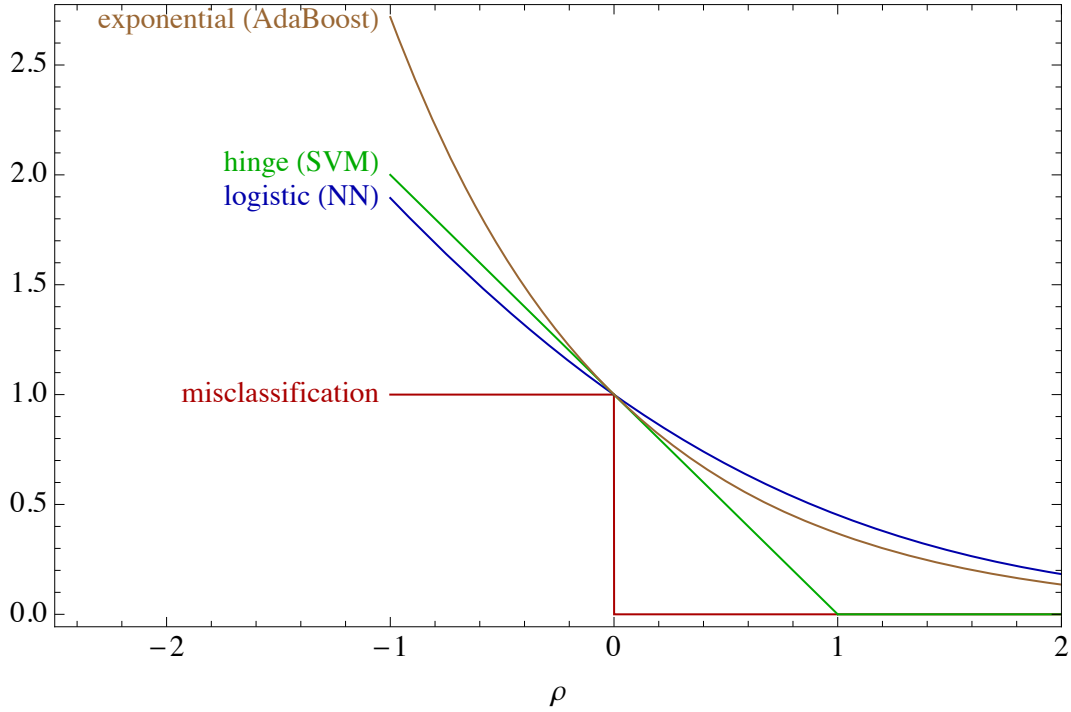


Figure 3.3: Convex margin losses used in binary classification.

3.4 Binary classification algorithms

The three most popular binary classification algorithms are the *multi-layer perceptron* or *neural network* (NN) [21, 1], the *support vector machine* (SVM) [22, 23, 9], and AdaBoost [24]. They have very different origins, and the algorithmic details also make them stand apart, nevertheless, they share some important concepts, and their practical success is explained, at least qualitatively, by the same theory on large margin classification.

An important pragmatic similarity is that they all learn *generalized linear* discriminant functions of the form

$$f(\mathbf{x}) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\mathbf{x}). \tag{3.11}$$

In neural networks $h^{(t)}(\mathbf{x})$ is a *perceptron*, that is, a linear combination of the input features followed by

[4] often of the form of L_1 or L_2 penalties on the coefficient vector of a *generalized linear* model; see Section 3.4

a sigmoidal nonlinearity σ (such as arctan)

$$h^{(t)}(\mathbf{x}) = \sigma\left(w_0^{(t)} + \sum_{j=1}^{d} w_j^{(t)} x^{(j)}\right),$$

where $x^{(j)}$ is the jth component of the d-dimensional input vector $\mathbf{x}$. The output weight vector $\boldsymbol{\alpha} = \left(\alpha^{(1)}, \ldots, \alpha^{(T)}\right)$ and the $T \times (d+1)$-dimensional input weight matrix

$$\mathbf{W} = \begin{pmatrix} w_0^{(1)} & \ldots & w_d^{(1)} \\ \vdots & \ddots & \vdots \\ w_0^{(T)} & \ldots & w_d^{(T)} \end{pmatrix}$$

are learned using gradient descent optimization (called *back-propagation* in the NN literature). In the case of binary classification the neural network is usually trained to minimize the logistic loss (3.9). One of the advantages of NNs is their versatility: as long as the loss function is differentiable, it can be plugged into the algorithm. The differentiability condition imposed by the gradient descent optimization, on the other hand, constrains the loss function and the nonlinearities used in the hidden units: one could not use, say, a Heaviside-type nonlinearity or the one-loss.

The invention of the multilayer perceptron in 1986 [21] was a technological breakthrough: complex pattern recognition problems (e.g., handwritten character recognition) could suddenly be solved efficiently using neural networks. At the same time, the theory of machine learning at these early days was not yet developed to the point of being able to explain the principles behind algorithmic design. Most of the engineering techniques (such as *weight decay* or *early stopping*) were developed using a trial-and-error approach, and they were theoretically justified only much later within the complexity regularization and large margin framework [25]. Partly because of the "empirical" nature of neural network design, a common belief developed about the "user-unfriendliness" of neural networks. Added to this reputation was the uneasiness of having a non-convex optimization problem: back-propagation cannot be guaranteed to converge to a global minimum of the empirical risk. This is basically a no-issue in practice (especially on today's large problems), still, it is a common criticism from people who usually have never experimented with neural networks on real problems. As a consequence, neural networks were slightly over-shadowed in the 90s with the appearance of the support vector machine and AdaBoost. Today neural networks are, again, becoming more popular partly because of user-friendly program packages (e.g., [26]), partly due to the computational efficiency of the training algorithm (especially its stochastic version), and partly because of the recent success of unsupervised feature-learning techniques that use deep neural architectures to solve hard computer vision and language processing problems [13, 14, 15].

Support vector machines also learn generalized linear discriminant functions of the form (3.11) with

$$h^{(t)}(\mathbf{x}) = y_t K(\mathbf{x}_t, \mathbf{x}),$$

where $K(\mathbf{x}, \mathbf{x}')$ is a positive semidefinite *kernel* function that expresses the similarity between two observations $\mathbf{x}$ and $\mathbf{x}'$. $K(\mathbf{x}, \mathbf{x}')$ is usually monotonically decreasing with the distance between $\mathbf{x}$ and $\mathbf{x}'$. As in Gaussian process regression (Section 3.2.1), the most common choice for K is the squared exponential (a.k.a. Gaussian) kernel. The index t ranges over $t = 1, \ldots, n$ which means that each training point $(\mathbf{x}_t, y_t) \in \mathcal{D}$ contributes to f a kernel function centered at $\mathbf{x}_t$ with a sign equal to the label y_t, so the final discriminant function can be interpreted as a weighted *nearest neighbor* classifier where the weight comprises of the "similarity term" $K(\mathbf{x}_t, \mathbf{x})$ and the "importance term" $\alpha^{(t)}$.

The objective of training the SVM is to find the weight vector $\boldsymbol{\alpha} = \left(\alpha^{(1)}, \ldots, \alpha^{(T)}\right)$ that minimizes the hinge loss (3.8) with a complexity penalty (the L_2 loss on the weights of features in the Hilbert space induced by $K(\mathbf{x}, \mathbf{x}')$). The objective function was explicitly designed based on the theory of large margin classification to ensure good generalization properties. A great advantage of the setup is that the objective is a quadratic function of $\boldsymbol{\alpha}$ with linear constraints $0 \leq \alpha^{(t)} \leq C$ (the so-called *box* constraints), so the global optimum can be found using quadratic programming. The result is sparse: only a subset of the training points in $\mathcal{D}$ have nonzero coefficients $\alpha^{(t)}$. These points are called *support vectors*, giving the

name of the method.

The biggest disadvantage of the technique is its training time: naïve quadratic programming solvers run in super-quadratic time, and even with the most sophisticated tricks it is hard to beat the $O(n^2)$ barrier. The second disadvantage of the method is that in high-dimensional problems the number of support vectors is comparable to the size of the training set[5], so, when evaluating f at the test phase is a bottleneck (see Section 3.6.1), SVMs can be prohibitively slow. Third, unlike neural networks, SVMs are designed for binary classification, and the generic extensions for multi-class classification and regression do not reproduce the performance of binary SVM. Despite these disadvantages, the appearance of the support vector machine in the middle of the 90s revolutionized the technology of pattern recognition. Besides its remarkable generalization performance, the small number of hyperparameters[6] and the appearance of turn-key software packages[7] made SVMs the method of choice for a wide range of applications involving small-to-moderate size training sets. With the appearance of extra-large training sets in the last decade, training time and optimization became more important issues than generalization [27], so SVMs lost somewhat their dominant role.

AdaBoost learns a generalized linear discriminant function of the form (3.11) in an iterative fashion. It can be considered as constructing a neural network by adding one neuron at a time, but since back-propagation is no longer applied, there is no differentiability restriction on the *base classifiers* $h^{(t)}$. This opens the door to using domain-dependent features, making AdaBoost easy to adapt to a wide range of applications. The basic binary classification algorithm (Figure 3.4) consists of elementary steps that can be implemented by a first year computer science student in an hour. The only tricky step is the implementation of the base learner (line 3) whose goal is to return $h^{(t)}$ with a small *weighted* error

$$\widehat{R}_{\mathbb{I}}(h, \mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} w_i \mathbb{I}\{h(\mathbf{x}_i) \neq y_i\},$$

but most of the simple learning algorithms (e.g., decision trees, linear classifiers) can be easily adapted to this modified risk. The weights $\mathbf{w} = (w_1, \ldots, w_n)$ over the training points are initialized uniformly (line 1), and then updated in each iteration by a simple and intuitive rule (lines 7-10): if $h^{(t)}$ misclassifies $(\mathbf{x}_i, y_i)$, the weight w_i increases (line 8), whereas if $h^{(t)}$ correctly classifies $(\mathbf{x}_i, y_i)$, the weight w_i decreases (line 10). In this way subsequent base classifiers will concentrate on points that were "missed" by previous base classifiers. The coefficients $\alpha^{(t)}$ are also set analytically to the formula in line 5 which is monotonically decreasing with respect to the weighted error.

This simple algorithm has several beautiful mathematical properties. First it can be shown [24] that if each base classifier $h^{(t)}$ is slightly better than a random guess (that is, every weighted error $\epsilon^{(t)} \leq \frac{1}{2} - \delta$ with $\delta > 0$), then the (unweighted) training error $\widehat{R}_{\mathbb{I}}$ of the final discriminant function $f^{(T)}$ becomes zero after at most

$$T = \left\lceil \frac{\log n}{2\delta^2} \right\rceil + 1$$

steps. The logarithmic dependence of T on the data size n implies that the technique is formally a *boosting* algorithm from which the second part of its name derives.[8] Second, it can be shown that the algorithm minimizes the exponential risk $\widehat{R}_{\text{EXP}}(f)$ (3.7) using a greedy functional gradient approach [28, 29]. In this alternative but equivalent formulation, in each iteration $h^{(t)}$ is selected to maximize the decrease (derivative) of $\widehat{R}_{\text{EXP}}\big(f^{(t-1)} + \alpha h^{(t)}\big)$ at $\alpha = 0$, and then $\alpha^{(t)}$ is set to

$$\alpha^{(t)} = \arg\min_{\alpha} \widehat{R}_{\text{EXP}}\big(f^{(t-1)} + \alpha h^{(t)}\big).$$

Furthermore, for inseparable data sets (for which no linear combination of base classifiers achieves 0 training error), the exponential risk $\widehat{R}_{\text{EXP}}(f^{(T)})$ goes to the minimum achievable exponential risk as $T \to \infty$

[5] This loss of sparsity is one of the manifestations of the phenomenon known as the *curse of dimensionality*.

[6] C in the bound of the αs and a length scale parameter of the kernel function $K(\mathbf{x}, \mathbf{x}')$

[7] `http://svmlight.joachims.org`, `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, `http://www.torch.ch`, `http://www.loria.fr/~lauer/MSVMpack`

[8] The algorithm is also Ada*ptive* because the coefficients $\alpha^{(t)}$ depend on the errors $\epsilon^{(t)}$ of the base classifiers $h^{(t)}$.

AdaBoost$\big(\mathcal{D}, \text{Base}(\cdot,\cdot), T\big)$

$\mathbf{w}^{(1)} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weights*

for $t \leftarrow 1$ **to** T

  $h^{(t)} \leftarrow \text{Base}\big(\mathcal{D}, \mathbf{w}^{(t)}\big)$ ▷ *base classifier*

  $\epsilon^{(t)} \leftarrow \sum_{i=1}^{n} w_i^{(t)} \mathbb{I}\left\{h^{(t)}(\mathbf{x}_i) \neq y_i\right\}$ ▷ *weighted error of the base classifier*

  $\alpha^{(t)} \leftarrow \frac{1}{2} \ln\left(\frac{1-\epsilon^{(t)}}{\epsilon^{(t)}}\right)$ ▷ *coefficient of the base classifier*

  for $i \leftarrow 1$ **to** n ▷ *re-weighting the training points*

    if $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then** ▷ *error*

      $w_i^{(t+1)} \leftarrow \frac{w_i^{(t)}}{2\epsilon^{(t)}}$ ▷ *weight increases*

    else ▷ *correct classification*

      $w_i^{(t+1)} \leftarrow \frac{w_i^{(t)}}{2\left(1-\epsilon^{(t)}\right)}$ ▷ *weight decreases*

return $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$ ▷ *weighted "vote" of base classifiers*

Figure 3.4: The pseudocode of binary AdaBoost. $\mathcal{D} = \big\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\big\}$ is the training set, $\text{Base}(\cdot,\cdot)$ is the base learner, and T is the number of iterations.

[30]. It can also be proven [31] that for the *normalized* discriminant function

$$\widetilde{f}(\mathbf{x}) = \frac{\sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\mathbf{x})}{\sum_{t=1}^{T} \alpha^{(t)}},$$

the *margin-based* training error

$$\widehat{R}_{\mathbb{I}}^{(\theta)}(\widetilde{f}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\left\{\widetilde{f}(\mathbf{x}_i) y_i < \theta\right\}$$

also goes to zero exponentially fast for all $\theta < \widetilde{\rho}^*/2$, where

$$\widetilde{\rho}^* = \max_{\widetilde{\boldsymbol{\alpha}}: \sum_{t=1}^{T} |\widetilde{\alpha}_t| = 1} \min_i \sum_{t=1}^{T} \widetilde{\alpha}^{(t)} h^{(t)}(\mathbf{x}_i) y_i$$

is the maximum achievable (normalized) minimum margin. This results shows that AdaBoost, similarly to the support vector machine, leads to a large margin classifier. It also explains the surprising experimental observation that the generalization error $R_{\mathbb{I}}(f)$ (estimated on a test set) often decreases even after the training error $\widehat{R}_{\mathbb{I}}(f)$ reaches 0.

AdaBoost arrived to the pattern recognition scene in the late 90s when support vector machines were in full bloom. AdaBoost has a lot of advantages over its main rival: it is fast (its time complexity is linear in n and T), it is an *any-time* algorithm[9], it has few hyperparameters, it is resistant to overfitting, and it can be directly extended to multi-class classification [32]. Due to these advantages, it rapidly became the method of choice of machine learning experts in certain types of problems where the natural setup is to define a large set of plausible features of which the final solution $f^{(T)}$ uses only a few (a so called *sparse* classifier). Arguably the most famous application is the first face detector running real-time

[9] It outputs a classifier even if it is stopped before convergence.

on 30 frame-per-second video [33]. At the same time, AdaBoost is much less known among users with no computer science background than the support vector machine. The reason is paradoxically the simplicity of AdaBoost: since it is so easy to implement with a little background in programming, no machine learning expert took the effort to provide a turn-key software package easily usable for non-experts.[10] In fact it is not surprising that the only large non-computer-scientist community in which AdaBoost is much more popular than SVM is experimental physics: physicists, especially in high energy physics, are heavy computer-users with considerable programming skills. In the last five years AdaBoost (and other similar *ensemble* methods) seem to have been taking over the field of large-scale applications. In recent large-scale challenges [34, 35] the top entries are dominated by ensemble-based solutions, and SVM is almost non-existent in the most efficient approaches.

Although binary AdaBoost (Figure 3.4) is indeed simple to implement, multi-class AdaBoost has some nontrivial tricks. There are several multi-class extensions of the original binary algorithm, and it is not well-known, even among machine learning experts, that the original AdaBoost.M1 and AdaBoost.M2 algorithms [24] are largely suboptimal compared to AdaBoost.MH published a couple of years later [32].[11] On the other hand, the AdaBoost.MH paper [32] did not specify the implementation details of multi-class base learning, making the implementation non-trivial. For more information on multi-class AdaBoost.MH we refer the reader to the documentation of MultiBoost, available from the `http://multiboost.org` website.

3.5 Applications

In this section we illustrate the versatility of the abstract supervised learning model described in this chapter through presenting some of the machine learning applications we have worked on. It turns out that real-world applications are never so simple as just taking a turn-key implementation out of the box and running it. To make a system work, one needs both domain expertise and machine learning expertise, and so it often requires an interdisciplinary approach and considerable communication effort from experts of different backgrounds. Nevertheless, once the problem is reduced to the abstract setup described in Section 3.1, machine learning algorithms can be very efficient.

As the examples will show, classification or regression is only one step in the pattern recognition pipeline (Figure 3.5). Data collection is arguably the most costly step in most of the applications. In particular, harvesting the labels y_i usually involves human interaction. How to do this in a cost-effective way by, for example, using CAPTCHAs to obtain character labels [36] or making people label images by playing a game [37, 38], is itself a challenging research subject. On the other hand, in experimental physics simulators can be used to sample from the distribution $p(\mathbf{x} \mid y)$, so in these applications data collection is usually not an issue.

The second step of the pipeline is preprocessing the data. There is a wide range of techniques that can be applied here, usually with the objective of simplifying the supervised learning task. First, if the observations are not already in a format of a real vector $\mathbf{x} \in \mathbb{R}^d$, *features* (column vectors $(x_1^{(j)}, \ldots, x_n^{(j)})$ of the data matrix) should be extracted from the raw data. The goal here is to find features that are plausibly correlated with the label y, so domain knowledge is almost always necessary to carry out this step. Since learning algorithms usually deteriorate as the dimension of the input $\mathbf{x}$ increases (because of the so called *curse of dimensionality*), dimensionality reduction is often part of data preprocessing. Principal component analysis is the basic tool that is usually applied, but nonlinear manifold algorithms [39, 40] may also be useful if the training set is not too large. The output space $\mathcal{C}$ can also be transformed to massage the original problem into a setup that can be solved by standard machine learning tools. This can be done in an ad hoc way, but there exist also principled *reduction* techniques between machine learning problems (binary/multi-class classification, regression, even reinforcement learning) [41].

After training, postprocessing the results can also be an important step. If, for example, the original complex problem was reduced to an easy-to-solve setup, re-transforming the obtained labels and even re-

[10] We are attempting to improve the situation with our free multi-class program package available at `http://multiboost.org`.

[11] The commonly used WEKA package (`http://www.cs.waikato.ac.nz/ml/weka`), for example, only contains a badly written implementation of AdaBoost.M1, effectively harming the reputation of boosting as a whole.

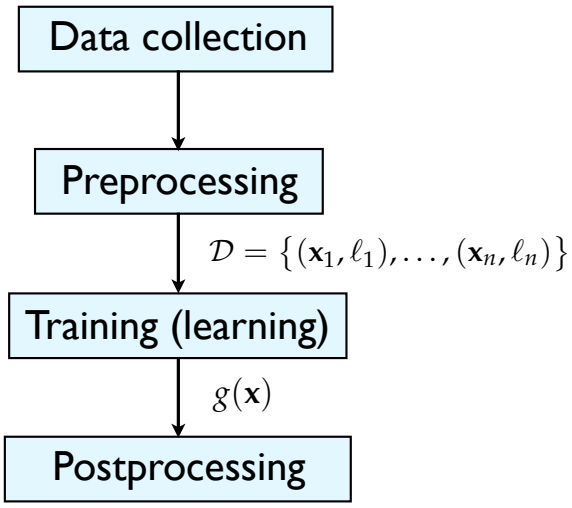


Figure 3.5: The pattern recognition pipeline.

calibrating the solution is often a good idea. In well-documented challenges of the last five years [34, 42, 35] we have also learned that the most competitive results are obtained when divers models trained using different algorithms are combined, so *model aggregation* techniques are also becoming part of the generic machine learning toolbox.

Sometimes it happens that an application poses a specific problem that can not be solved with existing tools. Solving such a problem and generalizing the solution to make it applicable for similar problems is one of the motivational engines behind the development of the machine learning domain.

3.5.1 Music classification, web page ranking, muon counting

In [43] we use AdaBoost for telling apart speech from music. The system starts by constructing the spectrogram on a set of recorded audio signals which constitute the observation vectors $\mathbf{x}$ in this case. AdaBoost is then run in a feature space inspired by image classification on the spectrogram "images". The output y of the system is binary, that is, $y \in \mathcal{C} = \{\text{MUSIC}, \text{SPEECH}\}$. In [44] we stay within the music classification domain but we tackle a more difficult problem: finding the performing artist and the genre of a song based on the audio signal of the first 30 s of the song. The first module of the classifier pipeline is an elaborate signal processor that collects a vector of features for each 2 s segment and then aggregates them to constitute an observation vector $\mathbf{x}$ with about 800 components per song. We train two systems, one for finding the artist performing the song and another for predicting its genre. This application also stretches the limits of the classical multi-class (single-label) setup. It is plausible that a song belongs to several genres leading to a so-called *multi-label* classification. It may also be useful to train a unique system for predicting both the artist and the genre at the same time. This problem is known as *multi-task* learning.

The multi-variate regression problem can be illustrated by our recent work [45] in which we aim to predict the number of muons in a signal recorded in a water Cherenkov detector of the Pierre Auger experiment [46]. The pipeline, again, starts by extracting features that are plausibly correlated with the muon content of the signal. The 172 features are quite correlated, so first we apply principal component analysis to reduce the size of the observation vector $\mathbf{x}$ to 19. Then we train a neural network [1, 26] and convert its output into a point estimate and a pointwise error bar (uncertainty) of the number of muons.

In [47, 48] we use AdaBoost.MH for web page ranking. The input $\mathbf{x}$ of the classifier g is a set of features representing a search query and a document, and the label y represents the relevance of the document to the query. The goal is to rank the documents in order of their relevance. Of course, if the relevance of the document is correctly predicted, the implied ranking will also be good. Nevertheless, it is hard to formally derive a meaningful loss for each (document, query, relevance) triplet from the particular loss defined on rankings. The solution to this problem is a post-processing technique called *calibration*: instead of directly using the output of the classifier g, we send it through another regressor or ranker which

is now trained to minimize the desired risk. Our concrete system also contains another postprocessing module called *model aggregation*: instead of training one classifier g, we train a large number of classifiers by varying data preprocessing and algorithmic hyperparameters, and combine the results using a simple weighted voting scheme.

3.6 Three open research problems

In this section we briefly describe three open research problems that are either relevant to or even motivated by certain unorthodox applications of multivariate discrimination in experimental physics.

3.6.1 Trigger design: classification with test-time constraints

One of the recent applications of multivariate discriminants in high-energy physics is trigger design [49]. The goal of a trigger is to separate *signals* generated by a phenomenon to be detected or measured from *background*, which is an observed event that just happen to look like real signal because of random fluctuations or due to some uninteresting phenomena. The final goal of the analysis is to collect a large statistics of observations that, with high probability, were generated by the targeted phenomenon. Since the background is often several orders of magnitudes more frequent than the signal, part of the background/signal separation cannot be done offline. Due to either limited disk capacity or limited bandwidth, most of the raw signal has to be discarded by online triggers.

In the machine learning paradigm, a trigger is just a binary classifier with

$$\mathcal{C} = \{\text{SIGNAL}, \text{BACKGROUND}\}.$$

There are several attributes that make the trigger a *special* binary classifier. First, it is extremely unbalanced: the probability of BACKGROUND is practically 1 in a lot of cases. This makes the classical setup of minimizing the error probability $R_{\mathbb{I}}(g)$ (3.2) inadequate: it is very hard to beat the trivial constant classifier $g(\mathbf{x}) \equiv \text{BACKGROUND}$ which has an error of $R_{\mathbb{I}}(g) = P(y = \text{SIGNAL})$. Indeed, the natural *gain* in trigger design is the *true positive* or *hit* indicator

$$G_{\text{TP}}\big(g, (\mathbf{x}, y)\big) = \mathbb{I}\{g(\mathbf{x}) = \text{SIGNAL} \mid y = \text{SIGNAL}\}.$$

Taking the complement of the true positive indicator

$$L_{\overline{\text{TP}}}\big(g, (\mathbf{x}, y)\big) = 1 - \mathbb{I}\{g(\mathbf{x}) = \text{SIGNAL} \mid y = \text{SIGNAL}\} = \mathbb{I}\{g(\mathbf{x}) = \text{BACKGROUND} \mid y = \text{SIGNAL}\}$$

as the loss, the implied risk is

$$R_{\overline{\text{TP}}}(g) = \mathbb{E}_{(\mathbf{x},y)\sim\mathfrak{D}}\left\{L_{\overline{\text{TP}}}\big(g, (\mathbf{x}, y)\big)\right\} = \mathbb{P}\{g(\mathbf{x}) = \text{BACKGROUND} \mid y = \text{SIGNAL}\}.$$

Again, minimizing $R_{\overline{\text{TP}}}(g)$ is trivial by setting $g(\mathbf{x}) \equiv \text{SIGNAL}$ this time, so the goal is to minimize $R_{\overline{\text{TP}}}(g)$ with a *constraint* that the *false positive* rate

$$R_{\text{FP}}(g) = P\big(g(\mathbf{x}) = \text{SIGNAL} \mid y = \text{BACKGROUND}\big)$$

is kept below a fixed level p_{FP}. As we mentioned above, in triggers we have $P(y = \text{SIGNAL}) \ll P(y = \text{BACKGROUND})$, so the false positive rate $R_{\text{FP}}(g)$ is approximately equal to the unconditional positive rate $P(g(\mathbf{x}) = \text{SIGNAL})$. In experimental physics terminology this means that a constraint is imposed on the *trigger rate*.

The second attribute that makes trigger design special is that we have strict computational constraints imposed on the evaluation of the classifier g on test samples $\mathbf{x}$. Typically, observations $\mathbf{x}$ arrive at a given rate and $g(\mathbf{x})$ has to be run online. The designer can have some flexibility on the parallel handling of the incoming events, but computational resources are often limited. In certain cases[12] the electric

[12] For example, in JEM EUSO [50], where the detector will be installed on the International Space Station.

consumption may also be limited and harsh conditions may require the use of robust hardware with low clock-rate.

Trigger design shares these two attributes (unbalanced classes and test-time constraints) with object detection in computer vision where machine learning has been applied widely. For example, when the goal is to detect faces in images, the probability of the signal class (face) is much lower then the probability of background (everything else). We have also computational constraints at test time if the goal is to detect faces *online* in video recordings with given frame rate and the detector hardware must fit into a compact camera. What makes trigger design slightly more challenging is the sometimes extremely low signal probability and the fact that the computational cost of each feature (components of $\mathbf{x}$) can vary in a large range. For example, the LHCb trigger [51, 49] can use "cheap" observables that can be evaluated fast to rapidly obtain a rough classifier. One can also almost reconstruct the collision event which can take up a large portion of the allotted time, but the resulting feature can be used reliably to discard background events.

This example shows why a natural answer to these challenges is to design *cascade* classifiers both in experimental physics and in object detection [33, 52, 53, 54, 55, 56]. A cascade classifier $g(\mathbf{x})$ is composed of a list of simpler binary classifiers $h_1(\mathbf{x}), \ldots, h_N(\mathbf{x})$ evaluated sequentially. For $j < N$, if $h_j(\mathbf{x})$ classifies the observation $\mathbf{x}$ negatively, the final classification of $g(\mathbf{x})$ is BACKGROUND, and if the output of $h_j(\mathbf{x})$ is positive, the observation is sent to the next $(j+1)$th *stage* (or *level* in the terminology of experimental physics). The classifier $h_N(\mathbf{x})$ at the last stage is the only one that can classify $\mathbf{x}$ as a SIGNAL. In computer vision, the stage classifiers $h_j(\mathbf{x})$ are usually learned using classification algorithms (most often AdaBoost). Some of the newest detection algorithms also attempt to learn the cascade structure automatically, nevertheless, manual experimentation is usually required to set hyperparameters (stagewise false positive/false negative rates, computational complexity of stage classifiers $h_j(\mathbf{x})$). In [57], we present a principled approach that can be used to automatically design test-time constrained classifiers. The automatic design also allows us to go beyond the cascade structure which is, although quite intuitive, an artificial constraint to keep the classifier structure simple and to accommodate manual tuning.

3.6.2 Learning to discover

The main application of multivariate discrimination in high-energy particle physics is, interestingly, not classification. The goal in these applications is to increase the sensitivity of counting-based statistical tests [58, 59]. Intuitively, the goal is to find regions of the feature space where the signal is present or where it is amplified with respect to its average abundance. Once the subregion is found, we claim the discovery of a novel phenomenon (particle) when the number of events in the region is significantly higher than that predicted by the pure background hypothesis. The simplest formal goal, motivated by a Poisson test, is to maximize

$$G(f) = \frac{s}{\sqrt{b}} \tag{3.12}$$

where

$$\begin{aligned} s &= \sum_{i=1}^{n} \mathbb{I}\{f(\mathbf{x}_i) = \text{SIGNAL} \wedge y_i = \text{SIGNAL}\} \\ b &= \sum_{i=1}^{n} \mathbb{I}\{f(\mathbf{x}_i) = \text{SIGNAL} \wedge y_i = \text{BACKGROUND}\}. \end{aligned}$$

Although $G(f)$ has the right "flavor", it does not take into consideration the statistical fluctuations of the test when s and/or b are small, so other, more sophisticated, approximate criteria have also been derived [60]. Whereas all these criteria (including (3.12)) are clearly related to the classical classification error $R_{\mathbb{I}}(f)$, the two are not equivalent. A notable difference is that the *expectation* of G depends on the sample size n, whereas increasing sample size only decreases the *variance* of $R_{\mathbb{I}}$. Nevertheless, the standard practice is to learn a discriminant function $f : \mathbb{R}^d \to \mathbb{R}$ on $\mathcal{D}$ using standard classification methods that minimize the classification error R, and then use G only to optimize a threshold θ which

defines the function g_θ by

$$g_\theta(\mathbf{x}) = \begin{cases} \text{SIGNAL} & \text{if } f(\mathbf{x}) > \theta \\ \text{BACKGROUND} & \text{otherwise.} \end{cases}$$

This way of using multivariate discriminants raises several interesting research questions. First, given a concrete test, what should the training criteria G be? Second, what is the relationship between G and $R_{\mathbb{I}}(f)$? Finally, how to adapt classical classification algorithms to the given criteria?

3.6.3 Deep learning for automatic feature construction

Finally let us raise a completely open research question. The current technology of multivariate discrimination allows us to classify objects that are represented by vectors $\mathbf{x} = \left(x^{(1)}, \ldots, x^{(d)}\right)$ of fixed length d. The representation has to be "semantically aligned", that is, $x_1^{(j)}$ must be comparable to $x_2^{(j)}$ in two objects $\mathbf{x}_1$ and $\mathbf{x}_2$. At the same time, raw observation, be it pixels of an image or electronic channels in a detector event, seldom comes in this form. Today, the process of translating the raw observation into a semantically aligned vector of features is in the hands of the domain expert. One of the most important movements of the last decade in machine learning is the development of a family of techniques for building deep unsupervised neural architectures [13]. The goal is to construct representations in a mostly non-supervised way that can capture deep invariances in the raw data. Although the field is rather young, it already had a major impact on natural language processing [14] and computer vision [15]. Interesting and natural questions are whether these techniques can be adapted to scientific data, whether the invariances in, say, raw detector data are sufficient to construct representations that could be useful in the ultimate analysis, and whether automatic or semi-automatic feature extraction can improve on the manually constructed representations.

References

[1] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[3] N. Cristianini and J. Shawe-Taylor, *Kernel methods for pattern recognition*, Cambridge University Press, 2004.

[4] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, Springer, New York, 1996.

[5] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer-Verlag, 2009.

[6] K.P. Murphy, *Machine Learning: A Probabilistic Perspective*, MIT Press, Cambridge, MA, 2012.

[7] Y. Freund and R.E. Schapire, *Boosting: Foundations and Algorithms*, MIT Press, Cambridge, MA, 2012.

[8] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.

[9] V. N. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.

[10] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, 2012.

[11] Y. Bengio, "Gradient-based optimization of hyperparameters," *Neural Computation*, vol. 12, no. 8, pp. 1889–1900, 2000.

[12] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, MIT Press, 2006.

[13] Y. Bengio, A.C. Courville, and P. Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," *CoRR*, vol. abs/1206.5538, 2012.

[14] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[15] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*. 2012, vol. 25, MIT Press.

[16] J. Bergstra, R. Bardenet, B. Kégl, and Y. Bengio, "Algorithms for hyperparameter optimization," in *Advances in Neural Information Processing Systems (NIPS)*. The MIT Press, 2011, vol. 24.

[17] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, 2012, vol. 25.

[18] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Automated selection and hyper-parameter optimization of classification algorithms," Tech. Rep., `http://arxiv.org/abs/1208.3719`, 2012.

[19] R. Bardenet, M. Brendel, B. Kégl, and M. Sebag, "Collaborative hyperparameter tuning," in *International Conference on Machine Learning (ICML)*, 2013.

[20] D. S. Johnson and F. P. Preparata, "The densest hemisphere problem," *Theoretical Computer Science*, vol. 6, pp. 93–107, 1978.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[22] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers," in *Fifth Annual Workshop on Computational Learning Theory*, 1992, pp. 144–152.

[23] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[24] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, pp. 119–139, 1997.

[25] P. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 525–536, 1998.

[26] I. Nabney, *Netlab: Algorithms for Pattern Recognition*, Springer, 2002.

[27] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Advances in Neural Information Processing Systems*, 2008, vol. 20, pp. 161–168.

[28] L. Mason, P. Bartlett, J. Baxter, and M. Frean, "Boosting algorithms as gradient descent," in *Advances in Neural Information Processing Systems*. 2000, vol. 12, pp. 512–518, The MIT Press.

[29] L. Mason, P. Bartlett, and J. Baxter, "Improved generalization through explicit optimization of margins," *Machine Learning*, vol. 38, no. 3, pp. 243–255, March 2000.

[30] M. Collins, R.E. Schapire, and Y. Singer, "Logistic regression, AdaBoost and Bregman distances," *Machine Learning*, vol. 48, pp. 253–285, 2002.

[31] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: a new explanation for the effectiveness of voting methods," *Annals of Statistics*, vol. 26, no. 5, pp. 1651–1686, 1998.

[32] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.

[33] P. Viola and M. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, pp. 137–154, 2004.

[34] G. Dror, M. Boullé, I. Guyon, V. Lemaire, and D. Vogel, Eds., *Proceedings of KDD-Cup 2009 competition*, vol. 7 of *JMLR Workshop and Conference Proceedings*, 2009.

[35] Olivier Chapelle and Yi Chang, "Yahoo! Learning-to-Rank Challenge overview," in *Yahoo! Learning-to-Rank Challenge (JMLR W&CP)*, 2011, vol. 14, pp. 1–24.

[36] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and Blum M., "reCAPTCHA: Human-based character recognition via web security measures," *Science*, vol. 321, no. 5895, pp. 1465–1468, 2008.

[37] L. Von Ahn and L. Dabbish, "Labeling images with a computer game," in *Conference on Human factors in computing systems (CHI04)*, 2004, pp. 319–326.

[38] L. Von Ahn, "Games with a purpose," *Computer*, vol. 39, no. 6, 2006.

[39] S. Roweis and Saul L. K., "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000.

[40] J. B. Tenenbaum, V. de Silva, and Langford J. C., "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.

[41] A. Beygelzimer, J. Langford, and B. Zadrozny, *Performance Modeling and Engineering*, chapter Machine Learning Techniques–Reductions Between Prediction Quality Metrics, Springer, 2008.

[42] J. Bennett and S. Lanning, "The Netflix prize," in *KDDCup 2007*, 2007.

[43] N. Casagrande, D. Eck, and B. Kégl, "Geometry in sound: A speech/music audio classifier inspired by an image classifier," in *International Computer Music Conference*, Sept. 2005, vol. 17.

[44] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kégl, "Aggregate features and AdaBoost for music classification," *Machine Learning Journal*, vol. 65, no. 2/3, pp. 473–484, 2006.

[45] B. Kégl, R. Busa-Fekete, K. Louedec, R. Bardenet, X. Garrido, I.C. Mariş, D. Monnier-Ragaigne, S. Dagoret-Campagne, and M. Urban, "Reconstructing $N_{\mu 19}(1000)$," Tech. Rep. 2011-054, Auger Project Technical Note, 2011.

[46] Pierre Auger Collaboration, "Pierre Auger project design report," Tech. Rep., Pierre Auger Observatory, 1997.

[47] R. Busa-Fekete, B. Kégl, T. Éltető, and Gy. Szarvas, "Ranking by calibrated AdaBoost," in *Yahoo! Ranking Challenge 2010 (JMLR workshop and conference proceedings)*, 2011, vol. 14, pp. 37–48.

[48] R. Busa-Fekete, B. Kégl, T. Éltető, and Gy. Szarvas, "A robust ranking methodology based on diverse calibration of AdaBoost," in *European Conference on Machine Learning*, 2011, vol. 22, pp. 263–279.

[49] V. Gligorov and M. Williams, "Efficient, reliable and fast high-level triggering using a bonsai boosted decision tree," Tech. Rep., arXiv:1210.6861, 2012.

[50] Y. Takizawa, T. Ebisuzaki, Y. Kawasaki, M. Sato, M. E. Bertaina, H. Ohmori, Y. Takahashi, F. Kajino, M. Nagano, N. Sakaki, N. Inoue, H. Ikeda, Y. Arai, Y. Takahashi, T. Murakami, James H. Adams, and the JEM-EUSO Collaboration, "JEM-EUSO: Extreme Universe Space Observatory on JEM/ISS," *Nuclear Physics B - Proceedings Supplements*, vol. 166, pp. 72–76, 2007.

[51] V. Gligorov, "A single track HLT1 trigger," Tech. Rep. LHCb-PUB-2011-003, CERN, 2011.

[52] L. Bourdev and J. Brandt, "Robust object detection via soft cascade," in *Conference on Computer Vision and Pattern Recognition*. 2005, vol. 2, pp. 236–243, IEEE Computer Society.

[53] R. Xiao, L. Zhu, and H. J. Zhang, "Boosting chain learning for object detection," in *Ninth IEEE International Conference on Computer Vision*, 2003, vol. 9, pp. 709–715.

[54] J. Sochman and J. Matas, "WaldBoost – learning for time constrained sequential detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 150–156.

[55] B. Póczos, Y. Abbasi-Yadkori, Cs. Szepesvári, R. Greiner, and N. Sturtevant, "Learning when to stop thinking and do something!," in *Proceedings of the 26th International Conference on Machine Learning*, 2009, pp. 825–832.

[56] M. Saberian and N. Vasconcelos, "Boosting classifier cascades," in *Advances in Neural Information Processing Systems 23*. 2010, pp. 2047–2055, MIT Press.

[57] D. Benbouzid, R. Busa-Fekete, and B. Kégl, "Fast classification using sparse decision DAGs," in *International Conference on Machine Learning*, June 2012, vol. 29.

[58] V. M. Abazov et al., "Observation of single top-quark production," *Physical Review Letters*, vol. 103, no. 9, 2009.

[59] Aaltonen, T. et. al, "Observation of electroweak single top-quark production," *Phys. Rev. Lett.*, vol. 103, pp. 092002, Aug 2009.

[60] G. Cowan, K. Cranmer, E. Gross, and O. Vitells, "Asymptotic formulae for likelihood-based tests of new physics," *The European Physical Journal C*, vol. 71, pp. 1–19, 2011.

Monte Carlo methods

Rémi Bardenet
Department of Statistics, Oxford University

Abstract

Bayesian inference often requires integrating some function with respect to a posterior distribution. *Monte Carlo* methods are sampling algorithms that allow to compute these integrals numerically when they are not analytically tractable. We review here the basic principles and the most common Monte Carlo algorithms, among which rejection sampling, importance sampling and Monte Carlo Markov chain (MCMC) methods. We give intuition on the theoretical justification of the algorithms as well as practical advice, trying to relate both. We discuss the application of Monte Carlo in experimental physics, and point to landmarks in the literature for the curious reader.

4.1 Introduction

Bayesian statistics (see B. Clément's and D. Sivia's contributions in this volume) quantify the degree of belief one has on quantities or hypotheses of interest, given the data collected. More precisely, let us assume that a model of an experiment is available under the form of a likelihood $p(\mathbf{data}|x)$, and that one has chosen a prior $p(x)$ on the parameters $x \in \mathbb{X} \subset \mathbb{R}^d$ of the experiment. Then all knowledge on x is encoded by the posterior distribution

$$\pi(x) = p(x|\mathbf{data}) = \frac{p(\mathbf{data}|x)p(x)}{\int p(\mathbf{data}|x)p(x)dx}. \tag{4.1}$$

To summarize the inference, one might, for example, want to compute the mean posterior estimate

$$x_{\text{MEP}} = \int x\pi(x)dx$$

and report a credible interval C such that $x_{\text{MEP}} \in C$ and

$$\int_C \pi(x)dx \geq 95\%. \tag{4.2}$$

Both of these tasks require to be able to compute integrals with respect to π. While in some cases these integrals might be analytically tractable, they are usually not in experimental physics, since the likelihood $p(\mathbf{data}|x)$ often takes a complex form, which π inherits, as we shall now see in an example inspired by the Pierre Auger experiment.

4.1.1 A model inspired by Auger

The Pierre Auger observatory[1] is a large-scale particle physics experiment dedicated to the observation of atmospheric showers triggered by cosmic rays. These showers are wide cascades of elementary particles raining on the surface of Earth, resulting from charged nuclei hitting our atmosphere with the highest energies ever seen.

The surface detector of the Pierre Auger experiment (henceforth *Auger*) consists of water-filled tanks and their associated electronics – arranged on a triangular grid, the distance between two tanks being 1.5 kilometers, with the grid covering a total area of 3 000 square kilometers. We model here the tankwise signal produced by one kind of particle in the shower: muons.

When a muon crosses a water tank, it generates Cherenkov photons and photons coming from other processes (e.g., delta rays) along its track at a rate depending on its energy. Some of these photons are captured by photomultipliers. The resulting photoelectrons (PEs) then generate analog signals that are discretized by an analog-to-digital converter.

In this section, we model the integer photoelectron (PE) count vector $\mathbf{n} = (n_1, \ldots, n_M) \in \mathbb{N}^M$ in the M bins of the signal, which means that we omit the model of the electronics. Formally, n_i is the number of PEs in the i-th bin

$$[t_{i-1}, t_i) = [t_0 + (i-1)t_\Delta, t_0 + it_\Delta), \tag{4.3}$$

where t_0 is the absolute starting time of the signal, and $t_\Delta = 25$ns is the signal resolution (size of one bin). The goal is to parametrize the likelihood $p(\mathbf{n}|t, A)$, where t is the arrival time of the muon and A is the integrated signal amplitude.

Given the arrival time t of a muon and the associated total number of PEs A, the PE count in the ith bin is a Poisson variable with parameter

$$\bar{n}_i(t, A) = A \int_{t_{i-1}}^{t_i} r(s-t)ds.$$

where the ideal unit response $r(\cdot)$ is given in Figure 4.1(a), the analytical expression being omitted for the sake of simplicity. Let the number N_μ of muons crossing the tank be known. Adding the contributions

[1] www.auger.org

of N_μ muons with signal amplitudes $\mathbf{A} = (A_1, \ldots, A_{N_\mu})$ and arrival times $\mathbf{t} = (t_1, \ldots, t_{N_\mu})$, the binwise signal expectation is

$$\bar{n}_i(\mathbf{t}, \mathbf{A}) = \sum_{j=1}^{N_\mu} \bar{n}_i(t_j, A_j).$$

Let now $x = (t_1, A_1, \ldots, t_{N_\mu}, A_{N_\mu})$. Our likelihood is finally

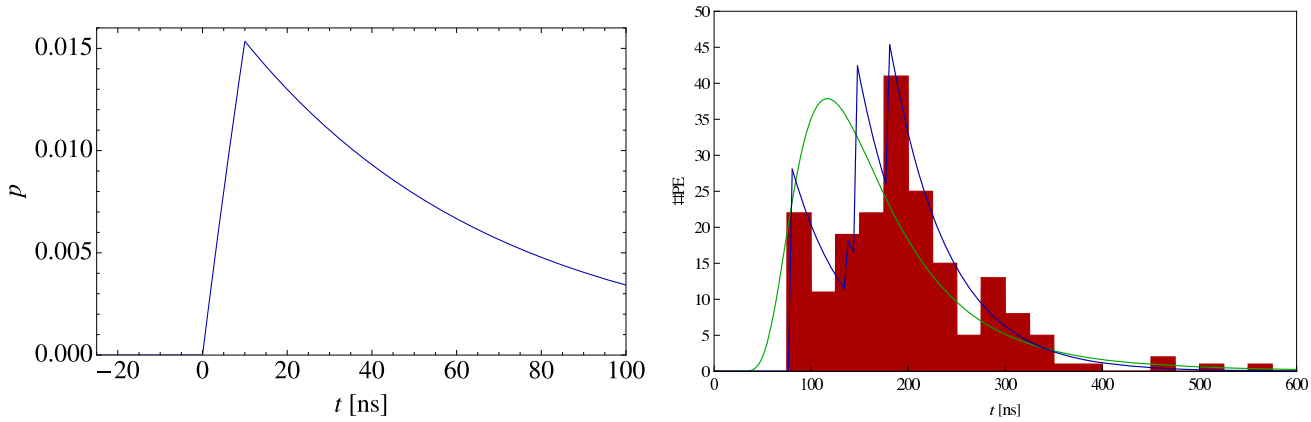
$$p(\mathbf{n}|x) = \prod_{i=1}^{M} \mathrm{Poi}_{\bar{n}_i(\mathbf{t},\mathbf{A})}(n_i). \tag{4.4}$$

A summary of the generating model is depicted in Figure 4.1(b).

This model is only a sketch of what is needed to achieve inference in Auger: in practice, more nuisance variables have to be added, to describe, e.g., the noise in the detector electronics. Still, even with simple flat priors on x, integrals with respect to the resulting posterior

$$\pi(x) \propto p(\mathbf{n}|x)p(x)$$

cannot be analytically computed.



(a) The muonic time response model $r(t)$.

(b) An example signal

Figure 4.1: The generative model of the muonic signal. (a) Ideal unit response function $r(\cdot)$. (b) The green curve is the time-of-arrival distribution $p(t_\mu)$ used to generate this example with $N_\mu = 4$. The amplitude distribution is not shown, but derives from the geometry of the tank. The blue curve is the ideal response $\sum_{j=1}^{4} A_j r(t - t_j)$, and the red histogram is the signal (PE count vector) $\mathbf{n}$.

4.1.2 The Monte Carlo principle

Since integrals like (4.2) cannot be analytically derived, we have to rely on numerical approximation techniques. The cost of classical non-probabilistic numerical integration based on regular grids grows exponentially with the dimension d. The rationale behind Monte Carlo (MC) methods is to replace grids by stochastic samples. Let first

$$\widehat{I}_N = \frac{1}{N} \sum_{i=1}^{N} h(X_i). \tag{4.5}$$

If $X_1, \ldots, X_N \sim \pi$ are independent and identically distributed (i.i.d.), then

$$\widehat{I}_N \approx I = \int h(x)\pi(x)dx. \tag{4.6}$$

Note that $\widehat{I}_n$ in (4.6) is a random variable. Its expectancy is precisely I (we say $\widehat{I}_N$ is an *unbiased estimator* or I) and its variance is V/N, where V is the variance of $h(X)$ when $X \sim \pi$. This justifies the saying

that Monte Carlo error decreases as $\sqrt{N}$.

Interestingly, one can see the MC principle as the randomization of a grid method: points are not regularly spread across the space anymore, but sampled according to π. This is intuitively efficient, since regions of the space should be examined all the more finely that they contribute to the integral I. In other words, putting a fine grid where π is large and a scarce one where π is small will yield to an estimator $\widehat{I}_N$ with small variance.

What makes a good MC method is thus its ability to sample from π. In this tutorial, we describe various ways of sampling according to π, exactly or approximately. Note that the sampling methods we describe are generic and can find other, non-Bayesian applications in experimental physics: simulators like CORSIKA [15] implement sampling from complex, hierarchized distributions with MC methods.

Finally, a generic MC method should require only that π is known *up to a normalization constant*, since in most applications, the denominator of (4.1) is intractable.

The rest of this tutorial is organized as follows. In Section 4.2 we review basic non-MC sampling methods and MC methods that are based on i.i.d. sampling. The latter are useful in small dimensions (say smaller than 10) and often require that π can be somehow approximated. Section 4.3 describes Markov chain Monte Carlo methods (MCMC). MCMC methods generate a dependent sample which asymptotically resembles a sample from π. Section 4.4 presents advanced MCMC tools that learn or exploit the structure of π for better sampling.

4.2 First sampling methods

4.2.1 The inverse cdf method

If the cumulative distribution function F of π is known and can be inverted, then it is enough to know how to sample U from a uniform distribution[2] on $[0, 1]$. Indeed one can show that $F^{-1}(U)$ is then distributed according to π. This can be applied to generate exponential variables, for example. However, this method is not applicable beyond simple distributions, since we usually cannot even compute F, as it requires integrating with respect to π.

4.2.2 The transformation method

It is sometimes possible to obtain samples from $X \sim \pi$ by applying a transformation to variables Y that are easier to sample. For example, building on the exponential generator of Section 4.2.1, we can add two independent exponential variables with parameter 1 to obtain a Gamma variable with parameters $(2, 1)$, as is easily proven by a convolution. Again, this method is limited in its applications to simple distributions.

4.2.3 Rejection sampling

Rejection sampling is one of the simplest MC algorithms. It requires the knowledge of a distribution q on $\mathbb{X}$ which is easy to sample from, such as a Gaussian or a Gamma distribution, and a constant $M > 0$ such that

$$\pi \leq Mq. \tag{4.7}$$

It works by repeatedly sampling according to a *proposal distribution* q and accepting or rejecting each sample with a certain probability that guarantees that the final accepted samples are distributed according to π. The algorithm is presented in Figure 4.2.

Note that the tighter the bound in the right-hand side of (4.7), the less samples are rejected. Thus, a good knowledge of q and M is necessary for rejection sampling to be efficient. When such a bound is not known, one can resort to importance sampling.

[2] We shall assume here that a uniform generator is available, the good old rand() function. However, implementing such a generator is not trivial [18, Section 2.6 and references therein].

```
RejectionSampling(π, q, M, N)
1   S ← ∅,
2   i ← 1.
3   while i ≤ N,
4       Sample x_* ~ q and u ~ U_(0,1).
5       Form the acceptance ratio ρ = π(x_*) / (M q(x_*)).
6       if u < ρ, then
7           S ← S ∪ {x_*},
8           i ← i + 1.
9       else reject.
10  return S.
```

Figure 4.2: The pseudocode of the rejection sampling algorithm. The number of iterations to reach N samples from π is unknown beforehand and depends on the tightness of the bound in (4.7).

4.2.4 Importance sampling

Importance sampling takes as input a proposal q but does not require (4.7), only that q puts mass wherever π does. Furthermore, it only requires that π is known up to a normalization constant. For clarity, we denote by π_0 the available unnormalized version of π, and will write π only for the normalized target distribution.

Unlike other algorithms in this chapter, each of which yields a sample $\mathcal{S}$, importance sampling return a weighted sample or, equivalently, an approximation of the target as weighted sum of point masses. Importance sampling is based on the law of large numbers [13, Section 7.5] with a reweighting trick. Precisely, importance sampling approximates the integral I of (4.6) with the estimator

$$\widetilde{I}_N = \frac{1}{Z}\sum_{i=1}^{N} w_i h(x_i),$$

where

$$w_i = \frac{\pi_0(x_i)}{q(x_i)}, \quad \text{and} \quad Z = \sum_{j=1}^{N} w_j. \tag{4.8}$$

In general, the estimator $\widetilde{I}_N$ is not unbiased, but only asymptotically unbiased. Indeed, applying the law of large numbers to both the numerator and the denominator, we obtain

$$\widetilde{I}_N = \frac{\sum_{i=1}^{N} w_i h(x_i)}{\sum_{j=1}^{N} w_j} \to \frac{\int h(x)\pi_0(x)dx}{\int \pi_0(x)dx} = \int h(x)\pi(x)dx, \quad x_i \sim q \text{ i.i.d.}$$

where the convergence is almost sure[3]. Note, however, that if $\pi_0 = \pi$ is normalized, one can replace Z by N in (4.8) and obtain an unbiased estimator $\widetilde{I}_N$. The pseudocode of the importance sampling algorithm is given in Figure 4.3.

To understand the rôle of the proposal q, it is useful to derive the asymptotic behaviour of the variance of the estimator:

$$\mathrm{Var}(\widetilde{I}_N) = \frac{\sigma^2_{\mathrm{lim}}}{N} + o\left(\frac{1}{N}\right), \tag{4.9}$$

with

$$\sigma^2_{\mathrm{lim}} = \int [h(x) - I]^2 \frac{\pi(x)}{q(x)}\pi(x)dx.$$

[3] There are several modes of convergence for sequences of random variables, cf. [13, Section 7.2] for a summary.

Thus, in order to keep the variance of $\widetilde{I}_N$ – in physical terms the square of the statistical error – low, q has to be chosen close to π, and with heavier tails than π. The last requirement means that we must ensure that

$$\sup_{x \in \mathbb{X}} \frac{\pi(x)}{q(x)} < \infty.$$

ImportanceSampling(π_0, q, N)	
1	Sample independent $x_i \sim q, i = 1, \ldots, N$,
2	Form the weights $w_i = \frac{\pi_0(x_i)}{q(x_i)}$,
3	Compute the normalization constant $Z = \sum_{i=1}^N w_i$,
4	π is approximated by $\frac{1}{Z} \sum_{i=1}^N w_i \delta_{x_i}$.

Figure 4.3: The pseudocode of the importance sampling algorithm only requires that π is known up to a normalization constant. Unlike rejection sampling, no sample is wasted.

On the choice of the proposal for importance sampling

In practice, either a reasonable choice for q is available, or not. The first case occurs when, e.g., π is almost unimodal and concentrates its mass on a small region of $\mathbb{X}$. A Gaussian centered at this small region with a reasonable variance then yields a good choice for q. Easy-to-sample, heavy-tailed distributions like Student's distribution, are also handy. We have often seen the case in particle physics where π is a posterior that puts all its mass on a small region of $\mathbb{R}^d$. In that case, remember that importance sampling with the right q yields better accuracy than grid-based methods or uniform sampling.

If the choice of q is not obvious, we recommend the use of an adaptive strategy, such as population Monte Carlo. A description of population MC and an application to model selection in cosmology can be found in [24]. Basically, first make a wild guess $q^{(0)}$ for q, say a Gaussian with a large variance. Apply importance sampling a first time to obtain an estimate of π and fit a Gaussian $q^{(1)}$ to this estimate of π. Now re-apply importance sampling with $q^{(1)}$ as a proposal, and re-fit a new Gaussian $q^{(2)}$ to π, etc. After T iterations, $q^{(T)}$ should be a good proposal distribution for importance sampling. Of course, you can apply this procedure with other candidate proposals than Gaussians, you should indeed choose a family of distributions among which you think you may find a good approximation of π. If you have reasons to believe that π is bimodal, for example, you should probably fit a mixture of two distributions as in [24] rather than a Gaussian, which is unimodal. Usually, with the right choice of family of distributions, a few iterations are enough to get a reasonable q, and you can stop when $q^{(t)}$ does not change a lot with t.

A similar method for adaptively tuning the proposal in importance sampling has been quite popular in physics: nested sampling, on which we recommend [22]. Be careful, however, a common mistake is to forget the assumption that the final q has to put mass wherever π may.

Convergence diagnostics and confidence intervals

There are a variety of criteria to assess the good behaviour of an importance sampling estimator. An important thing to check is the empirical distribution of the weights w_i. If only a few of the weights are nonzero, the estimator $\widetilde{I}_N$ is based on too few points and thus has a large variance. It is thus desirable to obtain a weight distribution with a high number of large and comparable weights, which, again, is achieved by finding a good proposal q.

More formal quality measures also exist, such as the so-called *effective sample-size* ESS_N:

$$\mathrm{ESS}_N = \left(\sum_{i=1}^N \left(\frac{w_i}{\sum_{j=1}^N w_j} \right)^2 \right)^{-1}.$$

ESS_N ranges from 1 (when only one weight is nonzero) to N (when all weights are equal). Roughly, ESS_N

is telling how many of the samples $x_1, \ldots, x_N$ are really independent in the following sense: the accuracy of $\widetilde{I}_N$ is equivalent to the accuracy obtained with ESS_N samples that would be drawn directly from the real π.

Finally, it is possible to derive asymptotic confidence intervals for $\widetilde{I}_N - I$, since it can be first shown that

$$\sqrt{N}(\widetilde{I}_N - I) \to \mathcal{N}(0, \sigma^2_{\mathrm{lim}}),$$

where the convergence is in distribution, and second

$$N \sum_{i=1}^{N} \left(\frac{w_i}{\sum_{j=1}^{N} w_j} \right)^2 \left(h(X_i) - \sum_{k=1}^{N} \frac{w_k}{\sum_{j=1}^{N} w_j} h(X_k) \right)^2 \to \sigma^2_{\mathrm{lim}}$$

almost surely.

4.2.5 Going to higher dimensions

We now consider an insightful example on how rejection and importance sampling scale when the dimension d of the ambient space grows. Consider a simple unit Gaussian target $\pi = \mathcal{N}(0, I_d)$, where I_d is the $d \times d$ identity matrix. Say we are fortunate enough to know that π is an isotropic Gaussian, but ignore its variance. A relevant choice of proposal would then be an isotropic Gaussian $q(x) = \mathcal{N}(0, \sigma^2 I_d)$.

If applying rejection sampling, one must know *beforehand* that π has variance upper bounded by some constant σ^*, and then choose $\sigma \geq \sigma^*$, in order to satisfy (4.7). This is already a very strong assumption, but there is worse: the fraction of accepted samples goes as σ^{-d}. This means that if σ is not *exactly* 1, one should expect an exponentially small number of accepted samples with growing d. A similar *curse of dimensionality* happens with importance sampling: the variance of the weights is either infinite if $\sigma \leq \frac{1}{\sqrt{2}}$, or it goes as σ^d.

4.2.6 Conclusion on rejection and importance sampling

Rejection sampling is very easy to implement and can work very well in settings where the relevant information on the target is known, as is sometimes the case in simulators like CORSIKA [15]. Importance sampling is more generic and can deal with unnormalized targets, as is often the case in Bayesian data analysis. The efficiency of both methods depends on the design of the proposal distribution q, and both do not scale to high dimensions (say larger than 10).

4.3 MCMC basics

Rejection and importance sampling are Monte Carlo methods based on an i.i.d. sample from a proposal distribution q. To tackle large dimensions, other methods have been devised that are based on a non-independent sampling: Markov chain Monte Carlo methods (MCMC). The prototype of MCMC methods is the Metropolis-Hastings algorithm, of which almost all MCMC algorithms are variants. Note that although we concentrate here on applications to inference, MCMC is also used in simulators like CORSIKA, see [8] for an example.

4.3.1 The Metropolis-Hastings algorithm

We first describe Metropolis' algorithm in Figure 4.4. It builds a random walk (X_i) that explores $\mathbb{X}$, and eventually approximates independent samples from π.

Metropolis' algorithm is a **for** loop. At each iteration t, a candidate point x^* is proposed in the neighborhood of the current position x_{t-1}, according to a proposal $q(\cdot|x_{t-1})$. In Metropolis' algorithm, this proposal is assumed to be symmetric, i.e., $q(x|y) = q(y|x)$. In practice, a Gaussian with fixed covariance matrix Σ is often used: $q(y|x) = \mathcal{N}(y|x, \Sigma)$. After the candidate point has been generated, it is accepted as the next position x_t only with a certain probability ρ, which is 1 if $\pi(x^*)$ is larger than $\pi(x_{t-1})$, and smaller than 1 (but often not zero!) if not. This precise definition of ρ relates the random

MetropolisSampler(π_0, q, T, x_0)

1 $\mathcal{S} \leftarrow \emptyset$.
2 **for** $t \leftarrow 1$ **to** T,
3 Sample $x_* \sim q(.|x_{t-1})$ and $u \sim \mathcal{U}_{(0,1)}$.
4 Form the acceptance ratio

$$\rho = \min\left(1, \frac{\pi_0(x_*)}{\pi_0(x_{t-1})}\right).$$

5 **if** $u < \rho$, **then** $x_t \leftarrow x_*$ **else** $x_t \leftarrow x_{t-1}$.
6 $\mathcal{S} \leftarrow \mathcal{S} \cup \{x_t\}$.
7 **return** $\mathcal{S}$.

Figure 4.4: The pseudocode of the Metropolis algorithm.

MetropolisHastingsSampler(π_0, q, T, x_0)

1 $\mathcal{S} \leftarrow \emptyset$.
2 **for** $t \leftarrow 1$ **to** T,
3 Sample $x_* \sim q(.|x_{t-1})$ and $u \sim \mathcal{U}_{(0,1)}$.
4 Form the acceptance ratio

$$\rho = \min\left(1, \frac{\pi_0(x_*)}{q(x_*|x_{t-1})} \frac{q(x_{t-1}|x_*)}{\pi_0(x_{t-1})}\right).$$

5 **if** $u < \rho$, **then** $x_t \leftarrow x_*$ **else** $x_t \leftarrow x_{t-1}$.
6 $\mathcal{S} \leftarrow \mathcal{S} \cup \{x_t\}$.
7 **return** $\mathcal{S}$.

Figure 4.5: The pseudocode of the Metropolis-Hastings algorithm.

walk (X_i) to π and makes the algorithm different from an optimization algorithm: it does not always try to move for a point with larger π. Furthermore, the theory of Markov chains[4] guarantees that such an acceptance rule implies that π is the limiting distribution of the chain (X_i), in a sense that shall become clear soon.

Now we are ready to present the Metropolis-Hastings (MH) algorithm. It is simply Metropolis' algorithm, but with general, possibly nonsymmetric proposals. The pseudocode of MH is given in Figure 4.5. Note the new definition of the acceptance probability ρ, which ensures the final convergence to π. Intuitively, this acceptance rule cancels the influence of q on the limiting distribution: the probability of accepting a move that q is likely to draw often is reduced.

We refer the reader interested by a gentle introduction to theoretical results on MH to [18, Chapter 7]. We will limit ourselves to one simple but useful result, since it covers the common Metropolis algorithm with Gaussian proposals: assume $q(\cdot|x)$ puts mass over all $\mathbb{X}$ and that there exists $\delta > 0$ and $\varepsilon > 0$ such that

$$\|x - y\| < \delta \Rightarrow q(y|x) > \varepsilon,$$

then for any integrable function h, MH gives an asymptotically unbiased estimate of the integral I defined

[4] A Markov chain is a sequence of random variables (X_i) such that X_{i+1} depends on the past only through X_i. More formally: $X_{i+1}|X_1, \ldots, X_i \sim X_{i+1}|X_i$.

in (4.6):

$$\lim_{N\to\infty} \widehat{I}_N \to I.$$

This is an example of formal result that states that (X_i) behaves like independent draws from π for large i.

4.3.2 Assessing convergence

MH outputs a sample from a Markov chain that asymptotically approximates independent draws from π. From a practitioner's point of view, two questions arise, which we address successively in the rest of this section. First, since we are waiting for the chain (X_i) to converge to π whatever starting value x_0 we input, it is important to know from which iteration on the chain is independent from x_0. Second, even if the iteration number is big enough that the chain has "forgotten" about x_0, a Markov chain is not a series of independent draws, and it is relevant to ask whether our MCMC chain has reached a good approximation of independence, or, in other words, how much the variance of the estimator $\widehat{I}_N$ suffers from the statistical dependence among the X_is.

Has the chain forgotten its starting state?

Although theory on this question is unsatisfying as of today, experimental techniques exist, which help the practitioner assess his chain has converged. The first thing one might try is to launch in parallel many chains with different starting points, and check they all give similar results. Besides traceplots, one can plot an online estimate of the mean:

$$\frac{1}{n}\sum_{i=0}^{n} X_i$$

versus $n = 1, \ldots, N$ for all chains and check they all converge towards the same value. If not, then convergence has certainly not been reached. Online estimates of the variance of each chain, of quantiles, etc. can also be useful to plot. There are a number of statistics of the sample that formalize this principle of comparison between many chains. A popular such convergence assessment is known as the Gelman-Rubin diagnosis [18, Section 12.3.4]. See [18, Chapter 12] for a review of other convergence diagnostics.

To cancel the influence of the starting point in the evaluation of $\widehat{I}_N$, it is usually advised to discard the first B samples of the chain and replace $\widehat{I}_N$ by

$$\widehat{I}_{N,B} = \frac{1}{N-B+1}\sum_{i=B}^{N} X_i.$$

The discarded B samples are called a *burn-in* sample. Though reducing initialization bias, discarding the first B samples also usually makes the variance of $\widehat{I}_{N,B}$ larger than the variance of $\widehat{I}_N$, and so B should be as small as possible to keep the final statistical error low. The choice of an optimal B is an open question. In practice, our personal take is to keep the burn-in below 25% of the sample, and simply go for a large enough number of samples N that multiple chains with different initializations give similar answers.

How independent do the samples look?

After initialization bias, the second convergence issue is that of the independence of the samples. Identifying an MCMC chain converging to π to a dynamical system progressively stabilizing at equilibrium, Sokal speaks here of *autocorrelation in equilibrium* [23]. This is related to the variance of $\widehat{I}_N$ in the following way: the more correlation there is between samples (X_i) (the *autocorrelation* of the chain), the higher the variance of $\widehat{I}_N$. The variance of $\widehat{I}_N$ will still decrease in K/N, but the constant K might be much larger than in the independent case [23].

Again, theoretical answers to this question are not very satisfying as of today, but practical diagnostics exist. Besides plotting the different components of the chain (X_i) versus i and checking independence,

the simplest idea is to plot the autocorrelation function of the chain. If $d = 1$, it is defined as

$$\rho(t) = \frac{C(t)}{C(0)}, \text{ where } C(t) = \frac{1}{N-t+1} \sum_{i=0}^{N-t} (X_i - \overline{X})(X_{i+t} - \overline{X}), \tag{4.10}$$

where $\overline{X} = N^{-1} \sum_{i=1}^{N} X_i$. If $d > 1$ one considers the autocorrelation in each component. The autocorrelation function for $t \geq 0$ should look like a rapidly decreasing exponential starting at 1 and going to 0, as in Figure 4.7(b). If not, then one can *thin* the chain, i.e., keep only one sample every other 10 or higher if necessary. But while this may lead to more independent samples, it also leads to a waste of computational effort and an increase in the variance of the final estimator. If strong autocorrelation is revealed, we recommend to start all over with a different q. Indeed, strong autocorrelation often reveals a bad choice in the proposal. Finally, note that if one is only interested in estimating $\int h(x)\pi(x)dx$ for a single h, then one should monitor the autocorrelation function of $h(X)$ rather than X, which is obtained by replacing each occurrence of X in (4.10) by $h(X)$.

Tuning the proposal distribution of MH

Consider the Metropolis algorithm of Figure 4.4. If q proposes only small steps, then the candidates x^* will often be accepted since the ratio of the posteriors in Step 4 of Figure 4.5 will be close to 1. This will lead to high acceptance but also high autocorrelation: X_{i+1} is almost always in the neighborhood of X_i, and the chain needs a lot of iterations to cross the space, see Figure 4.6(a) for a typical traceplot. On the other hand, if q proposes too big steps, then it is likely that they will be rejected, especially if the current state of the chain has a high value of π. This will lead to the chain being blocked for several iterations, which is an even worse case of autocorrelation, see Figure 4.6(b) for a typical traceplot. There is thus a compromise to find between small steps and large acceptance rate, and large steps and small acceptance rate. Theory suggests that when the target and the proposals are Gaussian, the acceptance rate to reach minimum variance of $\widehat{I}_N$ is approximately 0.5 when $d \leq 2$ and 0.25 otherwise [19, 20]. In the common case where the proposal is Gaussian, practitioners usually proceed as follows: take a proposal with some free stepsize parameter σ:

$$q(y|x) = \mathcal{N}(y|x, \sigma\Sigma_0), \tag{4.11}$$

launch several preliminary runs with different values for σ, and finally select the value of σ that yielded the desired acceptance rate for the final run.

At this stage, we personally advocate the use of a *pseudoadaptive* strategy: *adaptive* because it learns a good σ along the run, and *pseudo* because we stop the adaptation before the end of the burn-in[5]. While $t \leq B'$, where B' is smaller than the burn-in length B, we advise to adapt σ in the following manner: at each iteration t, replace $\log(\sigma)$ by

$$\log(\sigma) + \frac{1}{t^{0.7}}(\alpha_t - \alpha^*) \tag{4.12}$$

where $\alpha_t \in [0, 1]$ is the current acceptance rate (the number of accepted samples so far divided by t), α^* is 0.5 if $d \leq 2$ and 0.25 else. The log transformation guarantees that σ remains positive. The rationale behind (4.12) is that if $\alpha_t > \alpha^*$, then too many steps are accepted, so that the stepsize should be increased, and vice versa. (4.12) with a large enough B' is simply an automatic way to perform the preliminary search for σ.

Asymptotic confidence intervals for $\widehat{I}_N$

After having obtained a sample with good properties as discussed previously, we are interested in deriving a confidence interval for $\widehat{I}_N - I$. Note that this is not the same as finding a credible interval as in Figure 4.7(d): we are here interested in knowing how $\widehat{I}_N$ varies when the full sample $(X_1, ..., X_N)$ varies, being drawn from the same chain. It is a tougher and more open question than with importance sampling.

[5] In Section 4.4.1, we discuss a *fully adaptive* algorithm where a similar adaptation is carried out to the end of the run.

First, we need to check that the chain satisfies a central limit theorem (CLT):

$$\sqrt{N}(\widehat{I}_N - I) \to \mathcal{N}(0, \sigma^2_{\text{lim}})$$

where the convergence is in distribution. A discussion on the CLT for Markov chains can be found in [18, Section 6.7.2], with a note on MH in [18, Section 7.8.2]. For our needs, we just state that a CLT holds for Metropolis' algorithm with Gaussian proposals and a target with bounded support, see [16, Theorem 4.1]. Now, when a CLT holds, confidence intervals can be built using proper[6] approximations of σ^2_{lim} [10]. These estimators of σ^2_{lim} are not difficult to understand, but their description and the conditions for convergence are fairly technical, and we refer the interested reader to [Theorems 1 and 2][10] for recent results on the so-called *spectral* and *overlapping batch means* method, our two favourites. A less technical but not up-to-date spectral method is described in [23, Section 3].

4.3.3 Implementation tips

First, MCMC can deal with constrained variables. If the variable is discrete, it is usually easy to find a proposal with the right constrained support. Constrained continuous variables are to be treated differently. One could, for example, include an indicator in the prior that will yield rejection of all points outside the allowed region. But if π is large near the boundary of this allowed region, it is likely that the chain will spend time there and thus a lot of points will be rejected only because they do not satisfy the constraint. This is a waste of computational effort, and, if possible, it is usually advised to reparametrize the problem so that it becomes unbounded. If x has to remain positive, then use $\log(x)$. If x has to remain in an interval, then use Argth, for example.

Second, as with most multivariate methods, it is usually better to scale one's variables. The rough idea is that a step in every direction should have the same effect on π. This should make Gaussians with covariance proportional to the identity reasonable proposals.

Third, whenever manipulating likelihoods, it is advisable to work in the log domain. Computing log-likelihoods is numerically more stable than doing large products of potentially very small numbers, and MCMC code can always be written using only log-likelihoods.

4.3.4 A worked out example

Consider the model of Section 4.1.1 with a single muon entering the tank: $N_\mu = 1$. We simulated data with an arrival time $t_{\text{true}} = 55$ and an amplitude $A_{\text{true}} = 20$. The simulated signal $\mathbf{n}$ is depicted in Figure 4.7(a).

We place a wide independent Gaussian prior on t and A: $p(t, A) = \mathcal{N}(t|100, 100^2)\mathcal{N}(A|20, 20^2)$. Let us apply MH to obtain the posterior $p(t, A|\mathbf{n})$. First, let us change variables for $\log(t)$ and $\log(A)$, to avoid dealing with a boundary in $\mathbb{R}^2$. $\log(t)$ and $\log(A)$ roughly have the same scale, so we do not modify them further. Let $T = 20\,000$ iterations, of which we discard the first $B = 5\,000$ as burn-in. We take a Gaussian with covariance $\sigma^2 I$ as a proposal and apply the pseudoadaptive tuning of (4.12).

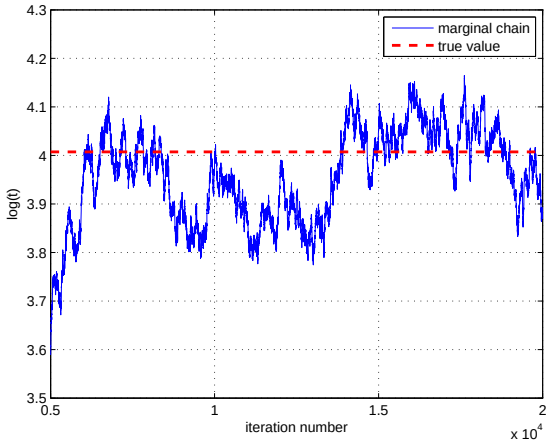
The results[7] can be seen in Figures 4.7(a) to 4.7(e). The autocorrelation function in Figure 4.7(b) indicates a fairly independent behaviour of the chain. This is confirmed by the traceplot in Figure 4.7(c) where no clear dependence can be detected by eye.

To obtain marginal distributions of π, one can simply collect the corresponding coordinate among the X_is. The marginal histograms here look regular, as shown by the $\log(t)$ marginal in Figure 4.7(d). The acceptance has approximately reached the optimal 50% by the end of the burn-in and stabilized there. To check independence from the starting point, we ran 10 chains with uniform initialization on a wide rectangle, and plotted the online sample mean and standard deviations of each chain in Figure 4.7(f): all chains are consistent.

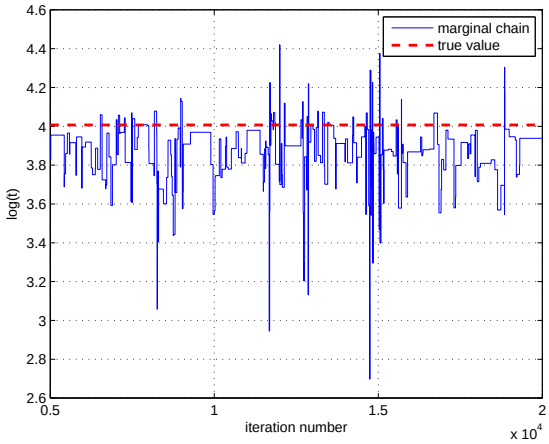
When reporting a result, the best practice is to give a summary of and make available the entire posterior sample. When $\mathbb{X}$ is high-dimensional, a series of marginal histograms, or 2D plots of one component versus the other, etc. can be given. In terms of estimation, credible intervals with level c can

[6] By *proper*, we mean a sequence $\widehat{\sigma}_N$ that converges to σ_{lim} at least in probability.

[7] The matlab code used to generate the figures in this section is available on `www.stats.ox.ac.uk/~bardenet`.



(a) Steps too small, acceptance too high



(b) Steps too big, acceptance too low

Figure 4.6: Two examples high autocorrelation with different causes.

be computed easily once the sample has been drawn: simply report any interval that contains a proportion c of the samples. Such an interval is computed and given in Figure 4.7(d), for example.

Finally, we give here two highly correlated traceplot examples of badly tuned chains that should ring an alarm, and be compared to the correct behaviour in Figure 4.7(c). On Figure 4.6(a), σ is too small, leading to an overly slow exploration of the parameter space. On Figure 4.6(b), σ is too large, causing the chain to stay blocked very often.

4.3.5 Physics-inspired variants of MH: Gibbs, Langevin and Hamilton

MH with a simple unimodal proposal (Gaussian, Student) is very generally applicable, but can be enhanced through the use of additional information on the target distribution π. Although, in our experience, they are rarely used in experimental physics due to the complexity of the models, we quickly review here some popular variants of MH that were actually inspired by physics.

The Gibbs sampler

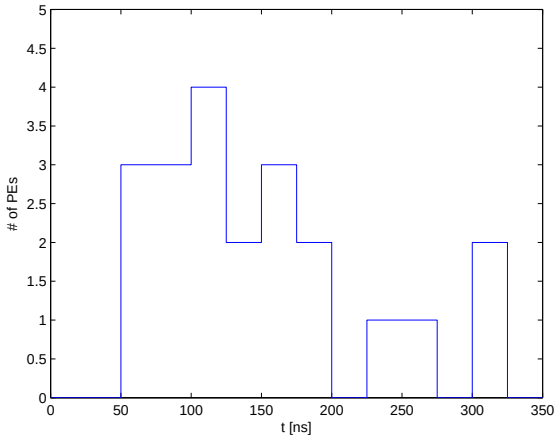
For $x = (x^1, \ldots, x^d) \in \mathbb{X}$ a vector[8] of length d, define $x^{-i} = (x^1, \ldots, x^{i-1}, x^{i+1}, \ldots x^d)$. If the posterior is simple enough that it is easy to sample from all conditional distributions $\pi(x^i|x^{-i})$, then one can replace Step 3 of MH in Figure 4.5 by a sequence of draws from the conditionals, conditioning on the components already drawn. The acceptance probability ρ in Step 4 in Figure 4.5 then evaluates to 1 and every proposal is thus accepted. The pseudocode of the Gibbs sampler is given in Figure 4.8.

The Gibbs sampler is useful in models where the dependencies between variables are non-trivial, but the conditional distributions are easy to sample. Even if this is rarely the case in particle physics, it can happen that some conditional distributions are available, and one might then use Gibbs proposals on some of the variables within an MH scheme. Any concatenation of MCMC steps is permitted, as long as the same concatenation is repeated over and over.

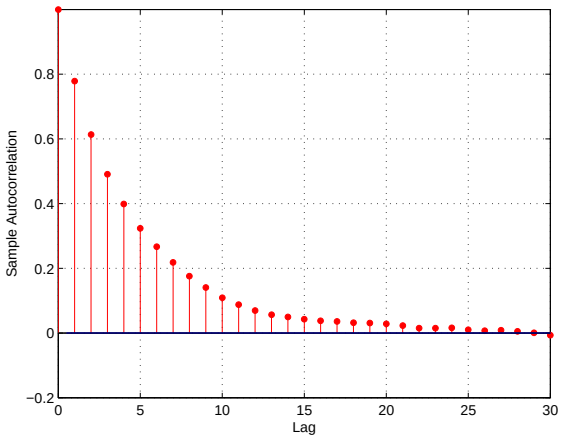
Langevin diffusion

If not the conditionals, one might know how to compute the gradient of π. This is also useful information: although MH is not an optimization algorithm, visiting all modes of π is essential. The Langevin sampler is an MH sampler with a proposal that is partly deterministic: from x_{t-1}, a small step in the direction of the gradient of π at x_{t-1} is done, and from there a small Gaussian step is performed. Basically, the

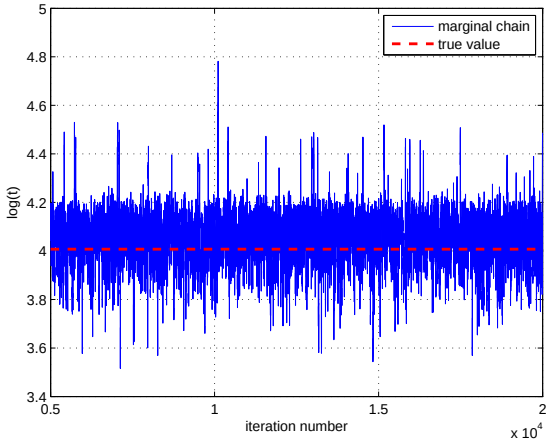
[8] We use here upper indices to number components of the vector, and keep lower indices for the iteration number in MH



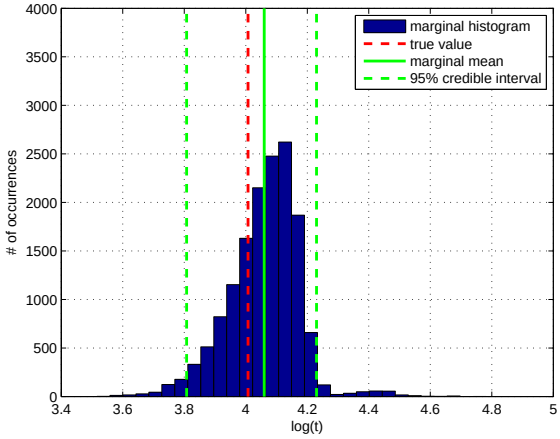
(a) Generated signal



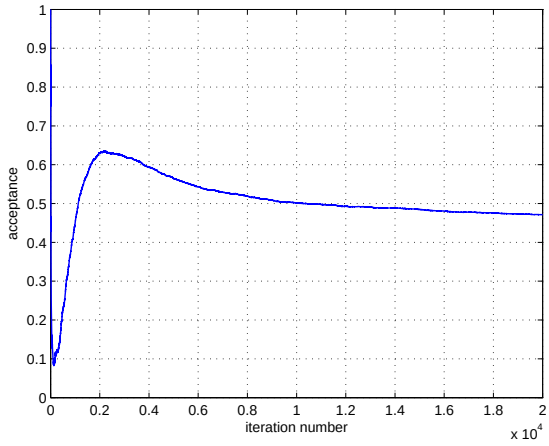
(b) Autocorrelation function



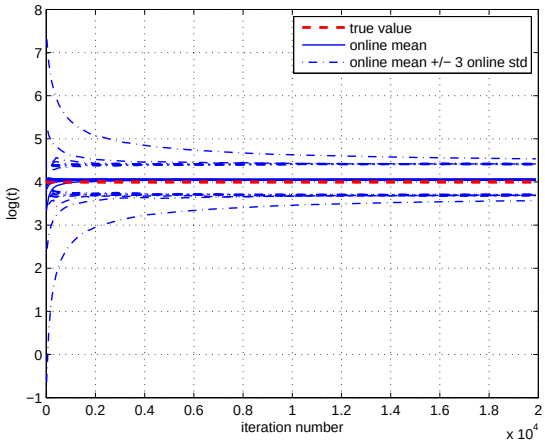
(c) $\log(t)$ chain



(d) $\log(t)$ histogram



(e) Acceptance



(f) Statistics of 10 independent chains

Figure 4.7: Results of applying MH in the setup of Section 4.3.4. See text for details.

```
GibbsSampler(π(x^i|x^{-i}) ∀i, T, x_0)
    S ← ∅.
    for t ← 1 to T,
        Sample x*_1 ~ π(x^1|x^{-1}_{t-1}),
        Sample x*_2 ~ π(x^2|x^1_*, x^3_{t-1}, ..., x^d_{t-1}),
            ⋮
        Sample x*_d ~ π(x^d|x^1_*, ..., x^{d-1}_*, x^d_{t-1}).
        Set x_t ← x_*.
        S ← S ∪ {x_t}.
    return S
```

Figure 4.8: The pseudocode of the Gibbs sampler.

Langevin sampler is MH with proposal

$$q(y|x) = \mathcal{N}(y|x + \frac{\sigma^2}{2}\nabla \log \pi(x), \sigma^2 I_d).$$

The Langevin sampler is actually inspired by the discretization of a diffusion equation [18, Section 7.8.5 and references therein]. Interestingly, the MH acceptance step can here be interpreted as a correction for the discretization error of the numerical scheme solving the diffusion equation.

Another popular sampler, inspired by mechanics, is Hamiltonian MCMC, or hybrid MCMC [17]. Its main features are that the target is plugged in an energy function, points in $\mathbb{X}$ are interpreted as space coordinates and augmented with an artificial momentum variable, and proposals include a deterministic move along an approximate solution to a system of differential equations, as well as an MH correction step. Hamiltonian MCMC in its original form also requires a closed form for the gradient, which is again rarely available. Details of the pseudocode of Hamiltonian MCMC are out of the scope of this chapter, but we refer interested readers to [17].

4.4 Advanced MCMC methods

Motivated by the Auger-inspired example of Section 4.1.1, we review in this section some useful advanced MCMC algorithms. Consider first the case where at least two muons entered the tank: $N_\mu > 1$. If two muons crossed the tank around the same time, the corresponding amplitude variables A_1 and A_2 are likely to be anticorrelated under the posterior: if A_1 is large and the first muon explains most of the signal, then A_2 should be small, and vice versa. Thus, having an MH proposal that detects and uses this correlation, proposing large "diagonal" jumps in the (A_1, A_2) subspace, would be more efficient than an independent proposal. In Section 4.4.1, we present an MCMC algorithm that learns its proposal covariance on the fly. Another difficulty one encounters with the model of Section 4.1.1 is that in practice N_μ itself is unknown and should be inferred. In Section 4.4.3 we present an MCMC algorithm that makes proposals across models with different numbers of muons. Finally, the likelihood (4.4) is invariant to the ordering of the muons. This has undesirable effects on marginal inference that MCMC can cope with, as presented in Section 4.4.2.

4.4.1 Adaptive MCMC

We already mentioned a pseudoadaptive update scheme in (4.12), where the stepsize of the MH proposal was tuned during a finite number of iterations before being frozen for the rest of the run. Now a legitimate question is whether the asymptotic guarantees on MCMC hold if we let the adaptation run forever?

Adaptive MCMC is a research topic that focuses on designing provably valid MCMC algorithms with

proposals that are tuned on the fly until the complete end of the run. The literature is dense, and we single out here one adaptive MCMC algorithm that we use in almost every physics application: the adaptive Metropolis algorithm (AM; [14]).

The pseudocode of AM is given in Figure 4.9. Note that in practice, it may help to wait for say $10d$ iterations before using the adapted covariance matrix in the proposal. β should be taken in $(\frac{1}{2}, 1]$, 1 meaning freezing the covariance Σ_t faster. By default, we personally use 0.7. In the literature, the covariance matrix scale factor, or stepsize, σ is often set to $(2.38)^2/d$, as it is shown in [19] that this stepsize is optimal in a sense for Gaussian proposals and targets. However, in practice, we recommend the use of an adaptive scheme such as Step 9 in Figure 4.9, which preserves the convergence of AM. Other adaptive scalings are discussed in [2].

Remark that AM is still called an MCMC algorithm, although the chain is not Markov anymore: given X_{t-1}, X_t is still dependent on the previous history of the chain through Σ_{t-1}. Still, AM was proven to provide an asymptotically unbiased estimator $\widehat{I}_N$ [1].

AdaptiveMetropolisSampler$(\pi_0, \mu_0, \Sigma_0, \beta \in (\frac{1}{2}, 1], \sigma_0, T, x_0)$

1 $\mathcal{S} \leftarrow \emptyset$.
2 **for** $t \leftarrow 1$ **to** T,
3 Sample $x_* \sim \mathcal{N}(.|x_{t-1}, \sigma_{t-1}\Sigma_{t-1})$ and $u \sim \mathcal{U}_{(0,1)}$.
4 Form the acceptance ratio

$$\rho = \min\left(1, \frac{\pi_0(x_*)}{\pi_0(x_{t-1})}\right).$$

5 **if** $u < \rho$, **then** $x_t \leftarrow x_*$ **else** $x_t \leftarrow x_{t-1}$.
6 $\mathcal{S} \leftarrow \mathcal{S} \cup \{x_t\}$.
7 $\mu_t \leftarrow \mu_{t-1} + \frac{1}{t^\beta}\left(X_t - \mu_{t-1}\right)$
8 $\Sigma_t \leftarrow \Sigma_{t-1} + \frac{1}{t^\beta}\left(\left(X_t - \mu_{t-1}\right)\left(X_t - \mu_{t-1}\right)^T - \Sigma_{t-1}\right)$
9 $\log(\sigma_t) \leftarrow \log(\sigma_{t-1}) + \frac{1}{t^\beta}(\rho - \alpha^*)$
10 **return** $\mathcal{S}$.

Figure 4.9: The pseudocode of the adaptive Metropolis algorithm. Modifications with respect to the Metropolis algorithm in Figure 4.4 are Steps 7 to 9 (in blue). The setting of free parameters β and c is discussed in the main text. When $d \leq 2$, we usually set α^* to 0.5, and 0.25 else, but this is only a rule of thumb.

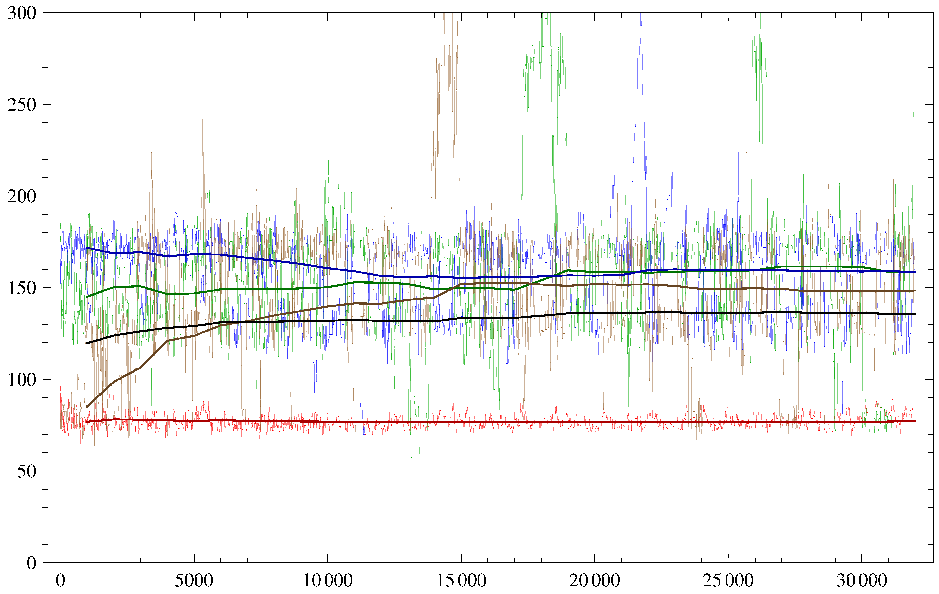
4.4.2 Label switching

Another feature of the likelihood (4.4) is that it is invariant to permutations of the muons. In the case where $N_\mu = 2$, for example,

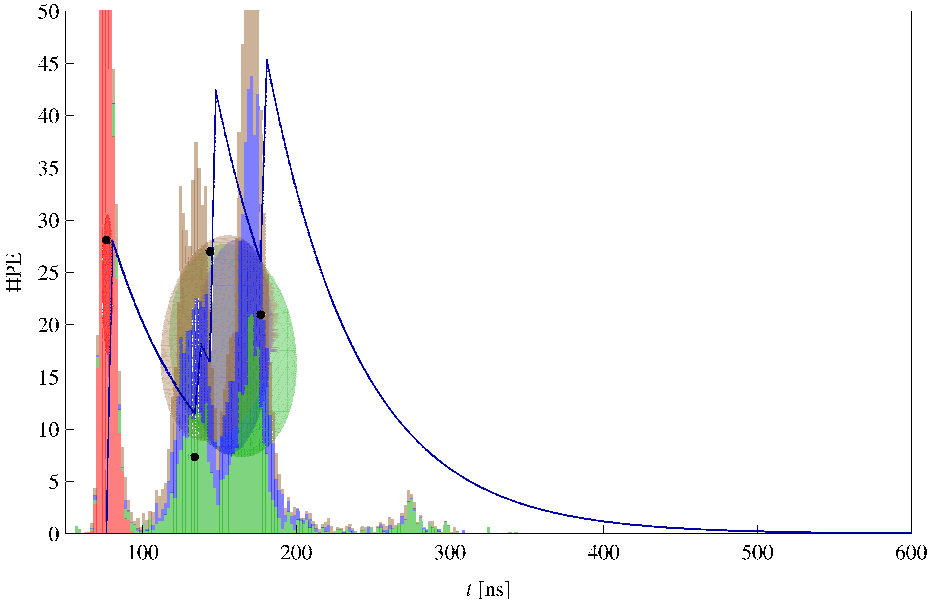
$$p(\mathbf{n}|t_1, A_1, t_2, A_2) = p(\mathbf{n}|t_2, A_2, t_1, A_1).$$

If the prior is also invariant, then the posterior π inherits the same property. π has then as many redundant modes as there are permutations to which it is invariant, and this is undesirable when it comes to marginal inference. Indeed, Figures 4.10(a) and 4.10(b) illustrate the challenges when running vanilla AM on the example presented in Figure 4.7(a). The red variable gets stuck in one of the mixture components, whereas the blue, green, and brown variables visit all the three remaining components, a phenomenon called *label switching*. As a result, marginal estimates computed for the blue, green, and brown variables are then mostly identical as seen on Figure 4.10(b). In addition, the shaded ellipses, depicting the marginal posterior covariances of the two parameters t and A of each muon, indicate that the resulting empirical covariance estimate is very broad, resulting in poor efficiency of the adaptive algorithm. Label switching is addressed by so-called *relabeling* algorithms [5, Chapter 8]. A recent relabeling mechanism

that interweaves favorably with AM can be found in [6], along with an application to the Auger model of Section 4.1.1.



(a) AM: component chains and means



(b) AM: component posteriors

Figure 4.10: The results of AM on the signal example of Figure 4.1(b). (a) Three of the four t chains switch position constantly as a result of the target π being invariant to permutations of the muons. The corresponding running means (thick lines) converge to similar values. For reference, the thick black line depicts the mean of the coloured thick lines. (b) Label switching causes artificially multimodal marginals. Black dots: the x-coordinates are the real time-of-arrival parameters t, and the y-coordinates are proportional to the amplitudes A. Colored ellipses are exp(1/2)-level sets of Gaussian distributions: the means are the Bayesian estimates of (t, A) for each muon, and the covariance is the marginal posterior covariance of each (t, A) couple.

4.4.3 Transdimensional problems

Until now, we have always considered N_μ fixed and known. Now consider the problem of letting N_μ free and trying to estimate it. This problem is termed *model selection* in statistics, and various Bayesian

answers have been given. One full MCMC solution is called reversible jump MCMC (RJMCMC) and is due to Green [12]. We will introduce the algorithm through an example here, the generic description of the method and implementation advice can be found in [18, Section 11.2].

Say we know there were $1 \leq N_\mu \leq 2$ muons in the tank from some other measurements, and we would like to infer N_μ as well as the corresponding times and amplitudes. Let the prior on N_μ be

$$p(N_\mu = 1) = p(N_\mu = 2) = \frac{1}{2}.$$

We need a chain that targets

$$\pi(x) \propto p(\mathbf{n}|N_\mu, t_1, A_1, \ldots, t_{N\mu}, A_{N_\mu}) p(t_1, A_1, \ldots, t_{N\mu}, A_{N_\mu}|N_\mu) p(N_\mu)$$

and that can take values such as $(1, t, A)$ and $(2, t_1, A_1, t_2, A_2)$. In other words, the state space of the chain would be

$$\{1\} \times \mathbb{R}^2 \cup \{2\} \times \mathbb{R}^4.$$

This chain should be able to jump *within models*, i.e., from $(1, t, A)$ to $(1, t', A')$ or from $(2, t_1, A_1, t_2, A_2)$ to some other point $(2, t_1', A_1', t_2', A_2')$ with two muons. The chain should also be able to jump *across models*, i.e., from $(1, t, A)$ to some point $(2, t_1', A_1', t_2', A_2')$ and vice versa. For jumps within the same model, we implement usual MH moves. The main contribution of RJMCMC is a general rule to build jumps across models and the corresponding acceptance probability. The key is to design jumps across models that are likely to be accepted. In our example, to jump from a point $(N_\mu = 1, t, A)$ to $(N_\mu = 2, t_1, t_2, A_1, A_2)$, we may break a muon into two separate muons with roughly half the original amplitude each:

- let u be sampled from a distribution $q(u)$, and let $t_1 = t - u$, $t_2 = t + u$.
- let v be sampled from another distribution $r(v)$, and let the two new amplitudes add up to A, by setting $A_1 = A/2 - v$ and $A_2 = A/2 + v$.

This move can be summarized by the application of a one-to-one transformation

$$T_{1\to2}(t, A, u, v) = (t - u, \frac{A}{2} - v, t + u, \frac{A}{2} + v),$$

whose jacobian is $J_{1\to2} = 2$. Now that this move from a model with one muon to a model with two muons is fixed, the opposite move is constrained in RJMCMC. A valid choice is the inverse transformation T^{-1}, with Jacobian $J_{2\to1} = 1/2$. Finally, the user has to specify the probabilities $p_{1\to2}$ and $p_{2\to1}$ that a move from $N_\mu = 1$ to $N_\mu = 2$ is proposed and vice versa. In the end, the acceptance probability ρ of a move from $(1, t, A)$ to $(2, t_1, A_1, t_2, A_2)$ is given by

$$\rho = \min\left(1, J_{1\to2} \frac{\pi(2, t_1, A_1, t_2, A_2)}{\pi(1, t, A) q(\frac{t_2 - t_1}{2}) r(\frac{A_2 - A_1}{2})} \frac{p_{2\to1}}{p_{1\to2}}\right),$$

and the acceptance probability ρ of a move from $(2, t_1, A_1, t_2, A_2)$ to $(1, t, A)$ is given by

$$\rho = \min\left(1, J_{2\to1} \frac{\pi(1, t, A) q(\frac{t_2 - t_1}{2}) r(\frac{A_2 - A_1}{2})}{\pi(2, t_1, A_1, t_2, A_2)} \frac{p_{1\to2}}{p_{2\to1}}\right).$$

RJMCMC is very generic, but transdimensional moves have to be designed with care to be accepted often enough. However, once the chain obtained, inference on N_μ is as easy as on any other parameter: simply count how many times $N_\mu = 1$ in the chain, divide it by the length of the chain, and you have the posterior probability that $N_\mu = 1$! Reporting the results of an RJMCMC chain can be tricky. Here, one could report the posterior distribution on N_μ, and plot the marginals of the other parameters for the most probable values of N_μ. Sophisticated summaries have recently been proposed [21], which can compute the probability that one muon in particular is present. Finally, we note that while AM and relabeling can be merged, further including reversible jumps is still research work.

4.5 Conclusion and the Monte Carlo ladder

We reviewed basic Monte Carlo ideas and methods, along with some advanced ones like adaptive MCMC. We tried to give intuition for picking the right method, since none is uniformly powerful, and practical advice on implementations. To sum up the algorithms presented here and point to other important ones that we did not cover, we adapt and complete Murray's integration ladder[9]. Growing item number means higher practical complexity, but also either higher efficiency or wider applicability. Check [18] when no further reference is given.

1. Quadrature,
2. Rejection sampling,
3. Quasi-MC, Importance sampling,
4. MCMC (MH, slice sampling, etc.),
5. Adaptive MCMC [4], hybrid MC [17], tempering methods [11], sequential MC [7], particle MCMC [3].
6. Approximate Bayesian computation (ABC; [9]).

4.6 Appendix: Notations, acronyms, and recommended readings

Target densities. π always denotes the target probability density function, which in Bayesian inference problems is a posterior. Its support is $\mathbb{X} \subset \mathbb{R}^d$, and so d denotes the dimension of the problem. π_0 denotes an unnormalized version of π, formally written as $\pi_0 \propto \pi$. In Bayesian inference problems, π_0 is often of the form likelihood$\times$prior.

Estimators. $\widehat{I}_N$ always denote the estimator defined in (4.5), but the X_is used to build it depend on the context. $\widetilde{I}_N$ only denotes the importance sampling estimator.

Distributions. We write $X \sim p$ when the probability density function of X is p. Used acronyms are summed up for reference in Figure 4.11.

MEP	mean posterior (estimate)
PE	photoelectron
MC	Monte Carlo
MCMC	Markov chain Monte Carlo
MH	Metropolis-Hastings (algorithm)
CLT	central limit theorem
AM	adaptive Metropolis (algorithm)

Figure 4.11: Glossary of acronyms, in order of appearance.

For further reading on MC methods, we strongly recommend the textbook [18], to which this tutorial owes a lot. We have tried to refer to specific parts of this book whenever possible. An introduction to MCMC specifically meant for physicists is [23]. While it is a bit outdated now, it still provides insightful and untraditional explanations, especially on assessing MCMC convergence.

Acknowledgments

I would like to thank the SOS organizers for inviting me to lecture, and for proposing a rich cross-disciplinary programme that generated many interesting discussions. A large part of this tutorial was written while I was working in the Auger group at LAL, Orsay (France).

[9] Cf. his recommended two-part video lecture at `http://videolectures.net/mlss09uk_murray_mcmc/`

References

[1] C. Andrieu, E. Moulines, and P. Priouret. Stability of stochastic approximation under verifiable conditions. *SIAM Journal on Control and Optimization*, 44:283–312, 2005.

[2] C. Andrieu and J. Thoms. A tutorial on adaptive MCMC. *Statistics and Computing*, 18:343–373, 2008.

[3] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society B*, 2010.

[4] Y. Atchadé, G. Fort, E. Moulines, and P. Priouret. *Bayesian Time Series Models*, chapter Adaptive Markov chain Monte Carlo: Theory and Methods, pages 33–53. Cambridge Univ. Press, 2011.

[5] R. Bardenet. *Towards adaptive learning and inference – Applications to hyperparameter tuning and astroparticle physics*. PhD thesis, Université Paris-Sud, 2012.

[6] R. Bardenet and B. Kégl. An adaptive Monte Carlo Markov chain algorithm for inference from mixture signals. In *Proceedings of ACAT'11, Journal of Physics: Conference series*, 2012.

[7] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo in practice*. Springer, 2001.

[8] H. J. Drescher, M. Hladik, S. Ostapchenko, T. Pierog, and Werner K. Parton-based Gribov-Regge theory. *Physics Reports*, 2001.

[9] P. Fearnhead and D. Prangle. Constructing summary statistics for approximate Bayesian computation: semi-automatic ABC. *Journal of the Royal Statistical Society B*, 2012.

[10] J. M. Flegal and G. L. Jones. Batch means and spectral variance estimators in Markov chain Monte Carlo. *Annals of Statistics*, 38(2):1034–1070, 2010.

[11] W.R. Gilks, S. Richardson, and D. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 1996.

[12] P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

[13] G. R. Grimmett and D. R. Stirzaker. *Probability and random processes*. Oxford science publications, second edition, 1992.

[14] H. Haario, E. Saksman, and J. Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, 7:223–242, 2001.

[15] D. Heck, J. Knapp, J. N. Capdevielle, G. Schatz, and T. Thouw. CORSIKA: A Monte Carlo code to simulate extensive air showers. Technical report, Forschungszentrum Karlsruhe, 1998.

[16] S. F. Jarner and E. Hansen. Geometric ergodicity of Metropolis algorithms. *Stochastic processes and their applications*, 341–361, 1998.

[17] R. M. Neal. *Handbook of Markov Chain Monte Carlo*, chapter MCMC using Hamiltonian dynamics. Chapman & Hall / CRC Press, 2010.

[18] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, New York, 2004.

[19] G. Roberts, A. Gelman, and W. Gilks. Weak convergence of optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7:110–120, 1997.

[20] G. O. Roberts and J. S. Rosenthal. Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, 16:351–367, 2001.

[21] A. Roodaki. *Signal decompositions using trans-dimensional Bayesian methods.* PhD thesis, Supélec, 2012.

[22] D. S. Sivia and J. Skilling. *Data Analysis: A Bayesian Tutorial.* Oxford University press, second edition, 2006.

[23] A.D. Sokal. Monte Carlo methods in statistical mechanics: Foundations and new algorithms, 1996. Lecture notes at the Cargèse summer school.

[24] D. Wraith, M. Kilbinger, K. Benabed, O. Cappé, J.-F. Cardoso, G. Fort, S. Prunet, and C. P. Robert. Estimation of cosmological parameters using adaptive importance sampling. *Phys. Rev. D*, 80(2), 2009.

Introduction to neural networks in high energy physics

Jan Therhaag
Rheinische Friedrich-Wilhelms-Universität Bonn

Abstract

Artificial neural networks are a well established tool in high energy physics, playing an important role in both online and offline data analysis. Nevertheless they are often perceived as *black boxes* which perform obscure operations beyond the control of the user, resulting in a skepticism against any results that may be obtained using them. The situation is not helped by common explanations which try to draw analogies between artificial neural networks and the human brain, for the brain is an even more complex black box itself. In this introductory text, I will take a problem-oriented approach to neural network techniques, showing how the fundamental concepts arise naturally from the demand to solve classification tasks which are frequently encountered in high energy physics. Particular attention is devoted to the question how probability theory can be used to control the complexity of neural networks.

5.1 Introduction: A simple classification problem

Two-class classification problems are among the tasks most frequently encountered in high energy physics data analysis, usually in the form of separating interesting data (*signal*) from unwanted noise (*background*). For our discussion of neural networks, we will always consider the situation where simulated data (referred to as *Monte Carlo*) for both signal and background is available in the form of separate sets of independent events with a fixed number of characteristic features (*variables*). The Monte Carlo enables us to learn distinctive features of both event species which may then be used to discriminate the signal against the background when analyzing the observed data. The data set used for feature extraction is often called *training data*. The Monte Carlo also serves as pseudo data (*test data*) on which the discriminating function we have inferred from the training data set can be assessed. Since the true class label (signal or background) is available for Monte Carlo, one can easily determine the fraction of events in the test data set that is correctly identified. To obtain an unbiased estimate of this fraction, it is of course important that the training data set and the test data set are statistically independent.

For simplicity we consider the training data set shown in the left panel of figure 5.1. Data points in this example are fully described by their position in the (x_1, x_2)-plane. Let's assume that the orange points are drawn from the signal distribution while the blue points are drawn from the background distribution. The black line separating the two shaded areas constitutes the optimal *decision boundary* between the two classes, which could be calculated if the underlying distributions for signal and background were known. Since this is usually not the case, we have to try to infer it from the training data. A very simple approach to distinguishing the two classes would be to examine the distributions of the variables x_1 and x_2 separately and perform what is known as a cut analysis. We can however find a better solution by taking into account linear combinations of the two variables: let's assign a binary label to each data point and use these labels as target values to fit a linear function to the data points using a simple least-squares approach:

$$\hat{a}(\mathbf{x}) = \sum_{i=1}^{2} w_i x_i + w_0 \tag{5.1}$$

The parameters of this model are the coefficients w_0, w_i which are adjusted to give the best fit of the

linear function to the training data. If we choose to label the signal events (orange color) with 1 and the background events with -1, we find the solution shown in the right panel of figure 5.1. We may now call $\{\mathbf{x}|\hat{a}(\mathbf{x}) > 0\}$ the signal-like region while $\{\mathbf{x}|\hat{a}(\mathbf{x}) < 0\}$ is called the background-like region. Note that the decision boundary defined by $\{\mathbf{x}|\hat{a}(\mathbf{x}) = 0\}$ is linear as would be expected from the model. The function $\hat{a}$ can now be used to select signal candidate events from newly observed data by accepting only those which fulfill $\hat{a}(\mathbf{x}) > 0$.

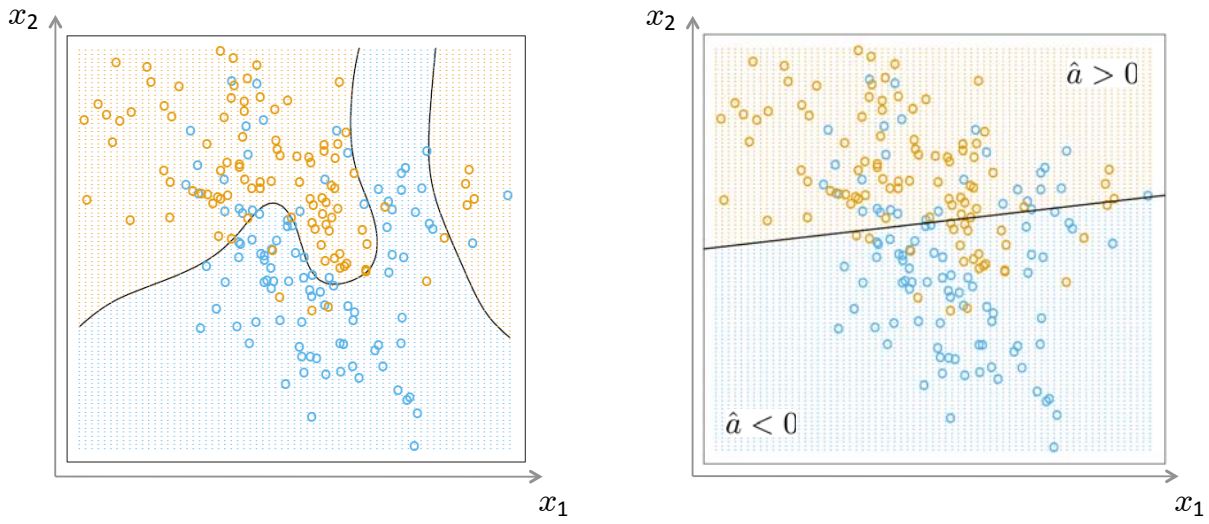


Figure 5.1: A two-dimensional classification problem. Left: Training data and optimal decision boundary calculated from the underlying distributions. Right: Fit of a linear discriminant function. (Figure adopted from [3], modified.)

5.2 (Re-)Inventing the neuron

While the simple linear approach outlined in the introduction already yields a decent solution to the problem of separating potential signal events from the background, we may ask the question: *"Given an event with values* $\mathbf{x}^{ev} \equiv (x_1^{ev}, x_2^{ev})$*, how reliable is our classification?"* Clearly, a quantitative measure of probability would be more desirable than a simple yes/no-decision. Obtaining such a measure is straightforward: we just have to map the unbounded co-domain of our function $\hat{a}$ onto the interval $[0, 1]$. A sigmoid transformation (eqn. 5.2) does the job.

$$\hat{a} \mapsto \sigma(\hat{a}) \equiv \frac{1}{1 + \exp(-\hat{a})} \tag{5.2}$$

Indeed, applying the sigmoid transformation offers a probabilistic interpretation of the classification problem, assigning values between 0 and 1 to the data points and mapping the decision boundary to 0.5 (equal probability for both signal and background) as expected. We may thus read the transformed output in the following way:

$$\sigma(\hat{a}(\mathbf{x})) \equiv p(signal|\mathbf{x}) \tag{5.3}$$

We have just invented the neuron! Indeed, in machine learning language a neuron is nothing but a linear combination of some input variables concatenated with a sigmoid transformation (figure 5.2). In this picture, our function $\hat{a}$ is called the *activation* of the neuron and $\sigma(\hat{a})$ is called its *activity*. Since we will mostly be concerned with the activity in the following, we will assign the symbol y to it from now on. Note that the entire behavior of the neuron is determined by the values of the parameters $\mathbf{w} \equiv (w_0, ..., w_i, ...w_N)$, which are called *weights*. The space of all possible weight configurations of a neuron is called the *weight space* (figure 5.3).

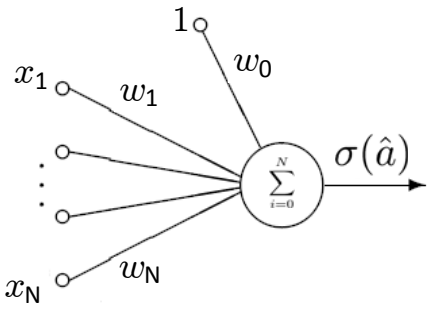


Figure 5.2: Visualization of a neuron: Weighted summation of the inputs and nonlinear (sigmoid) transformation at the output. (Figure adopted from [1], modified.)

Finding the neuron configuration which provides the best signal discrimination thus amounts to searching the weight space for the optimal set of weights for the training data set in question. This is called *training* or *learning* in the neural network jargon and is described in the following chapter.

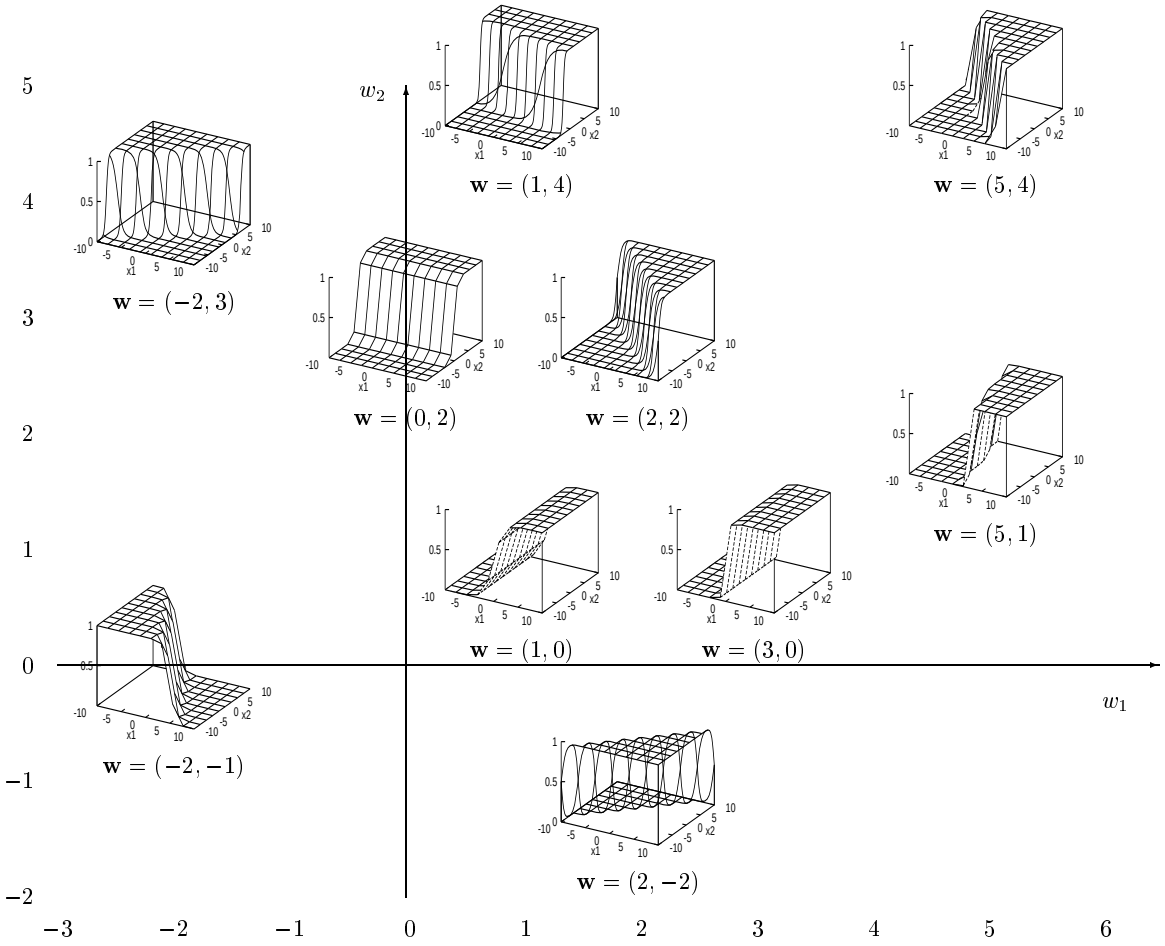


Figure 5.3: The *weight space* consists of all possible weight configurations of a neuron. Each one corresponds to a different discriminating function. (Figure adopted from [1].)

5.3 Neuron training

As we have learned in chapter 5.1, training the neuron for a given classification task amounts to searching the optimal set of weights. We can tackle this problem with a more familiar looking maximum likelihood approach if we make use of the probabilistic interpretation of the neuron output. Remember that $y(\mathbf{x}) = p(signal|\mathbf{x})$, which implies $1 - y(\mathbf{x}) = p(background|\mathbf{x})$. Combining the two into an expression for the likelihood of the training data set $\mathcal{D}$ whose true class labels are known and taking the negative logarithm, we obtain the *error function* $E(\mathbf{w})$. It is given by

$$E(\mathbf{w}) = -\ln P(\mathcal{D}|\mathbf{w}) = \sum_n (t^{(n)} \ln p(\mathcal{C}_1|\mathbf{x}^{(n)}) + (1 - t^{(n)}) p(\mathcal{C}_0|\mathbf{x}^{(n)})) \tag{5.4}$$

where we have used the class label $\mathcal{C}_1$ ($\mathcal{C}_0$) to denote signal (background) and introduced the truth label t which takes the value 1 (0) for signal (background). The optimal weight configuration of the neuron for a given training data set $\mathcal{D}$ can now be found by minimizing the error function. It is worthwhile to have a closer look at the minimization of the error function of the single neuron since we will rely on our insights when we discuss neural networks in chapter 5.4.
Although there are more sophisticated methods for minimization available, it is very instructive to consider a simple gradient descent approach for the error function. The idea is to move through the weight space by calculating the direction of steepest descent (which is just given by the negative gradient) and to take a step in that direction repeatedly. The update rule for the neuron weights thus reads:

$$\mathbf{w}^{(k)+1} = \mathbf{w}^{(k)} - \eta \frac{dE^{(k)}}{d\mathbf{w}} \tag{5.5}$$

The weights can either be updated separately for each event in the training data set (*online learning*) or after the entire training data set has been processed once (*batch learning*). A complete pass through the training data set is called an *epoch* and typically several epochs are needed to complete the training. Online learning is usually to be preferred in HEP applications since it speeds up the learning process for redundant data sets (like large MC samples with many similar events) and may help to avoid local minima by introducing a stochastic component to the minimization (see [5]). Leaving the determination of the optimal step size η aside (for a discussion see [5]), we find that the direction of the step for a weight update is given by

$$\frac{dE}{dw_j} = \sum_n -(t^{(n)} - y^{(n)}) x_j^{(n)}. \tag{5.6}$$

Note that this is proportional to the current output error $e^{(n)} = (t^{(n)} - y^{(n)})$ of the neuron which is the difference between the desired output (= the true class label) and the current output. This observation will later lead us to a simple scheme for the update rule in complex networks.

5.3.1 Overtraining in neurons

Although usually only an issue in more complex networks, we will now have a closer look at *overtraining* in the context of single neurons. This will lead to important insights on countermeasures that can also be applied in more complicated settings. Let's consider a simple training example with two variables. Since the training data set is small, it is sufficient to update the neuron weights after each epoch, i.e. use batch learning. The evolution of the weights (path in weight space) is shown in the first panel of figure 5.4. The other panels show subsequent updates of the decision boundary defined by the neuron during training. After a certain number of iterations, the weights start to evolve along a fixed direction in weight space, which means that the orientation of the decision boundary remains constant from this point on. However, the training does not stop here, but the weights keep growing, causing the slope of the neuron output function to get steeper and steeper as can be seen from the contours in figure 5.4.

This behavior is called *overtraining* and is undesirable because it means that the neuron will assign probabilities close to one (or close to zero) to any data point, even if that point is close to the decision boundary. Intuitively, one would expect that points close to the decision boundary get assigned almost

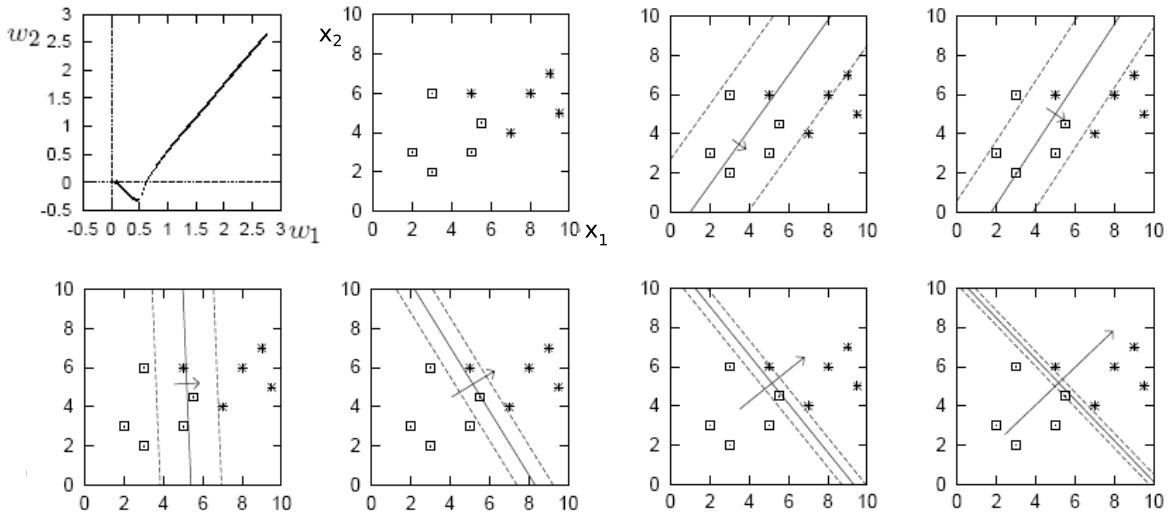


Figure 5.4: Evolution of the neuron weights and the decision boundary defined by the neuron during training. Overtraining occurs as the number of epochs during training gets large. The length of the arrow on the decision boundary is proportional to the gradient of the neuron output. (Figure adopted from [1], modified.)

identical signal and background probabilities (that is, a neuron output of 0.5), but overtraining shrinks this "zone of undecidedness" and leads to overly confident predictions. Assessing the classification performance of such an overtrained neuron on an independent test data set will usually yield a bad result, because the neuron may not only assign data points close to the decision boundary to the wrong class, but it will do so assigning very large (or very small) signal probabilities, resulting in a large output error.

Avoiding overtraining

There are several ways to avoid overtraining. An obvious and simple recipe would be to stop the training after a fixed number of iterations, interrupting the learning process before the weights can diverge ("early stopping"). But how should one determine the optimal point to stop? And how does one ensure that the training has already converged to the best set of weights? Clearly, a more informed approach is called for. We have already briefly mentioned that the performance of a given classifier can be assessed on an independent test data set. Since overtraining leads to a degradation of the performance on this data set, one can in principle monitor the classification performance of the neuron on the test data set during training and stop when the misclassification error reaches its minimum on the test data. (Note that the error function measured on the training data set will always keep decreasing and thus does not provide insights into overtraining). Despite being theoretically well founded, this approach is time consuming due to the extra evaluation of the test data set. It also requires a large amount of Monte Carlo data for two reasons: First, a test data set is definitely needed, and second, an unbiased performance assessment now calls for a third independent set, since the test data set has already been used in the optimization process.

It turns out that we can avoid this overhead by introducing the concept of *regularization*. This refers to the idea that the weights of the neuron are regularized by introducing a penalty for large weights. This can for example be realized by adding a term proportional to the sum of squared weights (often referred to as a *weight decay* term) to the error function:

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{\alpha}{2}\sum_i w_i^2 \tag{5.7}$$

Let's have a look at the behavior of the regularized neuron during training. Using the same training data as before, we now obtain the results displayed in figure 5.5. The weights are now bounded and the contours of the decision boundary remain unchanged after a certain number of iterations which is

determined by the size of the regularization parameter α. The resulting effect is similar to what can be achieved by early stopping. At first glance, one might thus say that we have just shifted the problem from making an arbitrary choice for the number of iterations to making an arbitrary choice for α. But when we return to the discussion of regularization in neural networks later on, we will see that the optimal choice of the regularization parameter can be inferred from the training data itself in what is called the *evidence framework*.

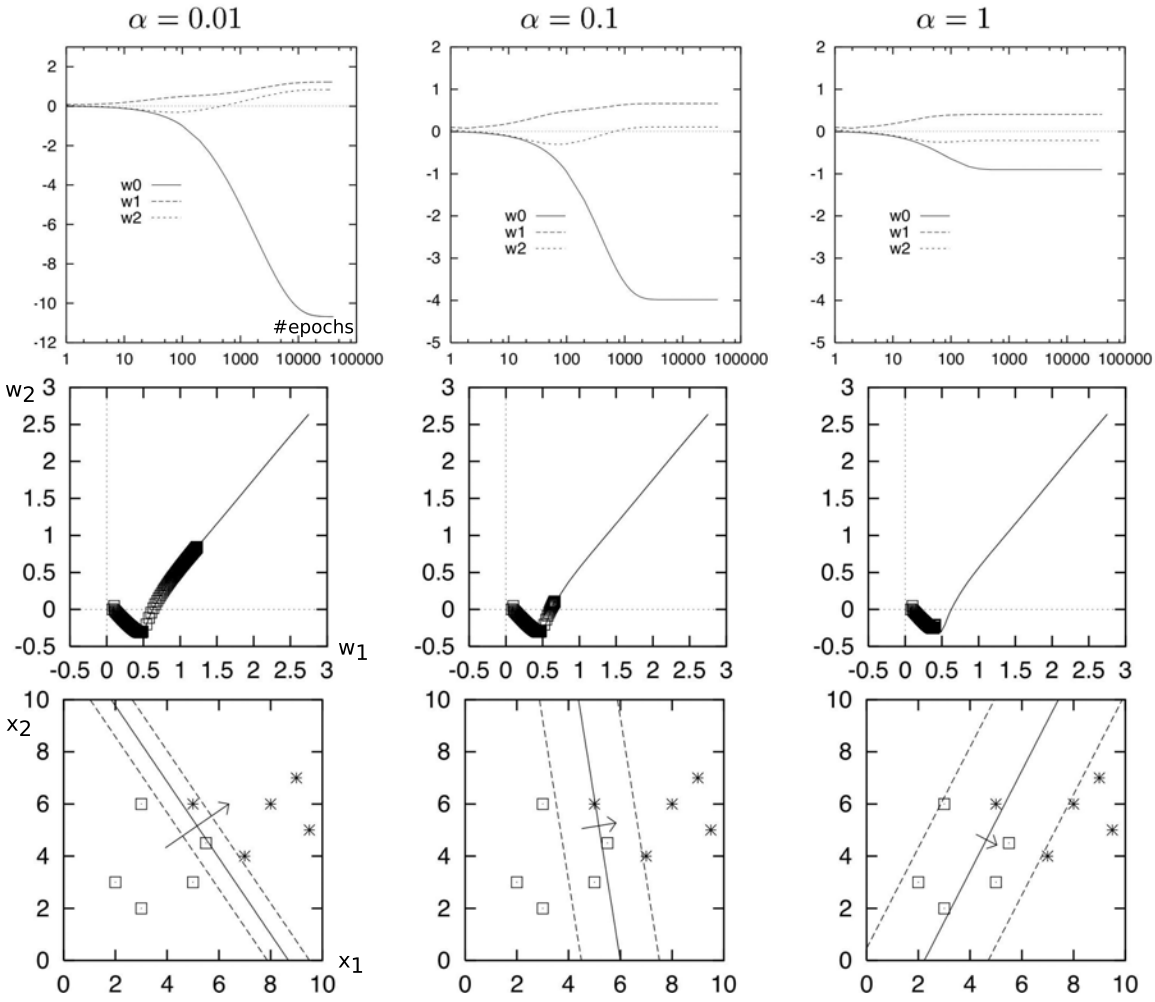


Figure 5.5: Effect of regularization for different values of the regularization parameter α on the evolution of neuron weights during training. Top row: The weights are bounded and stay constant after a certain number of iterations. Middle row: Weight evolution now follows the path marked by the black squares - a behavior similar to early stopping is observed. Bottom row: The decision boundary defined by the neuron after convergence of the weights. (Figure adopted from [1], modified.)

5.4 From neurons to networks

Having discussed the single neuron thoroughly in the first chapter, the generalization to neural networks composed from several neurons is now straightforward. But how should the neurons be arranged to form a neural network? Let me remind you of the universal approximation theorem, which forms the theoretical foundation for the modeling of functions through neural networks. It reads:

Let $\sigma(\cdot)$ be a non-constant, bounded and monotone-increasing continuous function. Let $\mathcal{C}(I_D)$ denote the space of continuous functions on the D-dimensional hypercube. Then, for any given function $f \in \mathcal{C}(I_D)$ and $\epsilon > 0$ there exist an integer M and sets of real constants w_j, w_{ji} where $i = 1, ..., D$ and $j = 1, ..., M$ such that

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=1}^{M} w_j \sigma \left(\sum_{i=1}^{D} w_{ji} x_i + w_{j0} \right) \tag{5.8}$$

is an approximation of $f(\cdot)$, that is $|y(\mathbf{x}) - f(\mathbf{x})| < \epsilon$.

The notation was not chosen arbitrarily here - identifying the parameters w_j, w_{ji} with the weights and $\sigma(\cdot)$ with the sigmoid transform, a sloppy translation of the theorem reads:

One can build any continuous function from neurons!

But the theorem not only tells us that we can approximate any function with neurons, but also how to do it: Figure 5.6 shows that the structure described by formula 5.8 is composed of neurons organized in layers, where the output of a neuron in one layer becomes an input to all the neurons in the following layer. This structure is called a *feed-forward network* : The first layer provides connections for all input variables and passes them on to an intermediate layer of neurons, where they are summed and transformed (innermost term in formula 5.8). Subsequent layers may follow until the network response is provided at the output of the neurons in the last layer (*output layer*). This last layer will often be linear, so that the entire non-linearity (and hence the flexibility to approximate arbitrary functions) of the network is provided by the intermediate layers. These layers are called *hidden layers* and the neurons they consist of are referred to as *hidden neurons* or *hidden units* because they are not directly connected to the network input or output. Different network architectures are often classified by the number of hidden layers and the term *single layer network* is often used to describe a network with one hidden layer only.

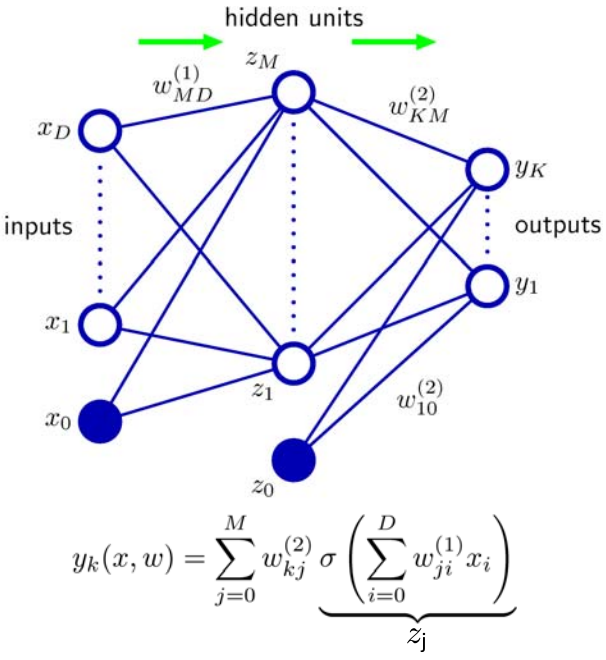


Figure 5.6: A feed-forward neural network consists of neurons organized in layers. Neurons in one layer provide the input for the neurons in the following layer. (Figure adopted from [4], modified.)

5.4.1 Network complexity

Figure 5.7 illustrates the approximation of several simple functions by a single layer network with three hidden units. There is only one input variable and the network output is one-dimensional. The dashed lines represent the output values of the hidden neurons which are then linearly combined in the output layer to form the network output. Since the approximation of functions by such a network resembles an expansion of the target function in a basis consisting of sigmoid functions (=neurons), it is intuitively clear that the number and complexity of the functions that a network can represent grows with the number of hidden neurons. But the complexity is not only determined by the number of hidden neurons, but also by the typical size of the weights $\mathbf{w}$ throughout the network. In our discussion of neuron training we have seen that large weights correspond to a steep decision boundary, that is a large variation of the neuron output over a small interval of the input values. The approximation of functions with intricate small scale features must thus contain neurons with large weights which provide a sufficient change of the output value in a small interval. It can even be shown that in the limit of infinite networks ($n_{neuron} \to \infty$) the complexity of the functions that can be represented is entirely determined by the characteristic size of the weights [9]. In our discussion of network training we will thus assume that the network in question is always large enough to represent the discriminating function we are interested in and that the training task amounts to finding the optimal set of weights.

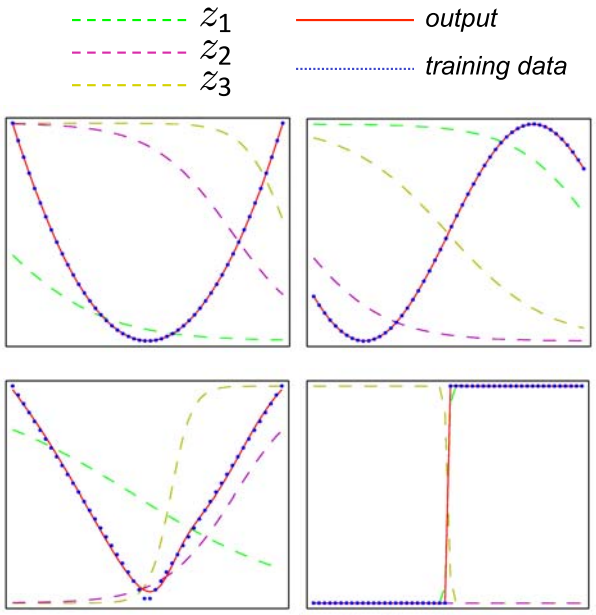


Figure 5.7: Approximation of simple functions by a single layer network with three hidden units. The outputs of the hidden units are labeled z_1, z_2, z_3. (Figure adopted from [4], modified.)

5.4.2 Network training

Recalling what we have learned from our discussion of neuron training, we know that the best configuration of the weights can be found by following the gradient of the error function through the weight space until we arrive at a minimum. But deriving an update rule for all the weights in a large network seems to be very difficult since a lot of complicated derivatives are involved. Luckily, Rumelhart et al. have shown that the update rules for any network can be easily derived by an algorithm known as *backpropagation* [6]. I will not give a detailed description of backpropagation here, but just briefly outline the rules that can be derived from it. First recall that the weights of a single neuron are updated by taking a step along the gradient of the error function which is proportional to the current size of the output error in each coordinate direction:

$$\frac{dE}{dw_k} = (y - t)x_k \tag{5.9}$$

It turns out that a very similar rule applies to the neurons in a network:

$$\frac{dE}{dw_{ij}} = \delta_j z_i \tag{5.10}$$

$$\text{where } \delta_k = (y_k - t_k) \text{ for output neurons} \tag{5.11}$$

$$\text{and } \delta_j \propto \sum_k w_{kj}\delta_k \text{ else.} \tag{5.12}$$

Careful examination of this rule shows that a neuron in the output layer behaves in exactly the same way as a single neuron: The update for a given weight is obtained by multiplying the input associated with that weight (which is one of the outputs from the previous layer) by the current output error $e_k \equiv \delta_k = (y_k - t_k)$ of that neuron. Note that the updates in the other layers can then be performed subsequently by propagating the output error "backwards" through the network according to equation 5.12 - hence the name "backpropagation".
We have just discovered one of the characteristic rules of network training: While input information is passed forward from layer to layer, errors are passed backward in the same way during training (figure 5.8). This allows for very efficient network implementations which exploit the forward-backward symmetry to speed up the training.

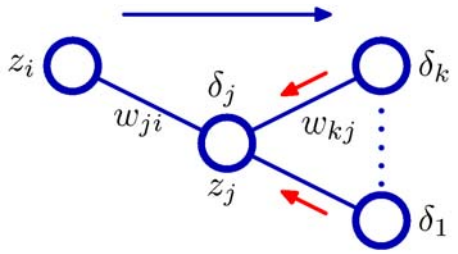


Figure 5.8: Backpropagation of errors in a neural network. (Figure adopted from [4].)

5.4.3 Overtraining revisited

We have seen that a network with a sufficient number of hidden neurons can evolve into a representation of the optimal discriminating function for a given problem by adapting to the training data set during the learning phase - but what if the network adapts *too well* to the training data? This situation may be compared to a fitting problem where one tries to describe a set of data points which where generated from a linear model with some noise on top (the noise is usually associated with an uncertainty of the measurement) by polynomials of varying order. Even though the underlying model is linear, a polynomial of sufficient order will always be able to pass through all the data points and thus produce smaller residuals. But this supposedly better fit will have low predictive power since it does not generalize well: An interpolation (or even more extreme an extrapolation) using this polynomial will yield predictions which typically differ significantly from the underlying model. *Overfitting* has occurred. In the realm of neural networks, a behavior like this is called *overtraining*, and we have already encountered it in our discussion of neuron training, where it lead to overly confident class assignments.
In complex networks, overtraining can be as extreme as "learning the class labels of the training data set" and lead to a severe deterioration of the classification performance on any other sample. The functions represented by such a heavily overtrained network will usually display numerous small scale features which correspond to the statistical fluctuations found in the training data (see figure 5.9). Just as a polynomial of high order will give poor results when interpolating a linear model, a network like this will not generalize well to data not encountered during training.

5.4.4 Regularization in neural networks

How can we avoid overtraining in neural networks? One may of course reduce the size of the network, thus limiting the complexity of functions representable by the network. But as the type of the optimal discriminating function is usually unknown, one might unwantedly exclude it from the set of accessible

functions this way. Stopping the training prematurely as discussed earlier in the context of neuron training is also discouraged: Undesirable small scale features may already develop in certain regions of the phase space before a satisfactory global solution has been found. The observation made earlier, stating that intricate small scale variations of the function represented by the network indicate the presence of large weights, provides a strong hint that weight regularization may be effectively used to reduce overtraining and produce smooth and well generalizing decision boundaries. Adding the quadratic regularizer term encountered earlier to the error function of the network and choosing an appropriate regularization strength parameter α indeed leads to a well behaved solution to our initial classification problem displayed in figure 5.9. Not only do unwanted statistically induced features of the original (unregularized) discriminating function (like disconnected regions in which data will be assigned to the same class) vanish, but also does the regularized function follow the true decision boundary as induced from the underlying distributions more closely. An inspection of the network weights would show that this behavior can indeed be attributed to a reduction of the average weight size. One nagging question remains, though: How can we determine the optimal amount of regularization? Is there a well informed approach to set α or are we left with heuristics only? To answer this question, we have to see how neural networks can be understood from the viewpoint of Bayesian statistics - a field which seems utterly unrelated at first glance.

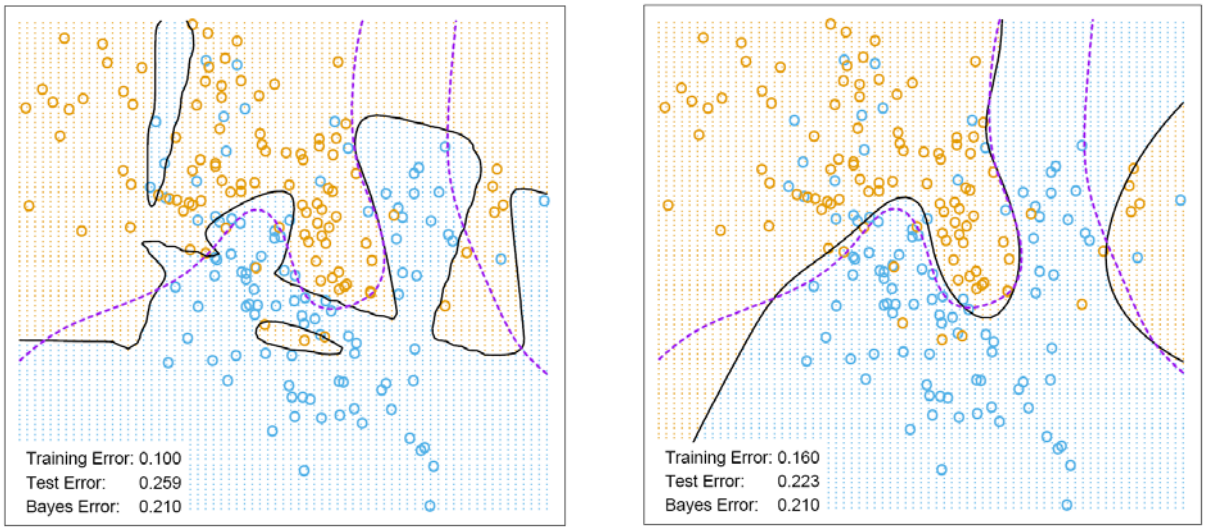


Figure 5.9: Application of a single layer network with ten hidden units to the classification problem from section 5.1. Left: Overtraining is clearly visible, the error on the training data set is much smaller than the test error. Right: Regularization smooths the decision boundary and improves the performance on the test data. The Bayes error (measure of best achievable performance) is given for comparison. (Figure adopted from [3].)

5.5 Joining Bayesian statistics and neural networks

Bringing together Bayesian statistics and neural networks looks like a strange idea in the first place - an approach guided by the fundamental principles of probability and a field of statistical learning largely governed by heuristics don't seem to go together too well. Nevertheless we will see that it is this connection which sheds light on the problem of optimal regularization.
Remember that the neuron output y can be interpreted as a probability for each event and that we can express this probability using the truth class label $t \in \{0, 1\}$:

$$\begin{aligned} P(t=1|\mathbf{w},\mathbf{x}) &= y & (5.13)\\ P(t=0|\mathbf{w},\mathbf{x}) &= 1-y & (5.14)\\ \Rightarrow P(t|\mathbf{w},\mathbf{x}) &= y^t(1-y)^{1-t} = \exp[t\ln y + (1-t)\ln(1-y)] & (5.15) \end{aligned}$$

Note that the exponent is just the error function we introduced in section 5.3 which allows us to write

$$P(\mathcal{D}|\mathbf{w}) = \exp(-E(\mathbf{w})). \tag{5.16}$$

If we try a similar interpretation for the regularization (weight decay) term, we see that it resembles a Gaussian log-probability distribution for $\mathbf{w}$:

$$P(\mathbf{w}|\alpha) = \frac{1}{Z_{WD}(\alpha)}\exp(-\frac{\alpha}{2}(\mathbf{w}^\top\mathbf{w})) \tag{5.17}$$

where $Z_{WD}(\alpha)$ is an appropriate normalization constant.
We can now combine the two using Bayes' theorem to obtain an expression for the probability distribution of the networks parameters $\mathbf{w}$ for a given training data set $\mathcal{D}$ and regularization strength α. The likelihood is now represented by the error function while the prior is represented by the regularization term.

$$P(\mathbf{w}|\mathcal{D},\alpha) = \frac{P(\mathcal{D}|\mathbf{w})P(\mathbf{w}|\alpha)}{\int P(\mathcal{D}|\mathbf{w})P(\mathbf{w}|\alpha)d\mathbf{w}} = \frac{1}{Z_{\tilde{E}}}\exp(-\tilde{E}(\mathbf{w})) \tag{5.18}$$

With this insight we can reinterpret the training of a neural network as an inference task where the optimal set of weights maximizes the posterior probability defined in equation 5.18. This can be illustrated by returning to the simplest possible network - a single neuron - and following the evolution of the posterior distribution of the neuron weights $P(\mathbf{w}|\mathcal{D},\alpha)$ during training as more and more data is considered (figure 5.10).

5.5.1 Network regularization in the evidence framework

Let's return to the problem of optimal weight regularization. Making use of our new probabilistic interpretation of network training, this amounts to finding an expression for the posterior distribution of α:

$$P(\alpha|\mathcal{D}) \propto P(\mathcal{D}|\alpha)P(\alpha). \tag{5.19}$$

If we assume no prior knowledge about the regularization parameter α (e.g. a flat prior $P(\alpha)$), Bayes' theorem implies that $P(\alpha|\mathcal{D})$ is proportional to $P(\mathcal{D}|\alpha)$, which can be written as

$$P(\mathcal{D}|\alpha) = \int P(\mathcal{D}|\mathbf{w})P(\mathbf{w}|\alpha)d\mathbf{w}. \tag{5.20}$$

Close inspection of equation 5.18 reveals that this integral corresponds to the normalization constant $Z_{\tilde{E}}$ of the posterior distribution of $\mathbf{w}$, which is often called the *evidence*. The optimal regularization strength can thus be inferred from the training data by maximizing the evidence with respect to α. Unfortunately, the integral in equation 5.20 is analytically intractable, thus we have to resort to a Laplace approximation.

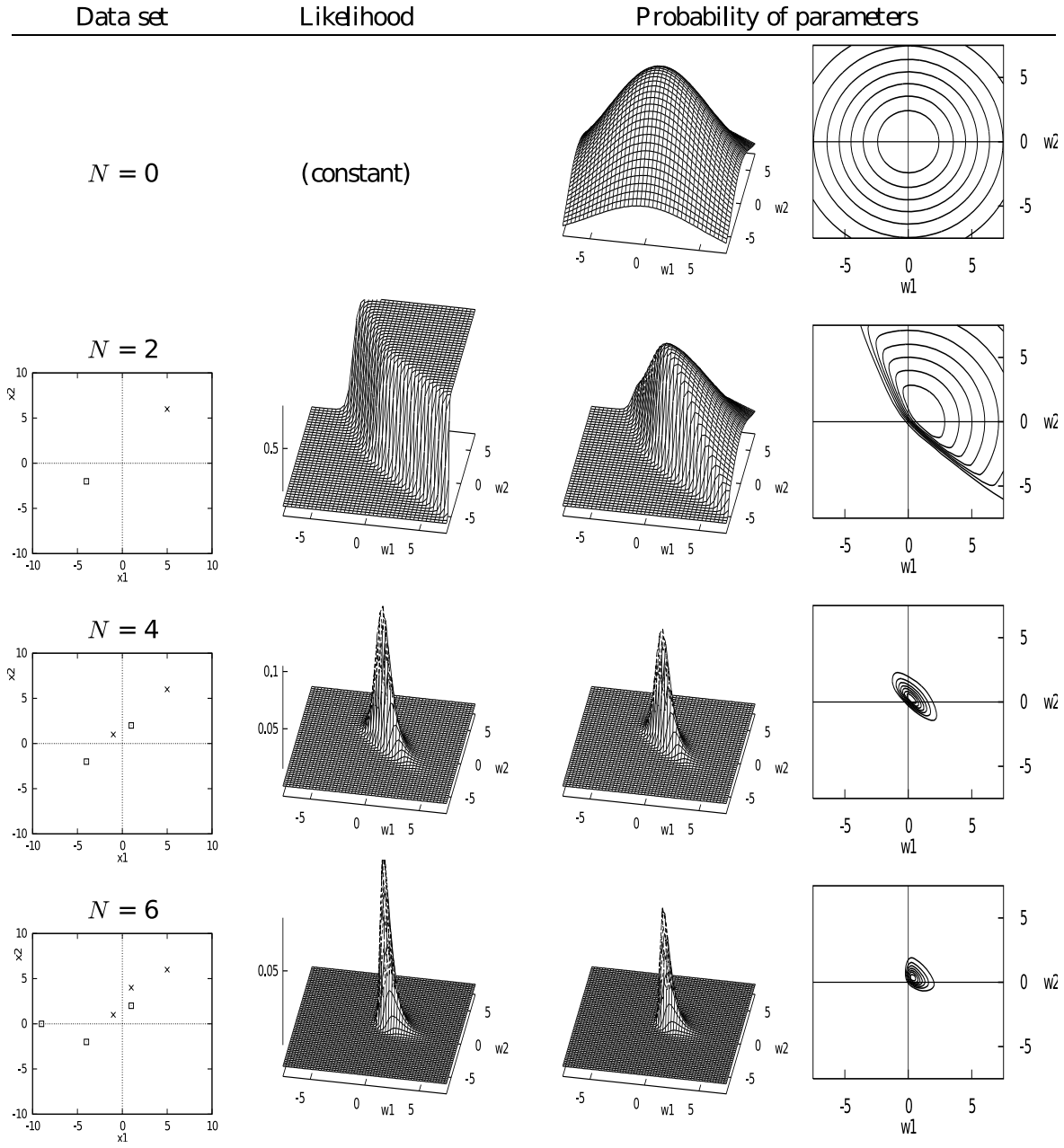


Figure 5.10: Evolution of the posterior distribution of the weights during the training of a neuron. (Figure adopted from [1]).

Evaluating the evidence

Consider a smooth function $f(\mathbf{w})$ of the network parameters $\mathbf{w}$, which has a maximum at the most probable value $\mathbf{w}_{MP}$. Obviously, also $\ln f(\mathbf{w})$ has a maximum at $\mathbf{w}_{MP}$ and Taylor expanding around that point gives

$$\begin{aligned} \ln f(\mathbf{w}) &\approx \ln f(\mathbf{w}_{MP}) - \frac{1}{2}(\mathbf{w}-\mathbf{w}_{MP})^\top \mathbf{A}(\mathbf{w}-\mathbf{w}_{MP}) \\ \text{with } \mathbf{A} &= -\nabla\nabla \ln f(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_{MP}}. \end{aligned} \tag{5.21}$$

Exponentiating this expression we obtain a Gaussian approximation of $f(\mathbf{w})$ around the maximum:

$$f(\mathbf{w}) \approx f(\mathbf{w}_{MP}) \exp(-\frac{1}{2}(\mathbf{w}-\mathbf{w}_{MP})^\top \mathbf{A}(\mathbf{w}-\mathbf{w}_{MP})) \tag{5.22}$$

We can now apply this approximation to the evidence $Z_{\tilde{E}}$ and solve the resulting Gaussian integral to obtain:

$$\begin{aligned} P(\mathcal{D}|\alpha) &= \int P(\mathcal{D}|\mathbf{w})P(\mathbf{w}|\alpha)d\mathbf{w} \\ &= \int \exp(-E(\mathbf{w})) \cdot (\frac{\alpha}{2\pi})^{W/2} \exp(-\frac{\alpha}{2}\mathbf{w}^\top\mathbf{w})d\mathbf{w} \\ &= (\frac{\alpha}{2\pi})^{W/2} \exp(-E(\mathbf{w}_{MP}) - \frac{\alpha}{2}\mathbf{w}_{MP}^\top\mathbf{w}_{MP}) \\ &\quad \cdot \int \exp(-\frac{1}{2}(\mathbf{w}-\mathbf{w}_{MP})^\top \mathbf{A}(\mathbf{w}-\mathbf{w}_{MP}))d\mathbf{w} \\ &= \frac{\alpha^{W/2}}{(\det \mathbf{A})^{1/2}} \exp(-\tilde{E}(\mathbf{w}_{MP})) \\ \Rightarrow \ln p(\mathcal{D}|\alpha) &= -\tilde{E}(\mathbf{w}_{MP}) + \frac{W}{2}\ln\alpha - \frac{1}{2}\ln(\det \mathbf{A}) \end{aligned} \tag{5.23}$$

The second and third term are called *Occam terms*. They balance the contribution from the error function in such a way that the resulting model complexity will provide both a good fit to the training data set and a high generalizability. A schematic view of the connection between $\ln p(\mathcal{D}|\alpha)$ and the classification performance on the training/test sample is depicted in figure 5.12. Figure 5.11 shows an example for evidence-based regularization in a two dimensional classification problem.

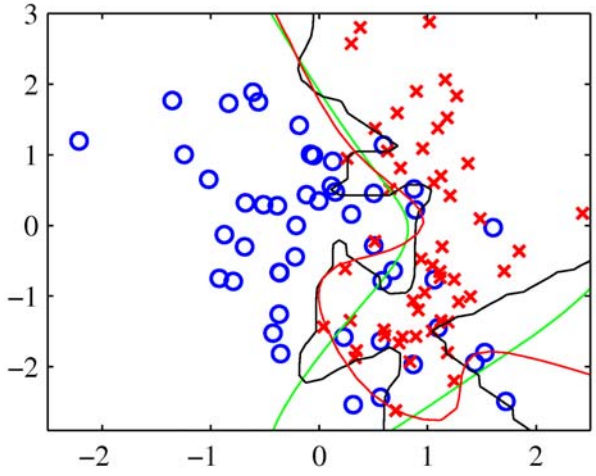


Figure 5.11: Application of evidence-based regularization to a single layer network with eight hidden units. Green: Optimal decision boundary. Black: Decision boundary obtained with unregularized training. Red: Decision boundary obtained with evidence-based regularization. (Figure adopted from [4].)

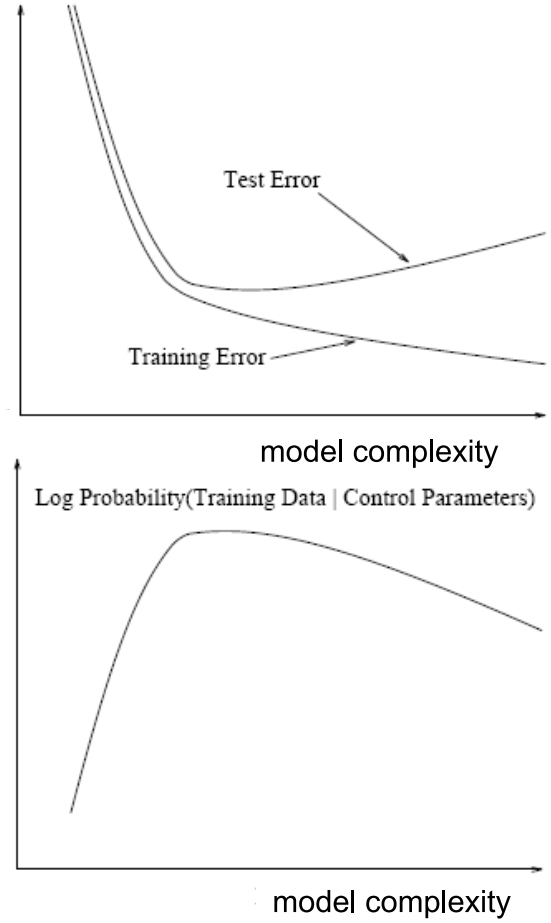


Figure 5.12: Relation between model complexity, evidence and classifier test performance. (Figure adopted from [2], modified.)

Variable selection through regularization

In high energy physics analyses, the selection of discriminating variables is usually motivated by the features of the underlying physics process. But in many situations it is not obvious which variables will provide the best discrimination power, in particular when they are combined in a multivariate method. Surprisingly, evidence-based network regularization can also be used to prune useless variables from a model and thus help to identify the relevant ones for a given problem. This is achieved by generalizing the weight-decay approach by introducing separate regularization constants α_i for all input variables of the network (see figure 5.13). The connection weights for different input nodes can now be regularized individually and the weights associated with uninformative variables tend to be constraint to values close to zero which means that they are effectively dropped from the model. This method is known as *Automatic Relevance Determination (ARD)* (MacKay and Neal, unpublished), see also [2].

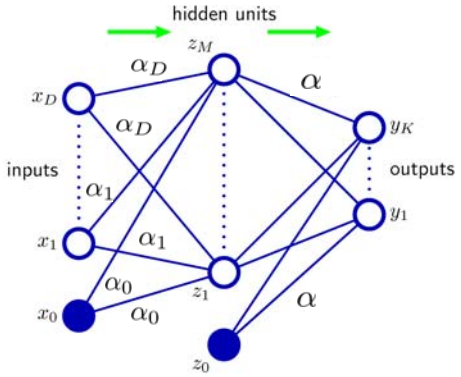


Figure 5.13: A neural network with separate regularization parameters for each input variable - a configuration used for *Automatic Relevance Determination*. (Figure adopted from [4], modified.)

5.5.2 Predictions and confidence

The Bayesian approach to neural networks also provides a natural way to embody the uncertainty of the network output. When the fully trained network is applied to data that was not encountered during training to make predictions, its output $P(t^{N+1}|\mathbf{x}^{N+1})$ is usually calculated using just the most probable set of weights derived during training. A look at our initial neuron training example may suffice to convince ourselves that this approach is not optimal: Consider two new data points **A** and **B** (left panel of figure 5.14). Not only will they be assigned to the same class, but they will also be assigned the same probability since they share the same distance from the decision boundary. Wouldn't it be more natural if the class prediction for **B** was less confident since the training data is scarce and thus the model description probably poor in that region? The reason for that shortcoming is that we have implicitly approximated the posterior distribution of the weights by a delta-distribution at $\mathbf{w}_{MP}$, neglecting the spread of $P(\mathbf{w}|\mathcal{D},\alpha)$ and thus the uncertainty associated with the weights:

$$\begin{aligned} P(\mathbf{w}|\mathcal{D},\alpha) &\approx \delta(\mathbf{w}-\mathbf{w}_{MP}) \\ \Rightarrow P(t^{N+1}|\mathbf{x}^{N+1},\mathcal{D},\alpha) &= \int P(t^{N+1}|\mathbf{x}^{N+1},\mathbf{w})\delta(\mathbf{w}-\mathbf{w}_{MP})d\mathbf{w} \\ &= P(t^{N+1}|\mathbf{x}^{N+1},\mathbf{w}_{MP}) \end{aligned} \tag{5.24}$$

If we drop the delta-approximation, we face another analytically intractable integral in equation 5.24 - but as seen before we can resort to a Gaussian approximation to obtain an estimate of the predictive posterior $P(t^{N+1}|\mathbf{x}^{N+1},\mathcal{D},\alpha)$. Using this improved method to make predictions yields the result shown in the right panel of figure 5.14: The farther one moves in the phase space from the bulk of the training data, the farther the contours spread from the decision boundary. The prediction for point **B** has a greater uncertainty than that for point **A**. The application of that principle to a neural network with eight hidden units is shown in figure 5.15.

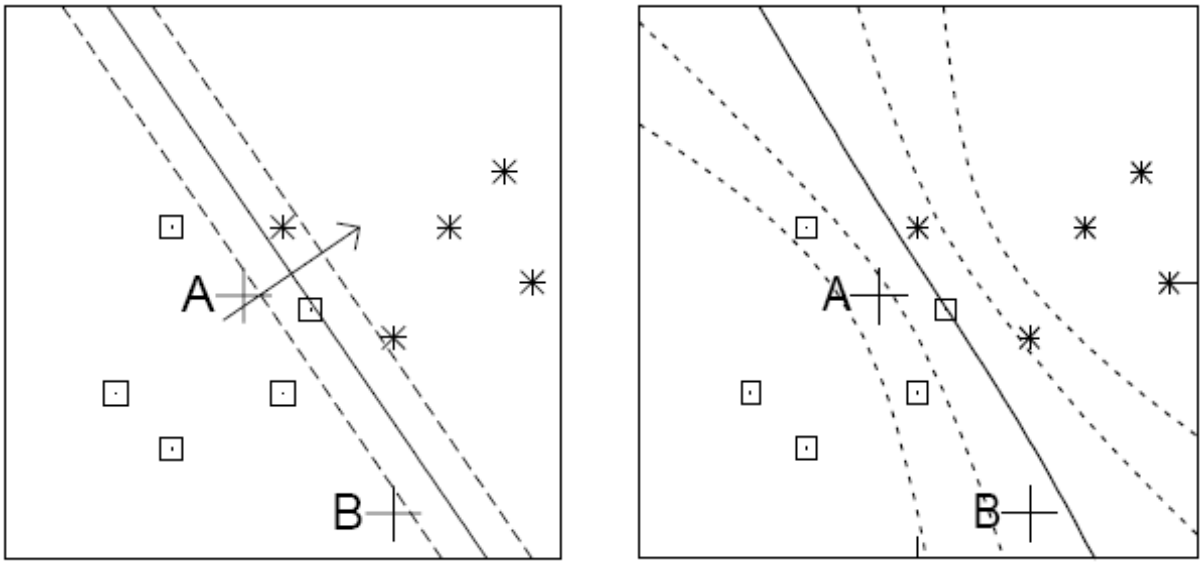


Figure 5.14: Contours of the decision boundary as defined by a fully trained neuron. Left: Approximation of the posterior distribution of the neuron weights by a delta-distribution. Right: Evaluation of the expectation by Gaussian approximation. (Figure adopted from [1].)

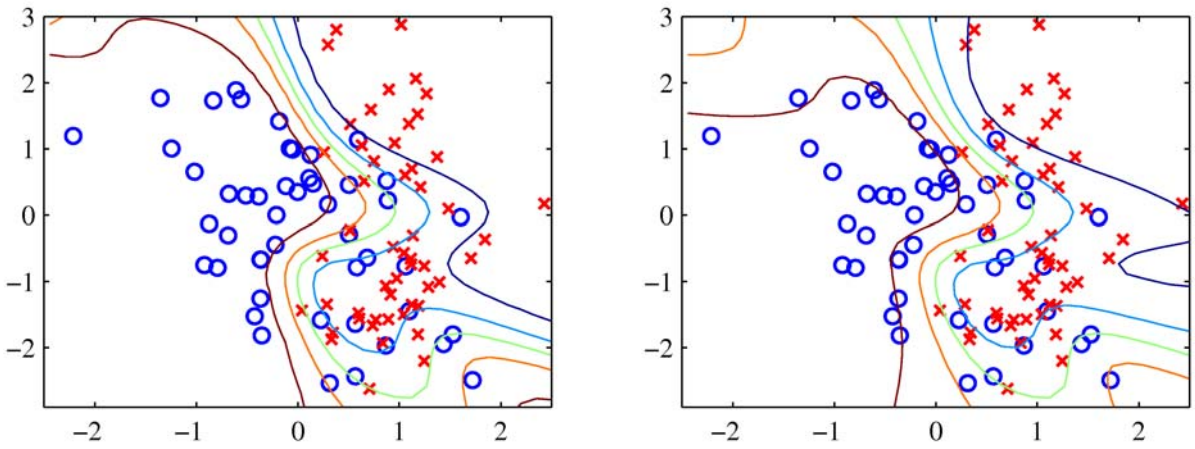


Figure 5.15: Contours of the discrimination function defined by a neural network with eight hidden units. Left: Approximation of the posterior of the weights by a delta-distribution. Right: Gaussian approximation of the posterior. (Figure adopted from [4].)

5.6 Conclusions

Neural networks are no black boxes - we have seen how a neuron can be constructed by transforming a linear discriminant function into a probability statement and how the universal approximation theorem dictates how neurons can be combined to form universal function approximators known as feed-forward neural networks.
The Bayesian view of neural network training as an inference task for the weights has led to interesting insights on network regularization and the uncertainty of network predictions.

5.7 Further reading and neural network software

Most of the figures in this text were taken from the books of MacKay [1], Bishop [4], and Hastie, Tibshirani and Friedman [3]. These books provide an excellent resource for further reading on neural networks. A lot of useful practical hints for users of neural networks are given in [5].
Among the neural network software packages most widely used in high energy physics are TMVA [7] and NeuroBayes [8].

5.8 Acknowledgments

I would like to thank the organizers for giving me the opportunity to give a lecture at the school of statistics. The program of the school was diverse and I enjoyed many interesting discussions. I would also like to thank the authors of the aforementioned books for the permission to use their illustrative figures for this introductory text.

References

[1] D.J.C. MacKay, *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003, www.inference.phy.cam.ac.uk/mackay/itila/

[2] D.J.C. MacKay, "Bayesian Methods for Neural Networks: Theory and Application, *Course notes for Neural Networks Summer School*, 1995, http://www.inference.phy.cam.ac.uk/mackay/BayesNets.html

[3] T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning*, Springer, 2nd edition, 2009

[4] C.M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006

[5] Y. LeCun, L. Bottou, G.B. Orr, K.R. Mueller, "Efficient BackProp", in *Neural Networks: Tricks of the trade*, Springer, 1998

[6] D.E. Rumelhart, G.E. Hinton, R.J. Williams, "Learning representations by back-propagating errors", in *Nature*, 323 533-536

[7] A. Hoecker, P. Speckmayer, J. Stelzer, J. Therhaag, E. von Toerne, and H. Voss, "TMVA: Toolkit for Multivariate Data Analysis", PoS A CAT 040 (2007) [physics/0703039].

[8] M. Feindt, U. Kerzel, "The NeuroBayes neural network package", *Nuclear Instruments and Methods in Physics Research*, 2006, Vol. 559 Issue 1, 190-194

[9] R.M. Neal, "Priors for infinite networks", Technical Report CRG-TR-94-1, Dept. of Computer Science, University of Toronto, 1994

Boosted Decision Trees & Applications

Yann COADOU
CPPM, Aix-Marseille Université, CNRS/IN2P3, Marseille, France

6.1 Introduction

Decision trees are a machine learning technique more and more commonly used in high energy physics, while it has been widely used in the social sciences. It was first developed in the context of data mining and pattern recognition, and gained momentum in various fields, including medical diagnostic, insurance and loan screening, and optical character recognition (OCR) of handwritten text.

It was developed and formalised by Breiman *et al.* [1] who proposed the CART algorithm (Classification And Regression Trees) with a complete and functional implementation of decision trees.

The basic principle is rather simple: it consists in extending a simple cut-based analysis into a multivariate technique by continuing to analyse events that fail a particular criterion. Many, if not most, events do not have all characteristics of either signal or background. If that were the case then an analysis with a few criteria would allow to easily extract the signal. The concept of a decision tree is therefore to not reject right away events that fail a criterion, and instead to check whether other criteria may help to classify these events properly.

In principle a decision tree can deal with multiple output classes, each branch splitting in many subbranches. In these proceedings only binary trees will be considered, with only two possible classes: signal and background.

Section 6.2 describes how a decision tree is constructed and what parameters can influence its development. Section 6.3 provides some insights into some of the intrinsic limitations of a decision tree and how to address some of them. One possible extension of decision trees, boosting, is introduced in Section 6.4, and other techniques trying to reach the same goal as boosting are presented in Section 6.5. Conclusions are summarised in Section 6.6 and references to available decision tree software are given in Section 6.7.

While starting with this powerful multivariate technique, it is important to remember that before applying it to real data, it is crucial to have a good understanding of the data and of the model used to describe them. Any discrepancy between the data and model will provide an artificial separation that the decision trees will use, misleading the analyser. The hard part (and interest) of the analysis is in building the proper model, not in extracting the signal. But once this is properly done, decision trees provide a very powerful tool to increase the significance of any analysis.

6.2 Growing a tree

Mathematically, decision trees are rooted binary trees (as only trees with two classes, signal and background, are considered). An example is shown in Fig. 6.1. A decision tree starts from an initial node, the root node. Each node can be recursively split into two daughters or branches, until some stopping condition is reached. The different aspects of the process leading to a full tree, indifferently referred to as growing, training, building or learning, are described in the following sections.

6.2.1 Algorithm

Consider a sample of signal (s_i) and background (b_j) events, each with weight w_i^s and w_j^b, respectively, described by a set $\vec{x}_i$ of variables. This sample constitutes the root node of a new decision tree.

Starting from this root node, the algorithm proceeds as follows:

1. If the node satisfies any stopping criterion, declare it as terminal (that is, a leaf) and exit the algorithm.
2. Sort all events according to each variable in $\vec{x}$.
3. For each variable, find the splitting value that gives the best separation between two children, one with mostly signal events, the other with mostly background events (see Section 6.2.4 for details). If the separation cannot be improved by any splitting, turn the node into a leaf and exit the algorithm.
4. Select the variable and splitting value leading to the best separation and split the node in two new nodes (branches), one containing events that fail the criterion and one with events that satisfy it.
5. Apply recursively from step 1 on each node.

At each node, all variables can be considered, even if they have been used in a previous iteration: this allows to find intervals of interest in a particular variable, instead of limiting oneself to using each variable only once.

It should be noted that a decision tree is human readable: one can trace exactly which criteria an event satisfied in order to reach a particular leaf. It is therefore possible to interpret a tree in terms, e.g., of physics, defining selection rules, rather than only as a mathematical object.

In order to make the whole procedure clearer, let us take the tree in Fig. 6.1 as an example. Consider that all events are described by three variables: the transverse momentum, p_T, of the leading jet, the reconstructed top quark mass M_t and the scalar sum of p_T's of all reconstructed objects in the event, H_T. All signal and background events make up the root node.

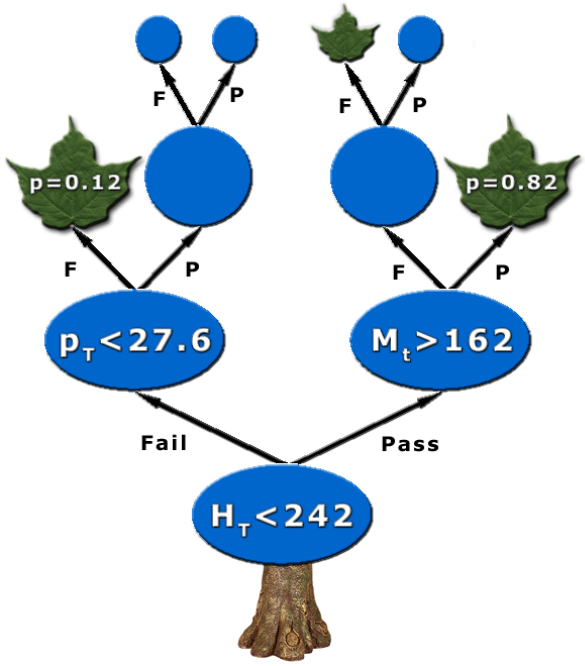


Figure 6.1: Graphical representation of a decision tree. Blue ellipses and disks are internal nodes with their associated splitting criterion; green leaves are terminal nodes with purity p.

Following the above algorithm one should first sort all events according to each variable:

- $p_T^{s1} \leq p_T^{b34} \leq \cdots \leq p_T^{b2} \leq p_T^{s12}$,
- $H_T^{b5} \leq H_T^{b3} \leq \cdots \leq H_T^{s67} \leq H_T^{s43}$,
- $M_t^{b6} \leq M_t^{s8} \leq \cdots \leq M_t^{s12} \leq M_t^{b9}$,

where superscript s_i (b_j) represents signal (background) event i (j). Using some measure of separation one may find that the best splitting for each variable is (arbitrary unit):

- $p_T < 56$ GeV, separation $= 3$,

- $H_T < 242$ GeV, separation = 5,
- $M_t < 105$ GeV, separation = 0.7.

One would conclude that the best split is to use $H_T < 242$ GeV, and would create two new nodes, the left one with events failing this criterion and the right one with events satisfying it. One can now apply the same algorithm recursively on each of these new nodes. As an example consider the right-hand-side node with events that satisfied $H_T < 242$ GeV. After sorting again all events in this node according to each of the three variables, it was found that the best criterion was $M_t > 162$ GeV, and events were split accordingly into two new nodes. This time the right-hand-side node satisfied one of the stopping conditions and was turned into a leaf. From signal and background training events in this leaf, the purity was computed as $p = 0.82$ (see the next Section).

6.2.2 Decision tree output

The decision tree output for a particular event i is defined by how its $\vec{x}_i$ variables behave in the tree:

1. Starting from the root node, apply the first criterion on $\vec{x}_i$.
2. Move to the passing or failing branch depending on the result of the test.
3. Apply the test associated to this node and move left or right in the tree depending on the result of the test.
4. Repeat step 3 until the event ends up in a leaf.
5. The decision tree output for event i is the value associated with this leaf.

There are several conventions used for the value attached to a leaf. It can be the purity $p = \frac{s}{s+b}$ where s (b) is the sum of weights of signal (background) events that ended up in this leaf during training. It is then bound to $[0, 1]$, close to 1 for signal and close to 0 for background.

It can also be a binary answer, signal or background (mathematically typically $+1$ for signal and 0 or -1 for background) depending on whether the purity is above or below a specified critical value (e.g. $+1$ if $p > \frac{1}{2}$ and -1 otherwise).

Looking again at the tree in Fig. 6.1, the leaf with purity $p = 0.82$ would give an output of 0.82, or $+1$ as signal if choosing a binary answer with a critical purity of 0.5.

6.2.3 Tree parameters

The number of parameters of a decision tree is relatively limited. The first one is not specific to decision trees and applies to most techniques requiring training: how to normalise signal and background before starting the training? Conventionally the sums of weights of signal and background events are chosen to be equal, giving the root node a purity of 0.5, that is, an equal mix of signal and background.

Other parameters concern the selection of splits. One first needs a list of questions to ask, like "is variable $x_i < \text{cut}_i$?", requiring a list of discriminating variables and a way to evaluate the best separation between signal and background events (the goodness of the split). Both aspects are described in more detail in Sections 6.2.4 and 6.2.5.

The splitting has to stop at some point, declaring such nodes as terminal leaves. Conditions to satisfy can include:

- a minimum leaf size. A simple way is to require at least N_{min} training events in each node after splitting, in order to ensure statistical significance of the purity measurement, with a statistical uncertainty $\sqrt{N_{min}}$. It becomes a little bit more complicated with weighted events, as is normally the case in high energy physics applications. One may then want to consider using the effective number of events instead:

$$N_{\mathit{eff}} = \frac{\left(\sum_{i=1}^{N} w_i\right)^2}{\sum_{i=1}^{N} w_i^2},$$

for a node with N events associated to weights w_i ($N_{\mathit{eff}} = N$ for unweighted events). This would ensure a proper statistical uncertainty.

- having reached perfect separation (all events in the node belong to the same class).
- an insufficient improvement with further splitting.
- a maximal tree depth. One can decide that a tree cannot have more than a certain number of layers (for purely computational reasons or to have like-size trees).

Finally a terminal leaf has to be assigned to a class. This is classically done by labelling the leaf as signal if $p > 0.5$ and background otherwise.

6.2.4 Splitting a node

The core of a decision tree algorithm resides in how a node is split into two. Consider an impurity measure $i(t)$ for node t, which describes to what extent the node is a mix of signal and background. Desirable features of such a function are that it should be:

- maximal for an equal mix of signal and background (no separation).
- minimal for nodes with either only signal or only background events (perfect separation).
- symmetric in signal and background purities, as isolating background is as valuable as isolating signal.
- strictly concave in order to reward purer nodes. This tends to favour end cuts with one smaller node and one larger node.

A figure of merit can be constructed with this impurity function, as the decrease of impurity for a split S of node t into two children t_P (pass) and t_F (fail):

$$\Delta i(S,t) = i(t) - p_P \cdot i(t_P) - p_F \cdot i(t_F),$$

where p_P (p_F) is the fraction of events that passed (failed) split S.

The goal is to find the split S^* that maximises the decrease of impurity:

$$\Delta i(S^*,t) = \max_{S \in \{\text{splits}\}} \Delta i(S,t).$$

It will result in the smallest residual impurity, which minimises the overall tree impurity.

Some decision tree applications use an alternative definition of the decrease of impurity [2]:

$$\Delta i(S,t) = i(t) - \min\left(p_P \cdot i(t_P), p_F \cdot i(t_F)\right),$$

which may perform better for, e.g., particle identification.

A stopping condition can be defined using the decrease of impurity: one may decide to not split a node if $\Delta i(S^*,t)$ is less than some predefined value. One should nonetheless always be careful when using such early-stopping criteria, as sometimes a seemingly very weak split may allow child nodes to be powerfully split further (see Section 6.3.1 about pruning).

For signal (background) events with weights w_s^i (w_b^i) the purity is defined as:

$$p = \frac{\sum_{i \in signal} w_s^i}{\sum_{i \in signal} w_s^i + \sum_{j \in bkg} w_b^j}.$$

Simplifying this expression one can write down the signal purity (or simply purity) as $p_s = p = \frac{s}{s+b}$ and the background purity as $p_b = \frac{b}{s+b} = 1 - p_s = 1 - p$. Common impurity functions (exhibiting most of the desired features mentioned previously) are illustrated in Fig. 6.2:

- the misclassification error: $1 - \max(p, 1-p)$,
- the (cross) entropy [1]: $-\sum_{i=s,b} p_i \log p_i$,
- the Gini index of diversity.

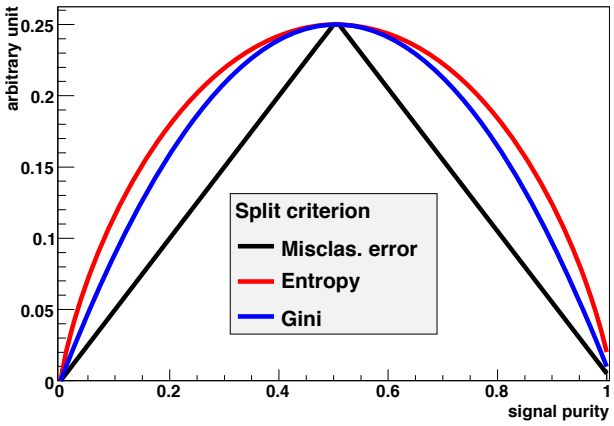


Figure 6.2: Various popular impurity measures as a function of signal purity.

For a problem with any number of classes, the Gini index [3] is defined as:

$$\text{Gini} = \sum_{i,j \in \{\text{classes}\}}^{i \neq j} p_i p_j.$$

The statistical interpretation is that if one assigns a random object to class i with probability p_i, the probability that it is actually in class j is p_j and the Gini index is the probability of misclassification.

In the case of two classes, signal and background, then $i = s$ and $j = b$, $p_s = p = 1 - p_b$ and the Gini index is:

$$\text{Gini} = 1 - \sum_{i=s,b} p_i^2 = 2p(1-p) = \frac{2sb}{(s+b)^2}.$$

The Gini index is the most popular in decision tree implementations. It typically leads to similar performance to entropy.

Other measures are also used sometimes, which do not satisfy all criteria listed previously but attempt at optimising signal significance, usually relevant in high energy physics applications:

- cross section significance: $-\frac{s^2}{s+b}$,
- excess significance: $-\frac{s^2}{b}$.

6.2.5 Variable selection

As stated before, the data and model have to be in good agreement before starting to use a decision tree (or any other analysis technique). This means that all variables used should be well described. Forgetting this prerequisite will jeopardise the analysis.

Overall decision trees are very resilient to most features associated to variables. They are not too much affected by the "curse of dimensionality", which forbids the use of too many variables in most multivariate techniques. For decision trees the CPU consumption scales as $nN \log N$ with n variables and N training events. It is not uncommon to encounter decision trees using tens or hundreds of variables.

A decision tree is immune to duplicate variables: the sorting of events according to each of them would be identical, leading to the exact same tree. The order in which variables are presented is completely irrelevant: all variables are treated equal. The order of events in the training samples is also irrelevant.

If variables are not discriminating, they will simply be ignored and will not add any noise to the decision tree. The final performance will not be affected, it will only come with some CPU overhead during both training and evaluation.

Decision trees can deal easily with both continuous and discrete variables, simultaneously.

Another typical task before applying a multivariate technique is to transform input variables by for instance making them fit in the same range or taking the logarithm to regularise the variable. This is totally unnecessary with decision trees, which are completely insensitive to the replacement of any subset of input variables by (possibly different) arbitrary strictly monotone functions of them. The explanation is trivial. Let $f : x_i \to f(x_i)$ be a strictly monotone function: if $x > y$ then $f(x) > f(y)$. Then any ordering of events by x_i is the same as by $f(x_i)$, which means that any split on x_i will create the same separation as a split on $f(x_i)$, producing the same decision tree. This means that decision trees have some immunity against outliers.

Until now the splits considered have always answered questions of the form "Is $x_i < c_i$?", while it is also possible to make linear combinations of input variables and ask instead "Is $\sum_i a_i x_i < c_i$?", where $a = (a_1, .., a_n)$ is a set of coefficients such that $||a||^2 = \sum_i a_i^2 = 1$. One would then choose the optimal split $S^*(a^*)$ and the set of linear coefficients a^* that maximises $\Delta i(S(a), t)$. This is in practise rather tricky to implement and very CPU intensive. This approach is also powerful only if strong linear correlations exist between variables. If this is the case, a simpler approach would consist in first decorrelating the input variables and then feeding them to the decision tree. Even if not doing this decorrelation, a decision tree will anyway find the correlations but in a very suboptimal way, by successive approximations, adding complexity to the tree structure.

It is possible to rank variables in a decision tree. To rank variable x_i one can add up the decrease of impurity for each node where variable x_i was used to split. The variable with the largest decrease of impurity is the best variable.

There is nevertheless a shortcoming with variable ranking in a decision tree: variable masking. Variable x_j may be just a little worse than variable x_i and would end up never being picked in the decision tree growing process. Variable x_j would then be ranked as irrelevant, and one would conclude this variable has no discriminating power. But if one would remove x_i, then x_j would become very relevant.

There is a solution to this feature, called surrogate splits [1]. For each split, one compares which training events pass or fail the optimal split to which events pass or fail a split on another variable. The split that mimics best the optimal split is called the surrogate split. One can then take this into consideration when ranking variables. This has applications in case of missing data: one can then replace the optimal split by the surrogate split.

All in all, variable rankings should never be taken at face value. They do provide valuable information but should not be over-interpreted.

6.3 Tree (in)stability

Despite all the nice features presented above, decision trees are known to be relatively unstable. If trees are too optimised for the training sample, they may not generalise very well to unknown events. This can be mitigated with pruning, described in Section 6.3.1.

A small change in the training sample can lead to drastically different tree structures, rendering the physics interpretation a bit less straightforward. For sufficiently large training samples, the performance of these different trees will be equivalent, but on small training samples variations can be very large. This doesn't give too much confidence in the result.

Moreover a decision tree output is by nature discrete, limited by the purities of all leaves in the tree. This means that to decrease the discontinuities one has to increase the tree size and complexity, which may not be desirable or even possible. Then the tendency is to have spikes in the output distribution at specific purity values, as illustrated in Fig. 6.3, or even two delta functions at ± 1 if using a binary answer rather than the purity output.

Section 6.3.2 describes how these shortcomings can for the most part be addressed by averaging.

6.3.1 Pruning a tree

When growing a tree, each node contains fewer and fewer events, leading to an increase of the statistical uncertainty on each new split. The tree will tend to become more and more specialised, focusing on

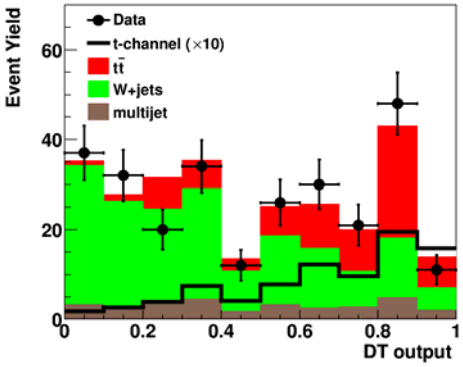


Figure 6.3: Comparison of signal, backgrounds and data for one of the decision tree outputs in an old D0 single top quark search. The discrete output of a decision tree is clearly visible, but data are well reproduced by the model.

properties of the training sample that may not reflect the expected result, had there been infinite statistics to train on.

A first approach to mitigate this effect, sometimes referred to as pre-pruning, has already been described in Section 6.2, using stopping conditions. They included requiring a minimum number of events in each node or a minimum amount of separation improvement. The limitation is that requiring too big a minimum leaf size or too much improvement may prevent further splitting that could be very beneficial.

Another approach consists in building a very large tree and then cutting irrelevant branches by turning an internal node and all its descendants into a leaf, removing the corresponding subtree. This is post-pruning, or simply pruning.

Why would one wish to prune a decision tree? It is possible to get a perfect classifier on the training sample: mathematically the misclassification rate can be made as little as one wants on training events. For instance one can build a tree such that each leaf contains only one class of events (down to one event per leaf if necessary). The training error is then zero. But when passing events through the tree that were not seen during training, the misclassification rate will most likely be non-zero, showing signs of overtraining. Pruning helps in avoiding such effects, by eliminating subtrees (branches) that are deemed too specific to the training sample.

There are many different pruning algorithms available. Only three of them, among the most commonly used, are presented briefly.

Expected error pruning was introduced by Quinlan [4] in the ID3 algorithm. A full tree is first grown. One then computes the approximate expected error for a node (using the Laplace error estimate $E = \frac{n_e+n_c-1}{N+n_c}$, where n_e is the number of misclassified events out of N events in the node, and n_c is the number of classes, 2 for binary trees), and compares it to the weighted sum of expected errors from its children. If the expected error of the node is less than that of the children, then the node is pruned. This algorithm is fast and does not require a separate sample of events to do the pruning, but it is also known to be too aggressive: it tends to prune large subtrees containing branches with good separation power.

Reduced error pruning was also introduced by Quinlan [4], and requires a separate pruning sample. Starting from terminal leaves, the misclassification rate on the pruning sample for the full tree is compared to the misclassification rate when a node is turned into a leaf. If the simplified tree has better performance, the subtree is pruned. This operation is repeated until further pruning increases the misclassification rate.

Cost-complexity pruning is part of the CART algorithm [1]. The idea is to penalise complex trees (with many nodes and/or leaves) and to find a compromise between a good fit to training data (requiring a larger tree) and good generalisation properties (usually better achieved with a smaller tree).

Consider a fully grown decision tree, T_{max}. For any subtree T (with N_T nodes) of T_{max} with a misclassification rate $R(T)$ one can define the cost complexity $R_\alpha(T)$:

$$R_\alpha(T) = R(T) + \alpha N_T,$$

where α is a complexity parameter. When trying to minimise $R_\alpha(T)$, a small α would help picking T_{max} (no cost for complexity) while a large α would keep only the root node, T_{max} being fully pruned. The optimally pruned tree is somewhere in the middle.

In a first pass, for terminal nodes t_P and t_F emerging from the split of node t, by construction $R(t) \geq R(t_P) + R(t_F)$. If these quantities are equal, then one can prune off t_P and t_F.

Now, for node t and subtree T_t, by construction $R(t) > R(T_t)$ if t is non-terminal. The cost complexity for node t is:

$$R_\alpha(\{t\}) = R_\alpha(t) = R(t) + \alpha$$

since $N_t = 1$. If $R_\alpha(T_t) < R_\alpha(t)$, then the branch T_t has smaller cost-complexity than the single node $\{t\}$ and should be kept. But for a critical value $\alpha = \rho_t$, obtained by solving $R_{\rho_t}(T_t) = R_{\rho_t}(t)$, that is:

$$\rho_t = \frac{R(t) - R(T_t)}{N_T - 1},$$

pruning the tree and making t a leaf becomes preferable. The node with the smallest ρ_t is the weakest link and gets pruned. The algorithm is applied recursively on the pruned tree until it is completely pruned, leaving only the root node.

This generates a sequence of decreasing cost-complexity subtrees. For each of them (from T_{max} to the root node), one then computes their misclassification rate on the validation sample. It will first decrease, and then go through a minimum before increasing again. The optimally pruned tree is the one corresponding to the minimum.

It should be noted that the best pruned tree may not be optimal when part of a forest of trees, such as those introduced in the next Sections.

6.3.2 Averaging several trees

Pruning was shown to be helpful in maximising the generalisation potential of a single decision tree. It nevertheless doesn't address other shortcomings of trees like the discrete output or the sensitivity of the tree structure to the training sample composition. A way out is to proceed with averaging several trees, with the added potential bonus that the discriminating power may increase, as briefly described in the 2008 proceedings of this school [5].

Such a principle was introduced from the beginning with the so-called V-fold cross-validation [1], a useful technique for small samples. After dividing a training sample $\mathcal{L}$ into V subsets of equal size, $\mathcal{L} = \bigcup_{v=1..V} \mathcal{L}_v$, one can train a tree T_v on the $\mathcal{L} - \mathcal{L}_v$ sample and test it on $\mathcal{L}_v$. This produces V decision trees, whose outputs are combined into a final discriminant:

$$\frac{1}{V} \sum_{v=1..V} T_v.$$

Following this simple-minded approach, many other averaging techniques have been developed, after realising the enormous advantage it provided. Bagging, boosting and random forests are such techniques and will be described in the following Sections.

6.4 Boosting

As will be shown in this section, the boosting algorithm has turned into a very successful way of improving the performance of any type of classifier, not only decision trees. After a short history of boosting in Section 6.4.1, the generic algorithm is presented in Section 6.4.2 and a specific implementation (AdaBoost) is described in Section 6.4.3. Boosting is illustrated with a few examples in Section 6.4.4. Finally other examples of boosting implementations are given in Section 6.4.5.

6.4.1 A brief history of boosting

Boosting has appeared quite recently. The first provable algorithm of boosting was proposed by Schapire in 1990 [6]. It worked in the following way:

- train a classifier T_1 on a sample of N events.
- train T_2 on a new sample with N events, half of which were misclassified by T_1.
- build T_3 on events where T_1 and T_2 disagree.

The boosted classifier was defined as a majority vote on the outputs of T_1, T_2 and T_3.

In 1995 Freund followed up on this idea [7], introducing boosting by majority. It consisted in combining many learners with a fixed error rate. This was an impractical prerequisite for a viable automated algorithm, but was a stepping stone to the proposal by Freund&Schapire of the first functional boosting algorithm, called AdaBoost [8].

Boosting, and in particular boosted decision trees, have become increasingly popular in high energy physics. The MiniBooNe experiment first compared the performance of different boosting algorithms and artificial neural networks for particle identification [9]. The D0 experiment was the first to use boosted decision trees for a search, which lead to the first evidence (and then observation) of single top quark production [10].

6.4.2 Boosting algorithm

Boosting is a general technique which is not limited to decision trees, although it is often used with them. It can apply to any classifier, e.g., neural networks. It is hard to make a very good discriminant, but is it relatively easy to make simple ones which are certainly more error-prone but are still performing at least marginally better than random guessing. Such discriminants are called weak classifiers [5]. The goal of boosting is to combine such weak classifiers into a new, more stable one, with a smaller error rate and better performance.

Consider a training sample $\mathbb{T}_k$ containing N_k events. The i^{th} event is associated with a weight w_i^k, a vector of discriminating variables $\vec{x}_i$ and a class label $y_i = +1$ for signal, -1 for background. The pseudocode for a generic boosting algorithm is:

```
Initialise T_1
for k in 1..N_tree
  train classifier T_k on T_k
  assign weight α_k to T_k
  modify T_k into T_{k+1}
```

The boosted output is some function $F(T_1, .., T_{N_{tree}})$, typically a weighted average:

$$F(i) = \sum_{k=1}^{N_{tree}} \alpha_k T_k(\vec{x}_i).$$

Thanks to this averaging, the output becomes quasi-continuous, mitigating one of the limitations of single decision trees.

6.4.3 AdaBoost

One particularly successful implementation of the boosting algorithm is AdaBoost, introduced by Freund&Schapire [8]. AdaBoost stands for adaptive boosting, referring to the fact that the learning procedure adjusts itself to the training data in order to classify it better. There are many variations on the same theme for the actual implementation, and it is the most common boosting algorithm. It typically leads to better results than without boosting, up to the Bayes limit as will be seen later.

An actual implementation of the AdaBoost algorithm works as follows. After having built tree T_k one should check which events in the training sample $\mathbb{T}_k$ are misclassified by T_k, hence defining the misclassification rate $R(T_k)$. In order to ease the math, let us introduce some notations. Define $\mathbb{I} : X \rightarrow$

$\mathbb{I}(X)$ such that $\mathbb{I}(X) = 1$ if X is true, and 0 otherwise. One can now define a function that tells whether an event is misclassified by T_k. In the decision tree output convention of returning only $\{\pm 1\}$ it gives:

$$\text{isMisclassified}_k(i) = \mathbb{I}\big(y_i \times T_k(i) \leq 0\big),$$

while in the purity output convention (with a critical purity of 0.5) it leads to:

$$\text{isMisclassified}_k(i) = \mathbb{I}\big(y_i \times (T_k(i) - 0.5) \leq 0\big).$$

The misclassification rate is now:

$$R(T_k) = \varepsilon_k = \frac{\sum_{i=1}^{N_k} w_i^k \times \text{isMisclassified}_k(i)}{\sum_{i=1}^{N_k} w_i^k}.$$

This misclassification rate can be used to derive a weight associated to tree T_k:

$$\alpha_k = \beta \times \ln \frac{1 - \varepsilon_k}{\varepsilon_k},$$

where β is a free boosting parameter to adjust the strength of boosting (it was set to 1 in the original algorithm).

The core of the AdaBoost algorithm resides in the following step: each event in $\mathbb{T}_k$ has its weight changed in order to create a new sample $\mathbb{T}_{k+1}$ such that:

$$w_i^k \to w_i^{k+1} = w_i^k \times e^{\alpha_k \cdot \text{isMisclassified}_k(i)}.$$

This means that properly classified events are unchanged from $\mathbb{T}_k$ to $\mathbb{T}_{k+1}$, while misclassified events see their weight increased by a factor e^{α_k}. The next tree T_{k+1} is then trained on the $\mathbb{T}_{k+1}$ sample. This next tree will therefore see a different sample composition with more weight on misclassified events, and will therefore try harder to classify properly difficult events that tree T_k failed to identify correctly. The final AdaBoost result for event i is:

$$T(i) = \frac{1}{\sum_{k=1}^{N_{\text{tree}}} \alpha_k} \sum_{k=1}^{N_{\text{tree}}} \alpha_k T_k(i).$$

As an example, assume for simplicity the case $\beta = 1$. A not-so-good classifier, with a misclassification rate $\varepsilon = 40\%$ would have a corresponding $\alpha = \ln \frac{1-0.4}{0.4} = 0.4$. All misclassified events would therefore get their weight multiplied by $e^{0.4} = 1.5$, and the next tree will have to work a bit harder on these events. Now consider a good classifier with an error rate $\varepsilon = 5\%$ and $\alpha = \ln \frac{1-0.05}{0.05} = 2.9$. Now misclassified events get a boost of $e^{2.9} = 19$ and will contribute decisively to the structure of the next tree! This shows that being failed by a good classifier brings a big penalty: it must be a difficult case, so the next tree will have to pay much more attention to this event and try to get it right.

It can be shown [11] that the misclassification rate ε of the boosted result on the training sample is bounded from above:

$$\varepsilon \leq \prod_{k=1}^{N_{tree}} 2\sqrt{\varepsilon_k(1 - \varepsilon_k)}.$$

If each tree has $\varepsilon_k \neq 0.5$, that is to say, if it does better than random guessing, then the conclusion is quite remarkable: the error rate falls to zero for a sufficiently large N_{tree}! A corollary is that the training data is overfitted.

Overtraining is usually regarded has a negative feature. Does this mean that boosted decision trees are doomed because they are too powerful on the training sample? Not really. What matters most is not the error rate on the training sample, but rather the error rate on a testing sample. This may well decrease at first, reach a minimum and increase again as the number of trees increases. In such a case one should stop boosting when this minimum is reached. It has been observed that quite often boosted decision trees do not go through such a minimum, but rather tend towards a plateau in testing error. One

could then decide to stop boosting after having reached this plateau.

In a typical high energy physics problem, the error rate may not even be what one wants to optimise. A good figure of merit on the testing sample would rather be the significance. Figure 6.4 (top left) illustrates this behaviour, showing how the significance saturates with an increasing number of boosting cycles. One could argue one should stop before the end and save resources, but at least the performance does not deteriorate with increasing boosting.

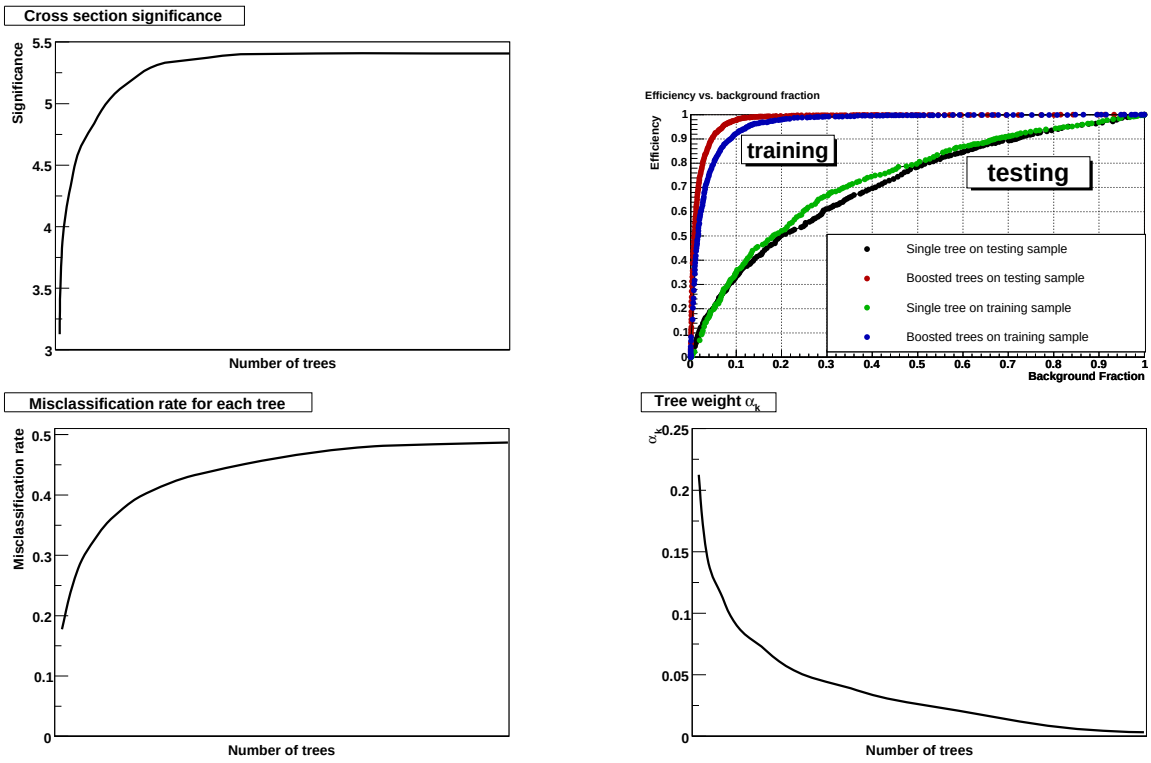


Figure 6.4: Behaviour of boosting. Top left: Significance as a function of the number of boosted trees. Top right: Signal efficiency vs. background efficiency for single and boosted decision trees, on the training and testing samples. Bottom left: Misclassification rate of each tree as a function of the number of boosted trees. Bottom right: Weight of each tree as a function of the number of boosted trees.

Another typical curve one tries to optimise is the signal efficiency vs. the background efficiency. Figure 6.4 (top right) clearly exemplifies the interesting property of boosted decision trees. The performance is clearly better on the training sample than on the testing sample (the training curves are getting very close to the upper left corner of perfect separation), with a single tree or with boosting, a clear sign of overtraining. But the boosted tree is still performing better than the single tree on the testing sample, proof that it does not suffer from this overtraining.

People have been wondering why boosting leads to such features, with typically no loss of generalisation performance due to overtraining. No clear explanation has emerged yet, but some ideas have come up. It may have to do with the fact that during the boosting sequence, the first tree is the best while the others are successive minor corrections, which are given smaller weights. This is shown at the bottom of Fig. 6.4, where the misclassification rate of each new tree separately is actually increasing, while the corresponding tree weight is decreasing. This is no surprise: during boosting the successive trees are specialising on specific event categories, and can therefore not perform as well on other events. So the trees that lead to a perfect fit of the training data are contributing very little to the final boosted decision tree output on the testing sample. When boosting decision trees, the last tree is not an evolution of the first one that performs better, quite the contrary. The first tree is typically the best, while others bring dedicated help for misclassified events. The power of boosting does not rely in the last tree in the sequence, but rather

in combining a suite of trees that focus on different events.

Finally a probabilistic interpretation of AdaBoost was proposed [12] which gives some insight into the performance of boosted decision trees. It can be shown that for a boosted output T flexible enough:

$$e^{T(i)} = \frac{p(S|i)}{p(B|i)} = BD(i),$$

where one recognises the Bayes discriminant [13]. This means that the AdaBoost algorithm will tend towards the Bayes limit, the maximal separation one can hope to reach.

6.4.4 Boosting practical examples

The examples of this section were produced using the TMVA package [14].

A simple 2D example

This example starts from code provided by G. Cowan [15]. Consider a system described by two variables, x and y, as shown in Fig. 6.5.

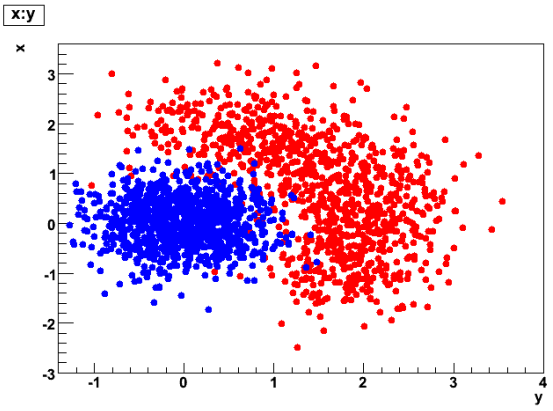


Figure 6.5: $x : y$ correlation of input variables used to illustrate how decision trees work. Signal is in blue, background in red.

Building a first tree leads to the result shown in Fig. 6.6 (top), as a decision tree (left) and with criteria applied in the $x : y$ plane (right). This single decision tree is already performing quite well. Its output is either $+1$ (signal) or -1 (background), so the only way to use the output is to keep either -1 or $+1$ testing candidates. After applying boosting, the separation between signal and background can be improved further. The boosted decision tree output is shown in Fig. 6.6 (bottom left). The bottom right plot in this figure shows the background rejection vs. signal efficiency curves for the first decision tree, for the boosted decision trees and for a Fisher discriminant analysis (for comparison), all run on the same testing events. The boosted decision trees perform best, with more freedom than on a single tree to choose a working point (either choose a signal efficiency or background rejection).

The XOR problem

Another way of showing how a decision tree can handle complicated inputs is the XOR problem, a small version of the checkerboard, illustrated in Fig. 6.7. With enough statistics (left column), even a single tree is already able to find more or less the optimal separation, so boosting cannot actually do much better. On the other hand this type of correlations is a killer for a Fisher discriminant, which fairs as bad as random guessing.

One can repeat the exercise, this time with limited statistics (right column in Fig. 6.7). Now a single tree is not doing such a good job anymore and the Fisher discriminant is completely unreliable. Boosted decision trees, on the other hand, are doing almost as well as with full statistics, separating almost perfectly signal and background. This illustrates very clearly how the combination of weak classifiers (see for instance the lousy performance of the first tree) can generate a high performance discriminant with a boosting algorithm.

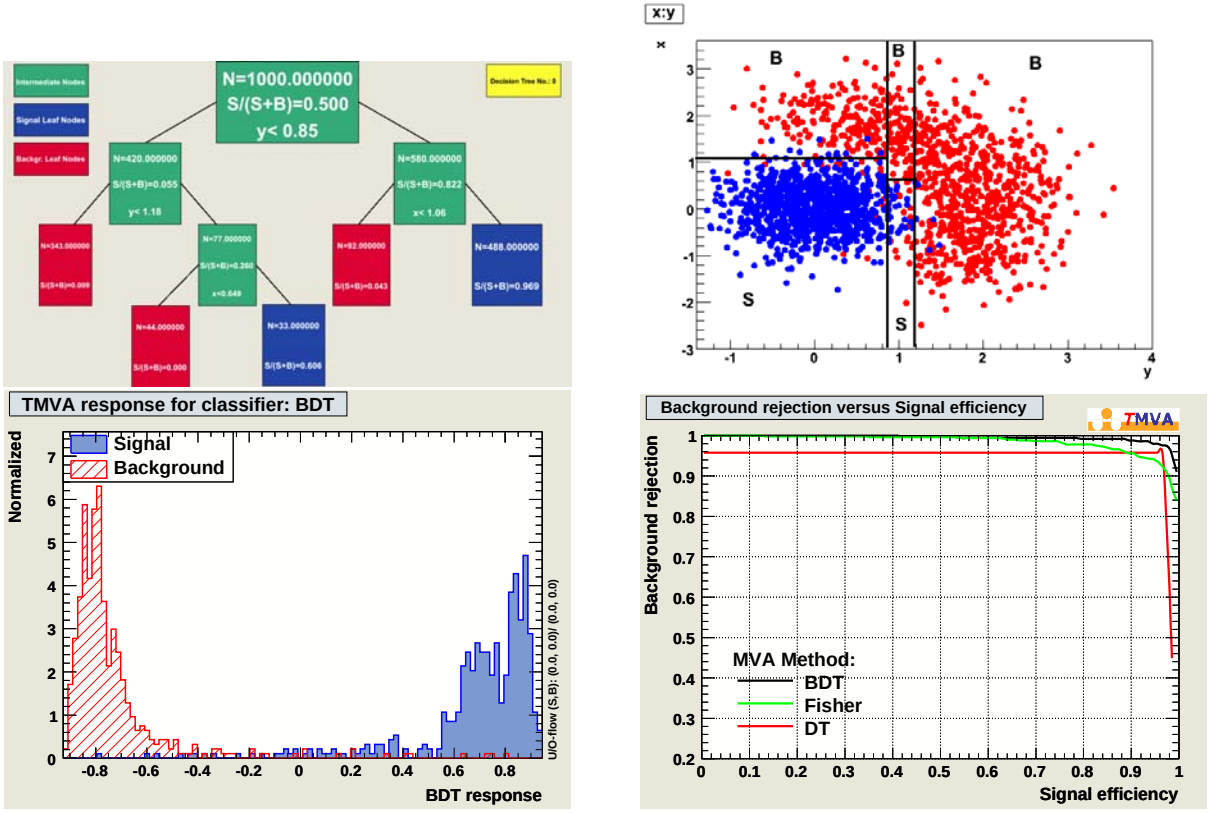


Figure 6.6: Top left: First decision tree built on x and y. Top right: Criteria applied in this first tree, shown in the $x : y$ plane; S (B) areas are labelled as signal (background). Bottom left: Output of the boosted decision trees. Bottom right: Background rejection vs. signal efficiency curves for the first decision tree (red), for the boosted decision trees (black) and for a Fisher discriminant analysis (green), all run on the same testing events.

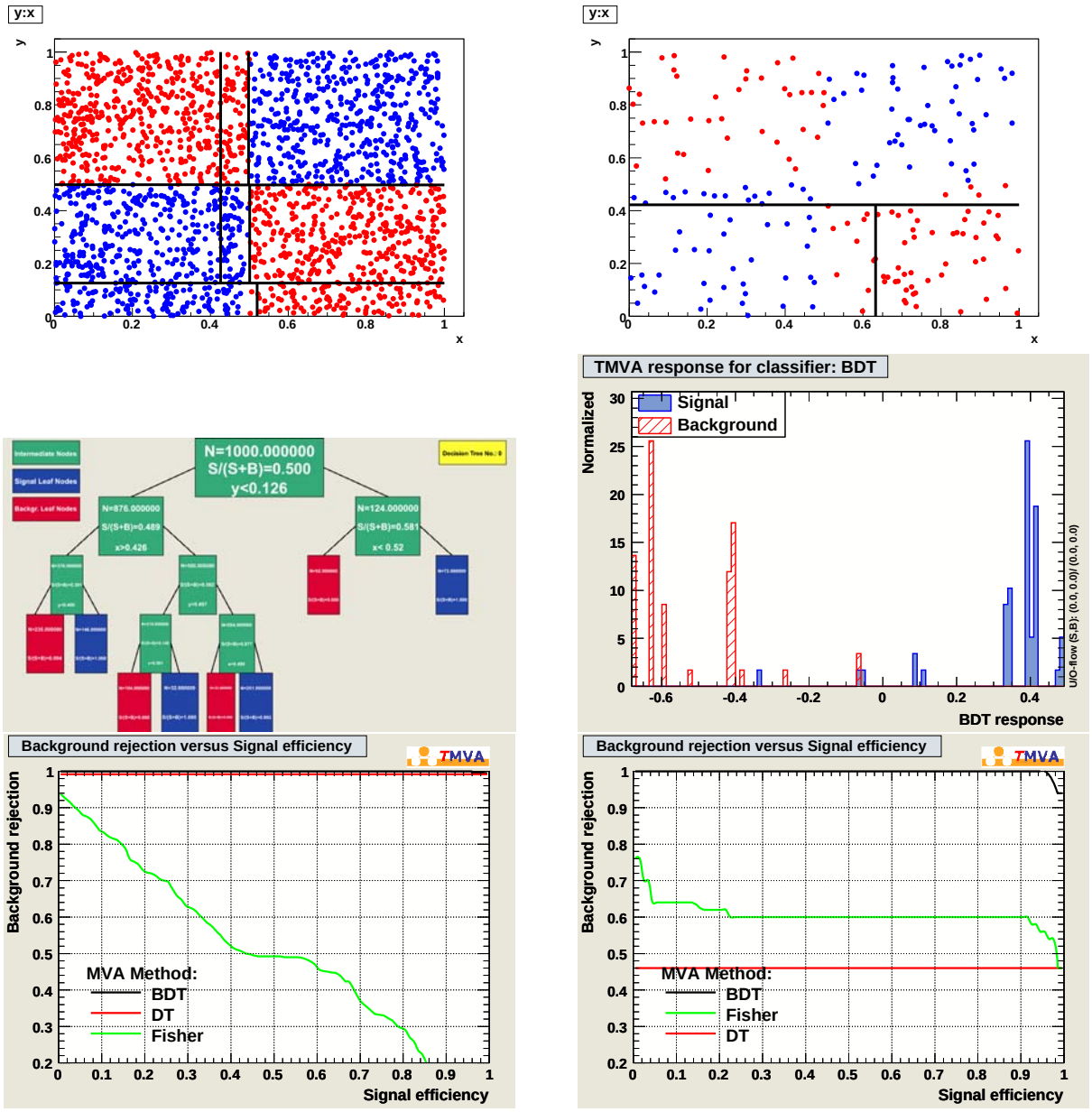


Figure 6.7: The XOR problem. Signal is in blue, background in red. The left column uses sufficient statistics, while the right column has a limited number of training events. The top plots show the signal and background distributions as well as the criteria used by the first decision tree. The middle left plot shows the first decision tree for the large statistics case. The middle right plot shows the boosted decision tree output for the limited statistics case. Bottom plots illustrate the background rejection vs. signal efficiency curves for the first decision tree (red), for the boosted decision trees (black) and for a Fisher discriminant analysis (green), all run on the same testing events.

Circular correlation

This example is derived from a dataset generated with the `create_circ` macro from `$ROOTSYS/tmva/test/createData.C`. The 1D and 2D representations of the two variables used are shown in Fig. 6.8.

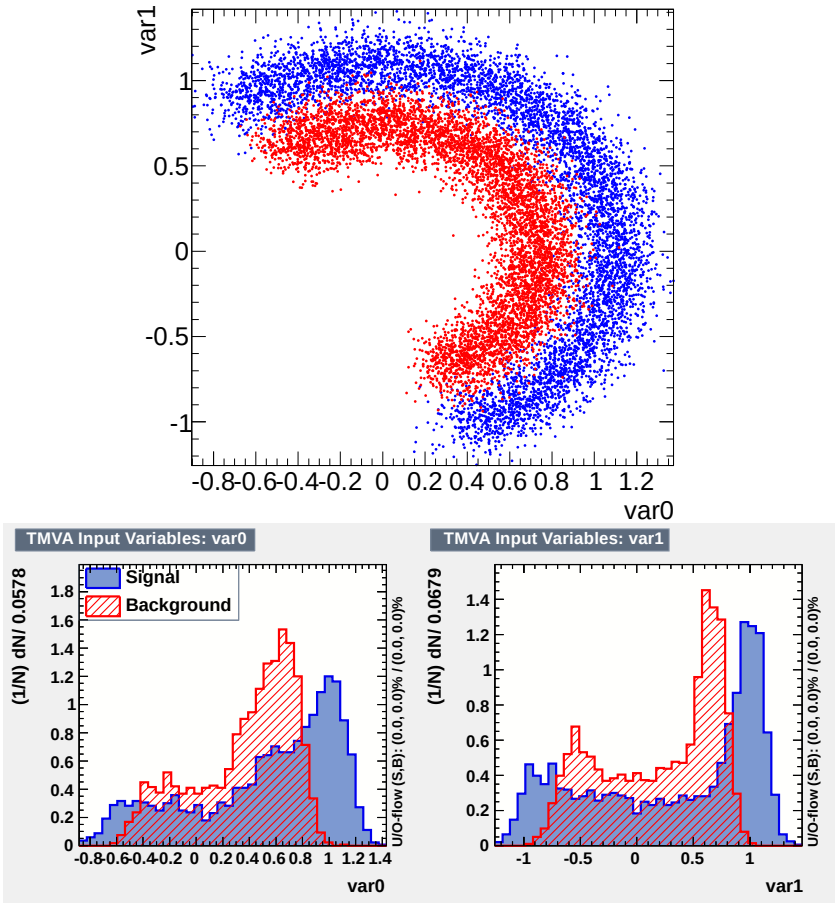


Figure 6.8: Dataset for the circular correlation example. Top: 2D representation. Bottom: 1D projections.

Several generic properties can be tested with this dataset: the impact of a longer boosting sequence (adding more and more trees), the meaningfulness of overtraining, the sensitivity to the splitting criterion and the difference between boosting many small trees or few big trees.

Figure 6.9 compares the performance of a Fisher discriminant (for reference), a single decision tree and boosted decision trees with an increasing number of trees (from 5 to 400). All other TMVA parameters are kept to their default value. One can see how a Fisher discriminant is not appropriate for non-linear correlation problems. The performance of the single tree is not so good, as expected since the default TMVA parameters make it very small, with a depth of 3 (it should be noted that a single bigger tree could solve this problem easily). Increasing the number of trees improves the performance until it saturates in the high background rejection and high signal efficiency corner. It should be noted that adding more trees doesn't seem to degrade the performance, the curve stays in the optimal corner. Looking at the contour plot one can however see that it wiggles a little for larger boosted decision trees, as they tend to pick up features of the training sample. This is overtraining.

Another sign of overtraining also appears in Fig. 6.10, showing the output of the various boosted decision trees for signal and background, both on the training and testing samples: larger boosted decision trees tend to show discrepancies between the two samples, as they adjust to peculiarities of the training

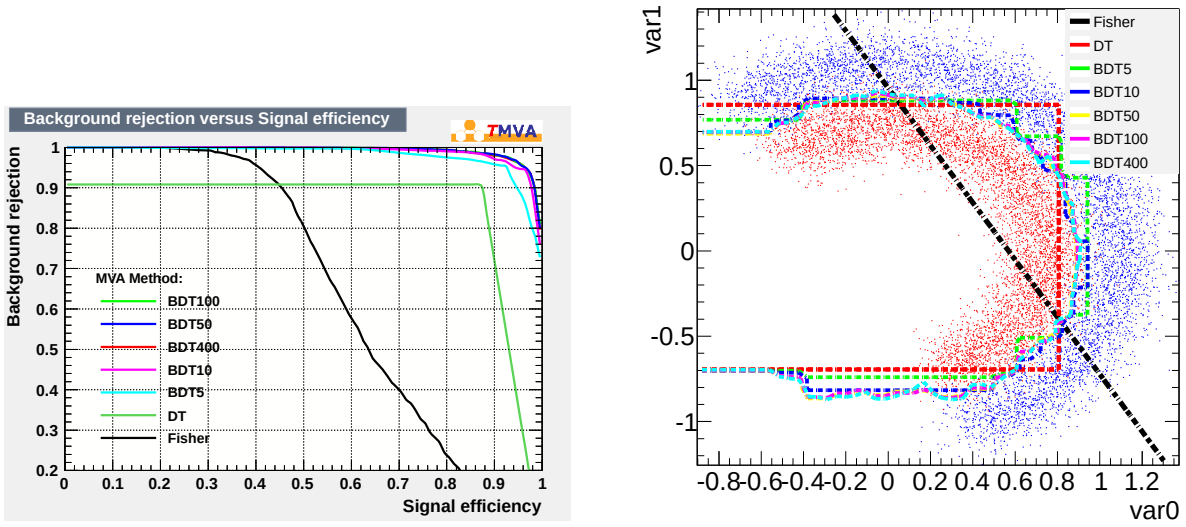


Figure 6.9: Circular correlation example. Left: Background rejection vs. signal efficiency curves for a Fisher discriminant (black), a single decision tree (dark green) and boosted decision trees with an increasing number of trees (5 to 400). Right: Decision contour corresponding to the previous discriminants.

sample that are not found in an independent testing sample. One can also see how the output acquires a "better" shape with more trees, really becoming quasi-continuous, which would allow to cut at a precise efficiency or rejection.

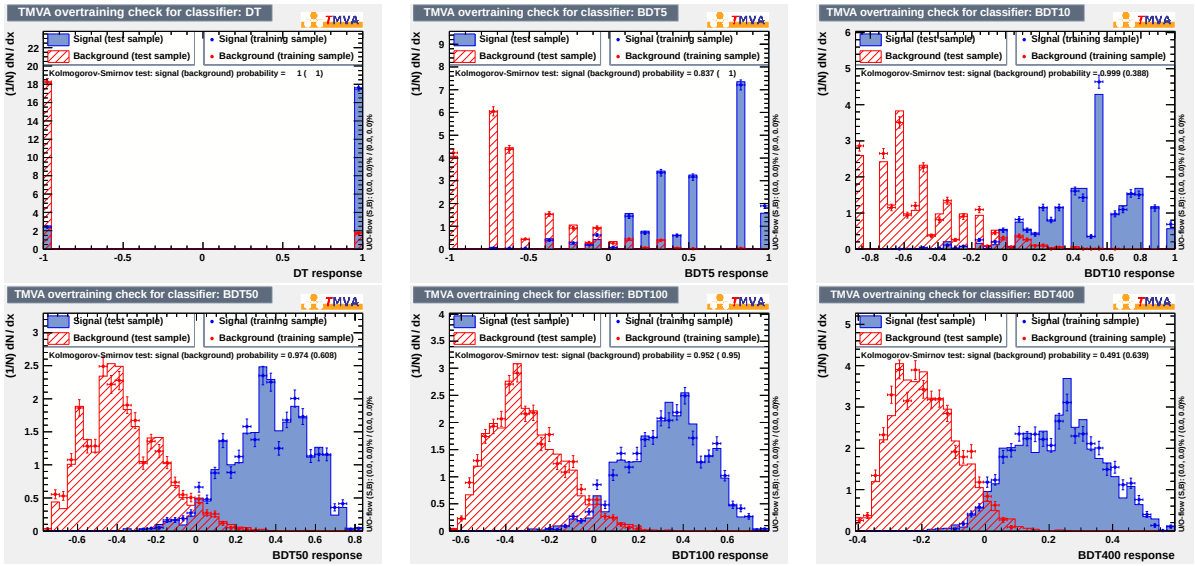


Figure 6.10: Circular correlation example. Comparison of the output on training (markers) and testing (histograms) samples for boosted decision trees with 1, 5, 10, 50, 100 and 400 trees (from top left to bottom right).

Both figures do exhibit clear signs of overtraining, but is it really an issue? As mentioned before what really matters in the end is the performance in data analysis. One way to evaluate this is to compute the significance $s/\sqrt{s+b}$. It is shown as the green curve in Fig. 6.11 for the same boosted decision trees as shown in Fig. 6.10, with increasing number of trees. The best significance is actually obtained with the 400-tree boosted decision tree! To be fair, the performance is very similar already with 10 trees. Now,

comparing the outputs, if interested in a smoother result, 10 trees might not be enough, but 50 would probably do, without the overhead of eight times more trees. Such a choice should in any case not be made based on overtraining statements, but rather on final expected performance.

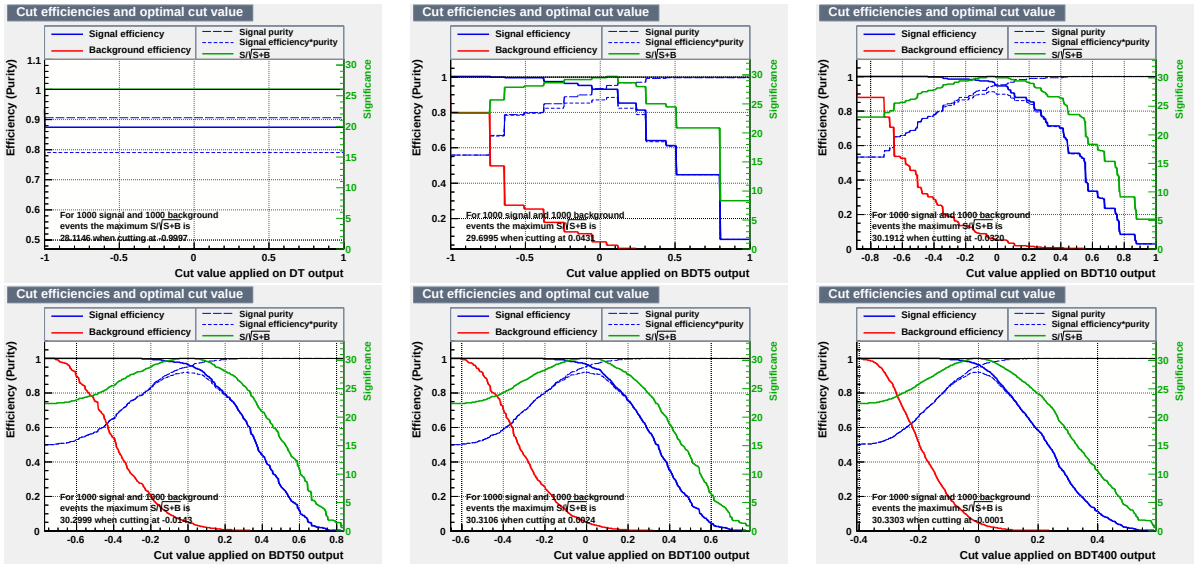


Figure 6.11: Circular correlation example. Signal efficiency, background efficiency, signal purity and significance $s/\sqrt{s+b}$ for boosted decision trees with 1, 5, 10, 50, 100 and 400 trees (from top left to bottom right).

This example can also be used to illustrate the performance of each tree in a boosting sequence. Figure 6.12 shows the rapid decrease of the weight α_k of each tree, while at the same time the corresponding misclassification rate ε_k of each individual tree increases rapidly towards just below 50%, that is, random guessing. It confirms that the best trees are the first ones, while the others are only minor corrections.

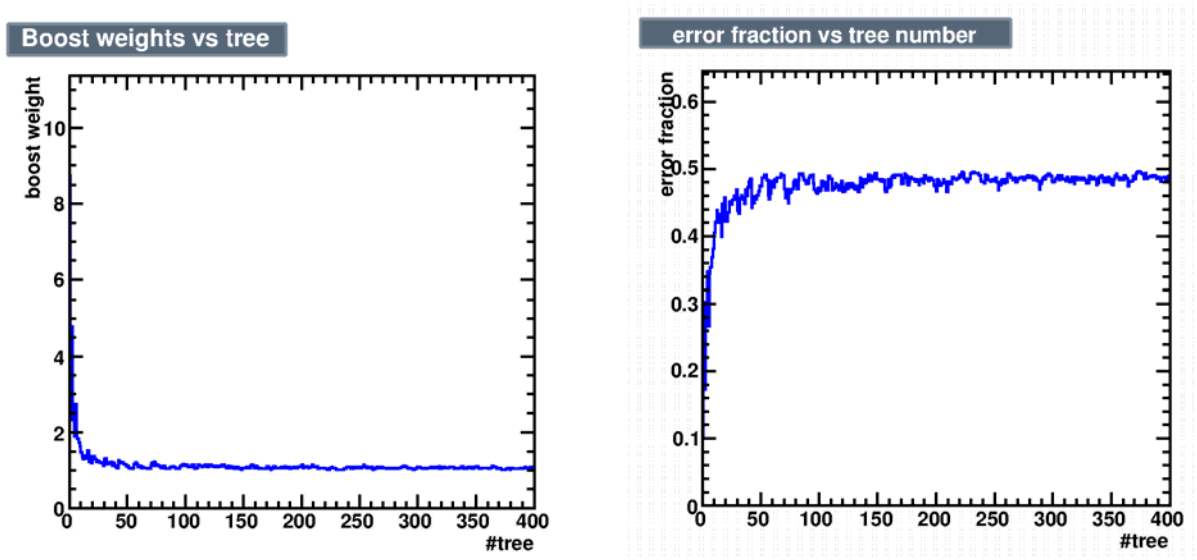


Figure 6.12: Circular correlation example, with a 400-tree boosted decision tree. Left: Boost weight of each tree. Right: Error fraction of each tree.

One can measure the impact of the choice of splitting criterion in Fig. 6.13. Results are shown for cross entropy, Gini index, Gini index with Laplace correction, misclassification error and $s/\sqrt{s+b}$ as separation criterion, while all other TMVA parameters are left at their default value. The top left plot

shows that they all have similar performance on this problem, even when zooming on the high efficiency side. One slight exception is the use of significance. It is also the only measure that doesn't respect some of the criteria suggested in Section 6.2.4. The other plots in the figure show the significance as a function of boosted decision tree output, which confirms that they all reach similar performance while the tree optimised with significance actually performs slightly worse than the others.

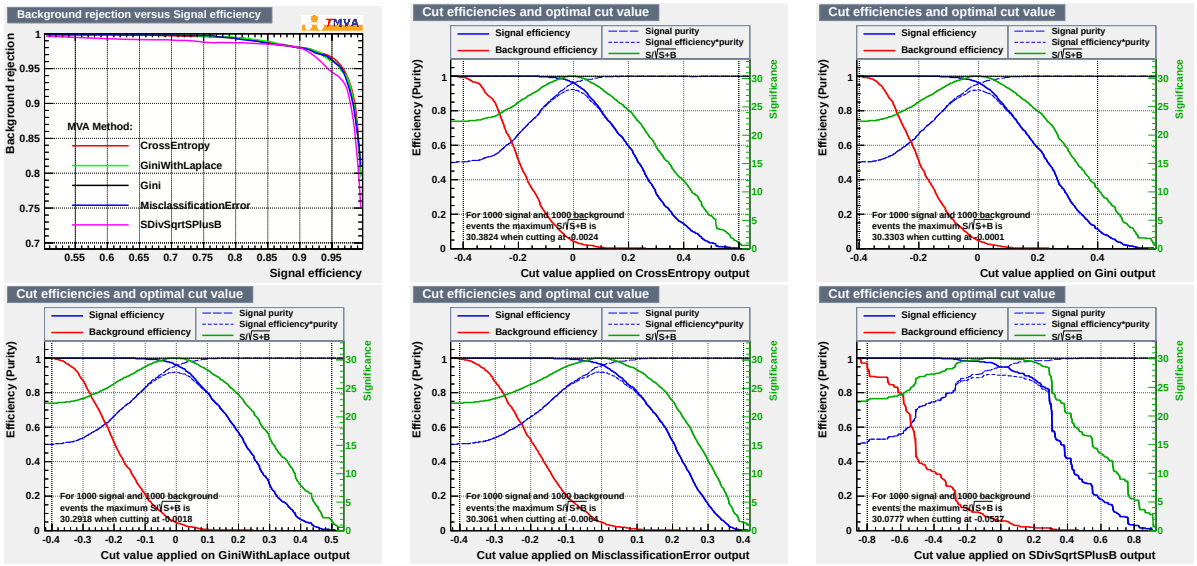


Figure 6.13: Circular correlation example: impact of the choice of splitting function. Top left: Background rejection vs. signal efficiency. From top middle to bottom left: Signal efficiency, background efficiency, signal purity and significance $s/\sqrt{s+b}$ for cross entropy, Gini index, Gini index with Laplace correction, misclassification error and $s/\sqrt{s+b}$.

A final illustration concerns the impact of the size of each tree in terms of number of leaves and size of each leaf, and their relation to the number of trees. In order to test this, the same `create_circ` macro was used, but to produce a much larger dataset such that statistics do not interfere with the test. All combinations of boosted decision trees with 20 or 400 trees, a minimum leaf size of 10 or 500 events and a maximum depth of 3 or 20 were tested, and results are shown on Fig. 6.14 as the decision contour and in terms of background rejection vs. signal efficiency curves. One can see an overall very comparable performance.

As is often the case for boosted decision trees, such optimisation on the number and size of trees, size of leaves or splitting criterion depends on the use case. To first approximation it is fair to say that almost any default will do, and optimising these aspects may not be worth the time required to test all configurations.

6.4.5 Other boosting algorithms

AdaBoost is but one of many boosting algorithms. It is also referred to as discrete AdaBoost to distinguish it from other AdaBoost flavours. The Real AdaBoost algorithm [12] defines each decision tree output as:

$$T_k(i) = 0.5 \times \ln \frac{p_k(i)}{1 - p_k(i)},$$

where $p_k(i)$ is the purity of the leaf on which event i falls. Events are reweighted as:

$$w_i^k \to w_i^{k+1} = w_i^k \times e^{-y_i T_k(i)}$$

and the boosted result is $T(i) = \sum_{k=1}^{N_{\text{tree}}} T_k(i)$. Gentle AdaBoost and LogitBoost (with a logistic function) [12] are other variations.

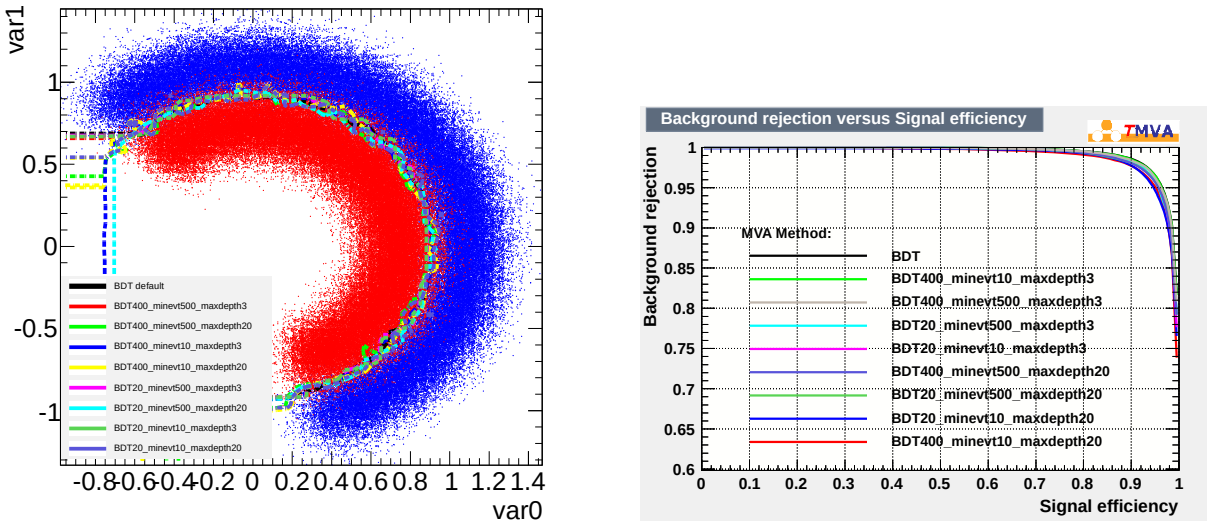


Figure 6.14: Circular correlation example with larger dataset and boosted decision trees with 20 or 400 trees, a minimum leaf size of 10 or 500 events and a maximum depth of 3 or 20. Left: Decision contour for each combination. Right: Background rejection vs. signal efficiency for each combination.

ε-Boost, also called shrinkage [16], consists in reweighting misclassified events by a fixed factor $e^{2\varepsilon}$ rather than the tree-dependent α_k factor of AdaBoost. ε-LogitBoost [9] is reweighting them with a logistic function $\frac{e^{-y_i T_k(i)}}{1+e^{-y_i T_k(i)}}$. ε-HingeBoost [9] is only dealing with misclassified events:

$$w_i^k \to w_i^{k+1} = \mathbb{I}(y_i \times T_k(i) \leq 0).$$

Finally one can cite the adaptive version of the "boost by majority" [7] algorithm, called BrownBoost [17]. It works in the limit where each boosting iteration makes an infinitesimally small contribution to the total result, modelling this limit with the differential equations that govern Brownian motion.

6.5 Other averaging techniques

As mentioned in Section 6.3.2 the key to improving a single decision tree performance and stability is averaging. Other techniques than boosting exist, some of which are briefly described below. As with boosting, the name of the game is to introduce statistical perturbations to randomise the training sample, hence increasing the predictive power of the ensemble of trees [5].

Bagging (Bootstrap AGGregatING) was proposed by Breiman [18]. It consists in training trees on different bootstrap samples drawn randomly with replacement from the training sample. Events that are not picked for the bootstrap sample form an "out of bag" validation sample. The bagged output is the simple average of all such trees.

Random forests is bagging with an extra level of randomisation [19]. Before splitting a node, only a random subset of input variables is considered. The fraction can vary for each split for yet another level of randomisation.

Trimming is not exactly an averaging technique per se but can be used in conjunction with another technique, in particular boosting, to speed up the training process. After some boosting cycles, it is possible that very few events with very high weight are making up most of the total training sample weight. One may then decide to ignore events with very small weights, hence introducing again some minor statistical perturbations and speeding up the training. ε-HingeBoost is such an algorithm.

These techniques, as was the case for boosting, are actually not limited to decision trees. Bagging applies to any training, random forests apply to any classifier where an extra level of randomisation is possible during the training, and trimming applies in particular to any boosting algorithm on any classifier. One could for instance build a boosted random forest with trimming.

6.6 Conclusion

In this lecture we have seen what decision trees are and how to construct them, as a powerful multivariate extension of a cut-based analysis. They are a very convincing technique, now well established in high energy physics. Advantages are numerous: their training is fast, they lead to human-readable results (not black boxes) with possible interpretation by a physicist, can deal easily with all sorts of variables and with many of them, with in the end relatively few parameters.

Decision trees are, however, not perfect and suffer from the piecewise nature of their output and a high sensitivity to the content of the training sample. These shortcoming are for a large part addressed by averaging the results of several trees, each built after introducing some statistical perturbation in the training sample. Among the most popular such techniques, boosting (and its AdaBoost incarnation) was described in detail, providing ideas as to why it seems to be performing so well and being very resilient against overtraining. Other averaging techniques were briefly described.

Boosted decision trees have now become quite fashionable in high energy physics. Following the steps of MiniBooNe for particle identification and D0 for the first evidence and observation of single top quark production, other experiments and analyses are now using them, in particular at the LHC. This warrants a word a caution. Despite recent successes in several high profile results, boosted decision trees cannot be thought of as the best multivariate technique around. Most multivariate techniques will in principle tend towards the Bayes limit, the maximum achievable separation, given enough statistics, time and information. But in real life resources and knowledge are limited and it is impossible to know *a priori* which method will work best on a particular problem. The only way is to test them. Figure 6.15 illustrates this situation for the single top quark production evidence at D0 [10]. In the end boosted decision trees performed only marginally better than Bayesian neural networks and the matrix elements technique, but all three analyses were very comparable, as shown by their power curves. The boosted decision tree analysis profited, however, of the fast turnaround of decision tree training in order to perform many valuable cross checks.

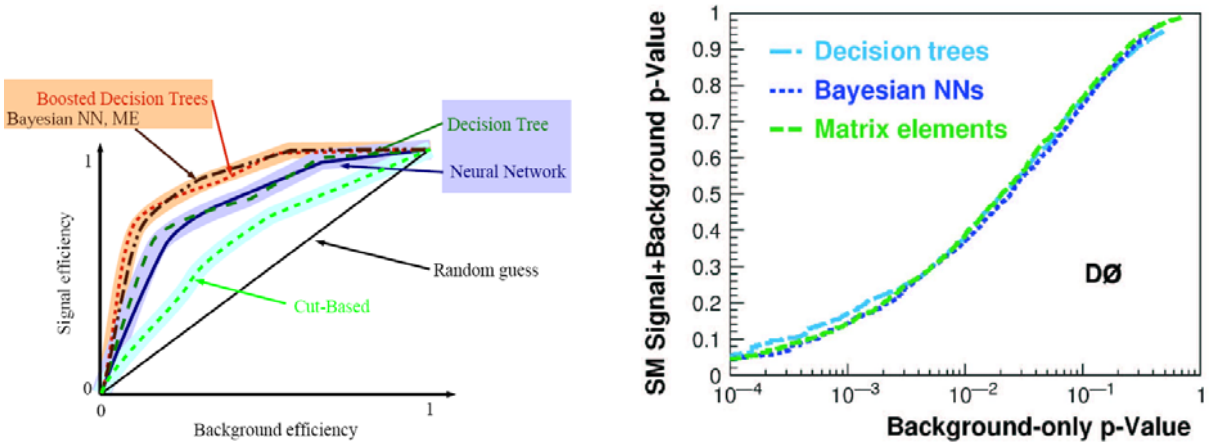


Figure 6.15: Comparison of several analysis techniques used in the D0 search for single top quark production [10]. Left: Signal vs. background efficiency curves for random guessing, a cut-based analysis, artificial neural networks, decision trees, Bayesian neural networks, the matrix elements technique and boosted decision trees. Right: Power curves (p-value of the signal+background hypothesis vs. p-value of the background-only hypothesis) for the boosted decision trees, Bayesian neural networks and matrix elements.

Finally, it cannot be stated often enough that using multivariate techniques is only the very last step of an analysis and is meaningful if and only if a proper model has been built that describes the data very well in all the variables one wishes to feed into the analysis.

6.7 Software

Many implementations of decision trees exist on the market. Historical algorithms like CART [1], ID3 [20] and its evolution C4.5 [21] are available in many different computing languages. The original MiniBoone [9] code is available at http://www-mhp.physics.lsa.umich.edu/∼roe/, so is the StatPatternRecognition [2] code at http://sourceforge.net/projects/statpatrec, and LHC experiments have various implementations in their software.

I would recommend a different approach, which is to use an integrated solution able to handle decision trees but also other techniques and flavours, allowing to run several of them to find the best suited to a given problem. Weka is a data mining software written in Java, open source, with a very good published manual. It was not written for HEP but is very complete. Details can be found at http://www.cs.waikato.ac.nz/ml/weka/.

Another recent development, now popular in HEP, is TMVA [14]. It is integrated in recent ROOT releases, which makes it convenient to use, and comes with a complete manual. It is also available online at http://tmva.sourceforge.net.

6.8 Acknowledgements

I would like to thank the organisers for their continuing trust in my lectures on decision trees. The school program, lectures, attendance and location were once again very enjoyable.

References

[1] L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone, *Classification and Regression Trees*, Wadsworth, Stamford, 1984

[2] I. Narsky, "StatPatternRecognition: A C++ Package for Statistical Analysis of High Energy Physics Data", arXiv:physics/0507143, 2005.

[3] C. Gini, "Variabilità e Mutabilità" (1912), reprinted in *Memorie di Metodologica Statistica*, edited by E. Pizetti and T. Salvemini, Rome: Libreria Eredi Virgilio Veschi, 1955.

[4] J.R. Quinlan, "Simplifying decision trees", *International Journal of Man-Machine Studies*, 27(3):221–234, 1987.

[5] H. Prosper, *Multivariate discriminants*, "Ensemble learning", EPJ Web of Conferences **4** 02001, 2010, SOS'08 - School of Statistics.

[6] R.E. Schapire, "The strength of weak learnability", *Machine Learning*, 5(2):197–227, 1990.

[7] Y. Freund, "Boosting a weak learning algorithm by majority", *Information and Computation*. 121(2):256–285, 1995.

[8] Y. Freund and R.E. Schapire, "Experiments with a New Boosting Algorithm" in *Machine Learning: Proceedings of the Thirteenth International Conference*, edited by L. Saitta (Morgan Kaufmann, San Francisco) p. 148, 1996.

[9] B.P. Roe, H.-J. Yang, J. Zhu, Y. Liu, I. Stancu, and G. McGregor, Nucl. Instrum. Methods Phys. Res., Sect.A 543, 577, 2005; H.-J. Yang, B.P. Roe, and J. Zhu, Nucl. Instrum. Methods Phys. Res., Sect. A 555, 370, 2005.

[10] V. M. Abazov *et al.* [D0 Collaboration], "Evidence for production of single top quarks and first direct measurement of $|V_{tb}|$", Phys. Rev. Lett. **98**, 181802, 2007; V. M. Abazov *et al.*, "Evidence for production of single top quarks", Phys. Rev. D**78**, 012005, 2008; V. M. Abazov *et al.*, "Observation of single top quark production", Phys. Rev. Lett. **103**, 092001, 2009.

[11] Y. Freund and R.E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[12] J.H. Friedman, T. Hastie and R. Tibshirani, "Additive logistic regression: a statistical view of boosting", *The Annals of Statistics*, 28(2), 377–386, 2000.

[13] H. Prosper, *Multivariate discriminants*, "Optimal classification", EPJ Web of Conferences **4** 02001, 2010, SOS'08 - School of Statistics.

[14] A. Höcker *et al.*, "TMVA: Toolkit for multivariate data analysis", PoS ACAT, 040, CERN-OPEN-2007-007, arXiv:physics/0703039, 2007.

[15] G. Cowan, "Multivariate statistical methods and data mining in particle physics", *CERN Academic Training Lectures*, June 2008. http://indico.cern.ch/event/24827

[16] J.H. Friedman, "Greedy function approximation: a gradient boosting machine", *The Annals of Statistics*, 29 (5), 1189–1232, 2001.

[17] Y. Freund, "An adaptive version of the boost by majority algorithm", *Machine Learning*, 43 (3), 293–318, 2001.

[18] L. Breiman, "Bagging Predictors", *Machine Learning*, 24 (2), 123–140, 1996.

[19] L. Breiman, "Random forests", *Machine Learning*, 45 (1), 5–32, 2001.

[20] J.R. Quinlan, "Induction of decision trees", *Machine Learning*, 1(1):81–106, 1986.

[21] J.R. Quinlan, *C4.5: programs for machine learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, 1993.

Part III

Tracking, limit extractions and unfolding techniques

Statistics for trajectometry

Pierre Billoir
LPNHE, Paris (IN2P3)

Abstract

A trajectometer is made of layers of charged particle detectors which measure successive positions along the trajectories; it is generally immersed in a magnetic field, so the curvature of the trajectory provides a measurement of the momentum. A method to perform a progressive fitting of the trajectory (Kalman Filter formalism), incorporating the measurements one after one, with an optimal account for the perturbations (multiple scattering, energy loss), is described with some indications for practical implementations in realistic detector layouts. Useful byproducts of the method and tests of validity are discussed. The procedure appears to be a combination *ad libitum* of elementary operations on vectors and matrices of fixed dimension (the number of parameters needed to define the trajectory), affording very flexible strategies, including a coupling of the pattern recognition of tracks with the fit of the trajectory, and combination with calorimetric or timing measurements. Extension to non-gaussian errors is discussed.

Once the trajectories of an event are independently reconstructed, they may be extrapolated back to the region of production of the particles (target, or zone of intersection of the beams in a collider) and associated to one or several vertices (primary interaction, and possible secondary interactions or decays): a fast and flexible method is described to perform these operations and improve the geometrical reconstruction, hence the kinematical one, by the constraint of a common origin; additional constraints may be added. Here again, the elementary steps consist in linear operations on vector and matrices of fixed dimension, allowing the user to easily proceed by successive trials and to optimize the strategy.

7.1 Introduction

7.1.1 What is a "trajectometer" ?

Most of the particle detectors include layers which aim at recognizing and reconstructing as precisely as possible the trajectories of the charged particles produced in the main interaction or in a secondary vertex, usually curved by a magnetic field to provide an information on the momentum. These layers are thin (in terms of radiation lengths) to avoid disturbing the movement. They are generally located close to the interaction point (inner part of the detector for an implementation around a collider, or in first position within a fixed target experiment), before the calorimetric components. However some elements may be set at remote positions to provide a "lever arm" effect for a precise measurement of the curvature: this is the case of external muon detectors, which give both a signature of muons and a relatively precise measurement of their position. A schematic layout of a trajectometer is shown in Fig.7.1, in the case on planar layers; for a detector installed around the intersection region of a collider, a better description is obtained by replacing the planes by cylinders.

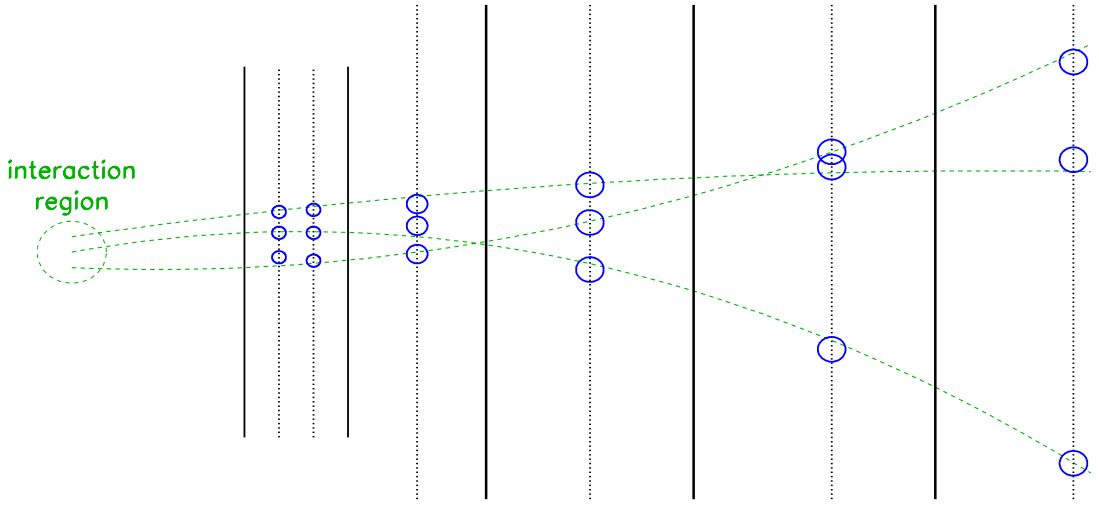


Figure 7.1: Schematic layout of a trajectometer. The green dashed lines are the trajectories of charged particles coming from the interaction region. The solid black lines are layers of material, the dotted black lines are measurement layers. The measurements are the blue circles (with a radius proportional to the uncertainty).

7.1.2 What is known and what is searched for ?

The following properties of the detector are supposed to be known (or determined at a previous stage from calibration data):

- the equation of propagation, which determines the shape of the trajectories. In practice, it is defined by the magnetic field map within the trajectometer.
- the *nature* and the *precision* of the measurements (drift time, amplitude on signals in strips or pads, etc). In simple cases they just give one coordinate or a fixed combination of coordinates. They may also depend on the direction of incidence on the detection layer. In practice, what is needed is to express the measured quantity as a function of the local parameters of the trajectory (position and direction). The distribution of errors is generally gaussian, but possible exceptions should be evaluated and accounted for.
- the nature and the amplitude of the "noise" processes (random perturbations of the trajectory): the multiple Coulomb scattering, which modifies the direction, and the fluctuations of the energy loss, which affect the curvature. These perturbations depend on the amount and the nature of the material crossed by the particles, but also on their momentum. As a consequence, a good

approximation of the momenta (or their inverse) has to be given by an external information (e.g. a calorimetric measurement), or by a first reconstruction where these perturbations are neglected.

In a first procedure (pattern recognition), a set of measurements is collected along the path of each particle. At this level there may be ambiguities or bad assignments, especially if there are many tracks, and even more if several interactions are "piled up" (this is the case in modern colliders). The next step (track fitting) may resolve some of these problems; actually, we will see that these two stages may be coupled into a mixed procedure including multiple trials and iterations. For the moment, we suppose that we have a clean collection of measurements along the trajectories of the charged particles. From this input, we want to extract the best estimation of:

- the properties of each particle (momentum, direction) at its origin. In practice, we want to obtain a backward extrapolation to the interaction region (the interior of the beam pipe for a collider), or to a secondary vertex of production. A magnetically curved trajectory may be described, when crossing a given reference surface (for example, for a given value of coordinate x), by 5 parameters: two for the position of the intersection with the reference surface (e.g. y, z for the example above) and 3 for the momentum vector (or the momentum and two angles). We suppose here that the mass is known or irrelevant; if not, several reconstructions may be needed, for different mass hypotheses.

- in some cases: a forward extrapolation towards external parts of the detector: e.g. the entry point into a calorimeter of into external muon chambers.

- the position and the composition of the primary vertex and secondary vertices within the detector, if any. This includes making one or several association of particles, and to use the constraint of a common origin to improve the reconstruction at the origin. Here again a procedure using iterative trials may be needed.

7.1.3 The problem of the noise

If the noise processes have negligible effects, we can choose a set of "initial" parameters $\mathbf{p}_0$ (position, direction, momentum) which gives a deterministic prediction of the expected measurements in layer k: $\bar{m}_k = F_k(\mathbf{p}_0)$. Because the measurement errors are independent we can write a χ^2 to be minimized:

$$\chi^2 = \sum_k \left(\frac{m_k - F_k(\mathbf{p}_0)}{\sigma_k} \right)^2$$

where m_k is the actual measurement, with an error σ_k. If needed (significantly non-gaussian errors), this may be replaced by a log-likelihood which is also a sum of independent terms. In most detectors the errors are small, so starting from a first approximation of the trajectory, F_k may be replaced by a linear function of the parameters and minimizing the χ^2 consists in solving a linear system of n_p equations, where n_p is the number of parameters. An iteration may be needed, but in general, the computation is fast.

The situation is more delicate if the errors induced by the noise are comparable to the measurement errors, or larger : a perturbation on the particle affects all measurements coming afterwards, so the measurements are no longer *independent*. The χ^2 should now be written as:

$$\chi^2 = \sum_{j,k} W_{jk}(m_j - F_j(\mathbf{p}_0))(m_k - F_k(\mathbf{p}_0))$$

where W is the inverse of the total covariance matrix, including the measurement errors and the noise induced errors, with non vanishing covariance terms: the measurements m_j and m_k are correlated through the perturbations occurring before both of them. This requires heavier computations; moreover, to make an optimal forward extrapolation we need to evaluate another covariance matrix, where m_j and m_k are correlated through the perturbations occurring *after* them.

In the following sections we apply to trajectometry an alternative method better suited to estimate the state of a dynamic system affected by random perturbations: the **Kalman Filter** (KF). The presentation

differs slightly from the traditional one, but it is mathematically equivalent. Here we use systematically a *weight matrix* and *weighted mean* formalism, which has the advantage of being more "compact", and hopefully more user friendly, in the sense that it relies on various combinations of very few intuitively understandable elementary operations on matrices and vectors. We describe with more or less details the operations within some frameworks (for example for some choices of parameters to define the trajectory), but we cannot give an exhaustive review of all possible implementations: the tools need to be adapted to a given context, after a clear understanding of the detector and the possible and useful approximations. In any case, the choices should be dictated by the gain in terms of the precision on physical quantities of interest for a further analysis.

7.2 The principle of the Kalman Filter (progressive fitting): a simple unidimensional example

7.2.1 The simplest model of measurement of a "noisy" process

We consider here a point with a random motion on a line. Its position $x(t)$ is measured without bias at times $1, 2, 3, \cdots$, with independent measurement errors of variance σ^2. The displacements between two measurements are 0 in average and independent, with the same variance τ^2 (for example, this is a brownian motion, as illustrated in Fig.7.2). The measurement errors are independent of the displacements.

Our problem is: if we have n successive measurements X_k of the true positions x_k, what is the combination of the X_k giving the best estimator of the initial position x_1, or the final one x_n ? and why not, the best estimator of any intermediate position x_k ? The solution is simple if τ^2 is negligible (at any time: the average of the measurements), or if τ^2 is very large compared to σ^2 (the best for x_k is just X_k, because the other measurements are too much disturbed). It is less clear if τ^2 is comparable to σ^2, or, more precisely, if $n\tau^2$ (the total variance of the displacements) is comparable to σ^2: in that case, the cumulated displacements and the measurement errors cannot be disentangled.

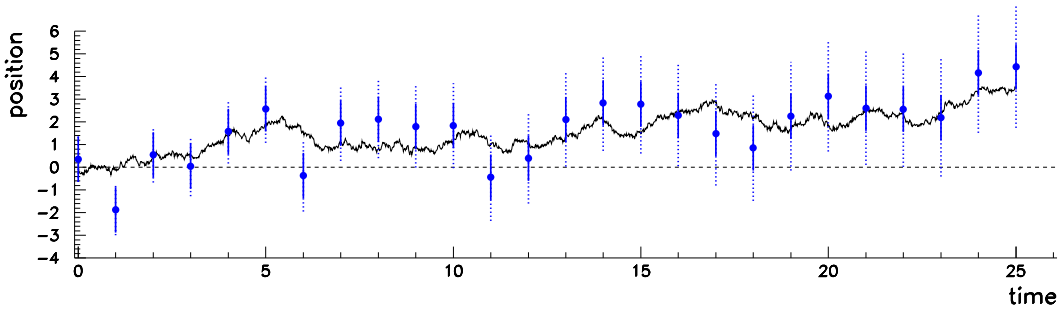


Figure 7.2: Measurements (blue points) of the position of a point moving randomly on a line. Here a brownian motion is simulated (solid line), and the variance of the displacement between two measurements is $\tau^2 = 0.25$. The solid bars represents measurement errors with $\sigma^2 = 1$; the dotted ones include the contribution of the displacements from the beginning, i.e. for point k it is $\sqrt{k\tau^2 + \sigma^2}$.

A first solution we may think about is the following: the difference between X_k and x_1 is the sum of $(k-1)$ independent displacements and one measurement error, so its variance is $V_k = (k-1)\tau^2 + \sigma^2$. We could make the *weighted average* of the X_k, with weights equal to $1/V_k$. This would be actually the best linear estimator, if the $X_k - x_1$ were *independent* random variables; but *this it not true* in our problem, because they include all displacements from the beginning: X_k and X_l both depend of the steps before k and l. As a consequence, if $l > k$, $cov(X_k, X_l) = k\tau^2$.

To build the best estimator in a standard way, we have to account for the $(n \times n)$ covariance matrix C of the X_k. A linear combination $\sum a_k X_k$ is an unbiased estimator of x_1 if $\sum a_k = 1$ and its variance is minimal if $\sum a_j a_k C_{jk}$ is minimal within the above constraint. That is, we have to solve a linear system of n equations; the solution may be written as a weighted mean of the X_k, with weights $w_k = \sum_j (C^{-1})_{jk} X_j$.

The complexity of the problem grows more than linearly with n. Moreover, if we want to evaluate the final position, or an intermediate one, we have to build another covariance matrix and then solve a different linear system.

Fortunately, there is another way to obtain the same results, with a number of operations *proportional* to n through the **Kalman Filter** methodology [1, 4] (*progressive fitting* [2, 3]).

7.2.2 Tools of the progressive fitting

The fundamental idea is to incorporate the measurements *one after one* in the algorithm, in such a way that *independent* random variables are combined at any stage. The procedure is illustrated on Fig.7.3, which may help to understand the simple operations behind the mathematical formulae.

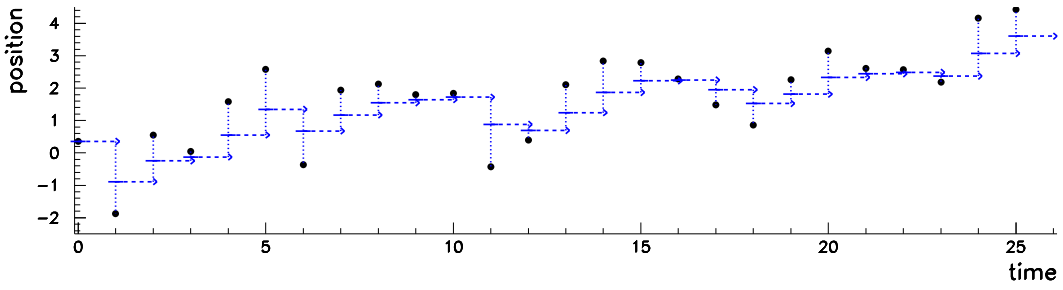


Figure 7.3: Principle of the forward filter, applied to the same dataset as in Fig.7.2 (the black points are the measurements X_k of the successive positions x_k). The blue solid line at time k represents $\tilde{X}_{1\to k,k}$, best estimator of x_k using the first k measurements (see text below), which is combined with the next measurement X_{k+1} to give $\tilde{X}_{1\to k+1,k+1}$, best estimator of x_{k+1}, and so on.

Let us consider the following elementary steps:

- $X_1 = x_1 + \varepsilon_1 = x_2 - \eta_1 + \varepsilon_1$, where ε_1 is the measurement error on point 1 (variance σ^2) and η_1 the displacement from time 1 to time 2 (variance τ^2). Because ε_1 and η_1 are independent, we see that $X_1 - x_2$ has a variance $\sigma^2 + \tau^2$, that is, X_1 is equivalent to a measurement of x_2 with variance $\sigma^2 + \tau^2$.

- By definition, $X_2 = x_2 + \varepsilon_2$ is another measurement of x_2, with variance σ^2. The crucial point is that ε_2 is *independent* of both ε_1 and η_1, so X_1 and X_2 are two *independent* measurements of x_2.

- The "best" linear combination of two independent measurements (that is, with the lowest variance) is their weighted mean with weights equal to the inverse of their variance. Here, this means that the "best" combination of X_1 and X_2 to estimate x_2 is:

$$\tilde{X}_{1\to 2,2} = \left(\frac{X_1}{\sigma^2+\tau^2} + \frac{X_2}{\sigma^2}\right) \Big/ \left(\frac{1}{\sigma^2+\tau^2} + \frac{1}{\sigma^2}\right) \quad \text{of variance} \quad \tilde{\sigma}^2_{1\to 2,2} = 1\Big/\left(\frac{1}{\sigma^2+\tau^2} + \frac{1}{\sigma^2}\right)$$

(where the notation $\tilde{X}_{l\to m,k}$ means: best estimator of x_k using measurements l to m).

In the case of gaussian errors, the steps described above may be interpreted as operations on a likelihood function $\mathcal{L}$: including only the measurement X_1 of x_1, $\mathcal{L}$ is a gaussian function; accounting for the random displacement consists in a *convolution* with another gaussian function; combining two independent measurements is a *multiplication* of the corresponding gaussian functions. Of course, maximizing $\mathcal{L}$ gives the same results as above. We will see later that this formulation can be extended to more complex cases.

The remarkable feature of this algorithm is that it can be iterated to include a further measurement (for convenience we replace $\tilde{\sigma}_{1\to 2,2}$ by $\tilde{\sigma}$):

- $\tilde{X}_{1\to 2,2}$ found above is a measurement of x_2 with variance $\tilde{\sigma}^2$, so it is a measurement of x_3 with variance $\tilde{\sigma}^2 + \tau^2$

- X_3 is another measurement of x_3, with variance σ^2, and $X_3 - x_3 = \varepsilon_3$ is independent of all random variables used to build $\tilde{X}_{1\to 2,2}$.

- so we obtain the best linear combination of X_1, X_2, X_3 to estimate x_3:

$$\tilde{X}_{1\to 3,3} = \frac{\frac{\tilde{X}_{1\to 2,2}}{\tilde{\sigma}^2+\tau^2} + \frac{X_3}{\sigma^2}}{\frac{1}{\tilde{\sigma}^2+\tau^2} + \frac{1}{\sigma^2}} \quad \text{of variance} \quad \frac{1}{\frac{1}{\tilde{\sigma}^2+\tau^2} + \frac{1}{\sigma^2}}$$

  Let us note that this may be written in a simpler way using the formalism of the *weight* (inverse of the variance) to express the combination as a *weighted mean*: [1]

$$\tilde{X}_{1\to 3,3} = \frac{\tilde{w}_{1\to 2,2}\tilde{X}_{1\to 2,2} + w_3 X_3}{\tilde{w}_{1\to 2,2} + w_3}$$

  The variance is $1/(\tilde{w}_{1\to 2,2} + w_3)$, in other terms: the weight of the combined estimator is just the sum of the weights of the two independent estimators.

We can in the same way incorporate the fourth measurement, and so on, and build $\tilde{X}_{1\to k,k}$ sucessively for all values of k. This is the "forward filter", which gives eventually the best estimator of the final position. It is clear that we can obtain the best estimator of the initial position through a similar formalism, starting from the last measurement and incorporating the other ones in reverse order. This is the "backward filter", giving, with our notations, $\tilde{X}_{k\to n,k}$ for any k. This is convenient if we have in mind a particle detector, where we are mainly interested to the initial parameters.

We can also build a $\chi^2_{\min}$ associated to each step of the procedure, in the usual way:

- combining X_1 and X_2 to estimate x_2, we have

$$\chi^2(X) = (X - X_1)^2/(\sigma^2 + \tau^2) + (X - X_2)^2/\sigma^2$$

  then $\chi^2_{\min} = (X_1 - X_2)^2/(2\sigma^2 + \tau^2)$

- at any further time $k+1$ we combine $\tilde{X}_{1\to k,k+1}$ ($\tilde{\sigma}^2_{1\to 2,2}$, $\chi^2_{min,k}$) with X_{k+1}. This gives a new value $\chi^2_{min,k+1} = \chi^2_{min,k} + (X_{k+1} - \tilde{X}_{1\to k,k})^2/(\tilde{\sigma}^2_{1\to 2,2} + \sigma^2)$

As usual, in the gaussian case, $\chi^2_{\min} = -2\ln(\mathcal{L}_{max})$, where $\mathcal{L}$ is the likelihood, omitting a constant normalization factor, and it follows the law of χ^2 with $N-1$ degrees of freedom, N being the number of measurements included.

The principle of one step of the filter is illustrated in Fig.7.4.

7.2.3 Useful byproducts

We can build further tools with very few more efforts: it is easy to obtain the best estimator of any intermediate position using all measurements, that is, to build the "smoother" in the KF terminology (optimal *interpolation*), by an adequate combination of the forward and backward filters. The forward filter gives $\tilde{X}_{1\to k,k}$ as an estimator of x_k with a variance that we call σ_f^2, and the backward one gives $\tilde{X}_{k+1\to n,k}$ with a variance $\sigma_b'^2$ (including a term τ^2 to go from $\tilde{X}_{k+1\to n,k+1}$ to $\tilde{X}_{k+1\to n,k}$) ; these estimators include independent measurement errors; the key point is that they also include *independent displacement errors* (the η_j with $1 \le j \le k$ for the forward one, $k < j \le n$ for the backward one). So they can be

[1] In the usual presentation of the KF, this step would be expressed as: find the best combination $\tilde{X}_{1\to 2,2} + \lambda(X_3 - \tilde{X}_{1\to 2,2})$, where λ is a "gain" coefficient. We prefer to use the weighted mean of independent variables, which will appear to be a universal tool for the next operations.

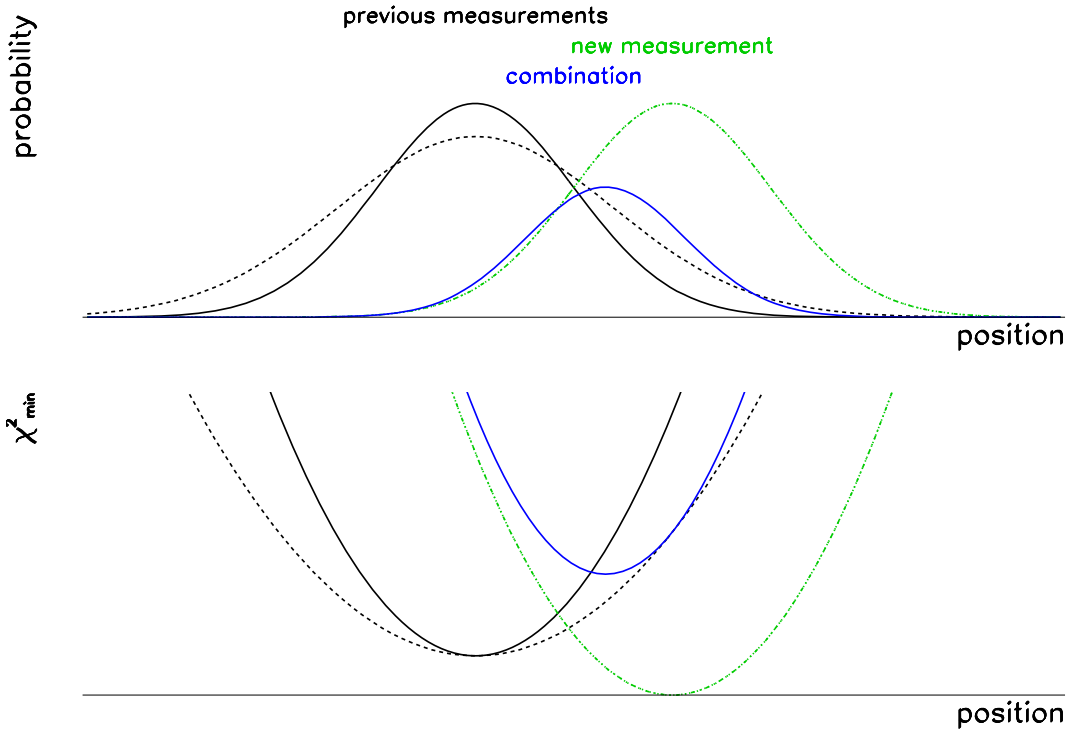


Figure 7.4: The estimator including all measurements up to a given point is represented in black (top: density of probability; bottom: minimum of χ^2), the dotted line accounts for the next random displacement. The green line represents the next measurement, and the blue one is the combination of these two independent estimators, which is the input for the next step.

simply combined as a weighted mean, defining $w_f = 1/\sigma_f^2$ and $w'_b = 1/{\sigma'_b}^2$:

$$\tilde{X}_{1\to n,k} = \frac{w_f \tilde{X}_{1\to k,k} + w'_b \tilde{X}_{k+1\to n,k}}{w_f + w'_b}$$

The same result is obtained by combining $\tilde{X}_{1\to k-1,p}$ and $\tilde{X}_{k\to n,k}$ with their proper variances (we just need to include X_k once and only once). We can also write an *exclusive* interpolation (without the measurement X_k) by combining $\tilde{X}_{1\to k-1,k}$ and $\tilde{X}_{k+1\to n,k}$: this will be useful for a discrimination of possible outliers, as discussed below.

Up to now we have assumed a perfect situation: both the displacements and the measurement errors have the expected distribution. In practice, in particle detectors, there may be deviations due to rare phenomena affecting the trajectory (e.g. a hard scattering), bad measurements or bad assignments of points to a given trajectory (especially if many particles cross the same detector element). It is useful to build tools to detect abnormal deviations ("outliers") and to define a strategy to get rid of them, as far as possible. Within our very simple model we can define two kinds of tests which can be extended to realistic situations:

- for a given time k, the difference $\tilde{X}_{1\to k,k} - \tilde{X}_{k+1\to n,k}$ between the forward and the backward filters has a predicted standard deviation $\sqrt{\sigma_f^2 + \sigma_b^2 + \tau^2}$. If $\sigma_f^2 + \sigma_b^2$ is smaller or comparable to τ^2, displacements which are large compared to τ may be detected; if not, the test can only discriminate very large deviations. If the distribution of the measurement and displacement errors are gaussian, a probability of χ^2 may be used as a discriminator.

- for time k, the measurement X_k and the exclusive interpolation (of variance σ_e^2) are independent,

therefore the variance of their difference is $\sigma_e^2 + \sigma^2$. Here again, if σ_e^2 is not too large compared to σ^2, outliers (abnormal measurements) may be detected. In the gaussian case, the probability of χ^2 gives a good discriminator.

Finally, let us remark that a large sample of clean measurements may be used to perform a *calibration* of the errors: if σ and τ are not known *a priori*, the distribution of the residuals mentioned above provide estimators for them, and possibly some hints on the distribution of errors.

7.2.4 Comments

The Kalman Filter was originally suited to dynamical problems like following the trajectory of engines from successive observations. In that case, the forward filter is the most natural tool: it can give in real time an updated estimation of the present state (position, direction, velocity). In applications to particle trajectometry, computations are not done in real time: even if some algorithms are implemented online, their input is a set of measurements coming after the particle has escaped the detector. Moreover the main purpose of the reconstruction of trajectories is to provide the best estimations at starting point (ideally, the vertex where this particle is originating from), so the backward filter is the basic tool. However, extrapolations to external detectors and interpolations may be needed, and discrimination of outliers is quite useful.

The number of operations to build the complete set of filters (forward, backward and smoother) is proportional to n if no outliers are removed. However, if a point k is to be removed after being compared to the interpolation, the forward filter has to be reevaluated at all points after k, and the backward filter at all points before k; the smoother has to be redone everywhere.

7.3 More complex examples

In these examples, we want to go progressively to the description of a real detector. In particular, we do not consider measurements labeled by times, but measurements of one or several coordinates as functions of one coordinate describing the successive layers of the detector. To simplify some expressions we will use the following notations for matrices A, B and vectors $\mathbf{q}$:

$$A\,[\mathbf{q}] = \mathbf{q}^T A \mathbf{q} \qquad A\,[B] = B^T A B$$

7.3.1 Straight line planar trajectory (2 parametres, linear model)

In this example (illustrated in Fig.7.5) the trajectory may be described as $y = ax + b$, and the coordinate y is measured as Y_k at x_k $(k = 1, x, \cdots n)$, with a variance σ_k^2. The noise consists in a random scattering (variation of the slope a) with a variance ρ_k^2 at each measurement layer ${x_k}^2$. All measurement and scattering errors are independent .

The parameters to be fitted are a and b; we call $\mathbf{p}$ the vector (b, a), or, more generally the vector of *local* parameters $(y, \mathrm{d}y/\mathrm{d}x)$ at any point of the trajectory. Let C be the (2×2) covariance matrix of an estimator, and W its inverse (the *weight matrix*). If all errors are gaussian, the log-likelihood is a quadratic function of $\mathbf{p}$:

$$-2\ln(\mathcal{L}) = -2\ln(\mathcal{L})_{max} + W\,[\mathbf{p} - \tilde{\mathbf{p}}]$$

where $\tilde{\mathbf{p}}$ is the best estimator. In the general case, we will build the "best" linear estimator (using all measurements Y_k up to a given position) through linear combinations using the matrices C and/or W at different stages of a progressive fitting procedure.

Before that, we will try to analyze this problem of estimating (a, b) with the standard method, by computing the variance/covariance of the measurements. We call ε_k the error on Y_k and ζ_k the variation

[2] We suppose ρ_k to be known *a priori*. For a physics interpretation in terms of multiple scattering, this means that the momentum is measured elsewhere; this model is just an intermediate step towards a complete trajectometer, aimed to introduce the fundamental tools.

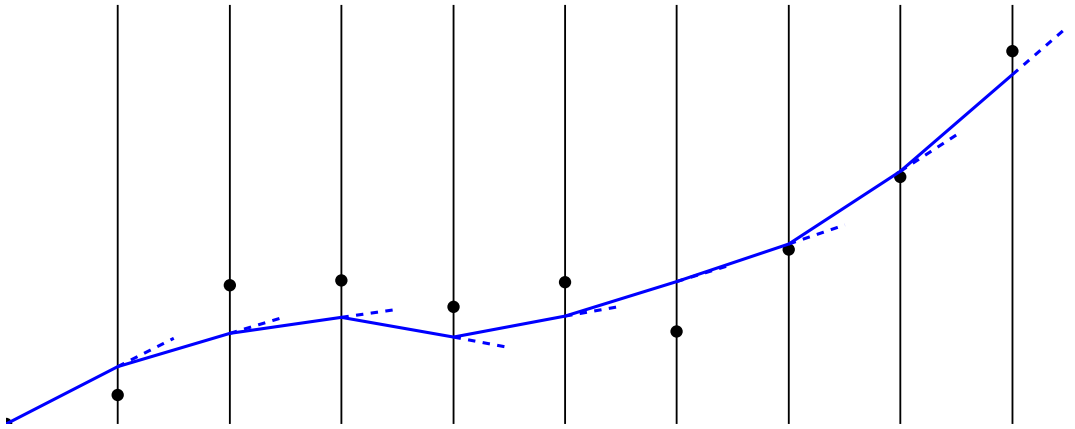


Figure 7.5: A planar trajectory made of straight line segments between measurement planes, where the slope is randomly modified (simple model for the multiple scattering on particles). The black points represent measurements.

of slope at x_k. Then:

$$Y_k = ax_k + b + \varepsilon_k + \sum_{j<k}(x_k - x_j)\zeta_j$$

$$var(Y_k) = \sigma_k^2 + \sum_{j<k}(x_k - x_j)^2\rho_j^2$$

$$cov(Y_k, Y_l) = \sum_{j<\inf(k,l)}(x_k - x_j)(x_l - x_j)\rho_j^2$$

The best linear estimator is obtained by minimizing the global χ^2:

$$\chi^2 = \sum_{j,k} W_{jk}(Y_j - ax_j - b)(Y_k - ax_k - b)$$

that is, we have to invert the $(n \times n)$ non-diagonal covariance matrix of the measurements.

Let us now describe one step of a progressive procedure; for convenience we will use *local* parameters (the value of y at x_k and the slope a). For the moment we suppose that we have built the best estimator $\tilde{\mathbf{p}}_{1\to k,k}$ (matrices $\tilde{C}_{1\to k,k}$, $\tilde{W}_{1\to k,k}$) using $Y_1, Y_2 \cdots Y_k$, associated the a minimum value $\tilde{\chi}^2_{1\to k,k}$, and we want to build $\tilde{\mathbf{p}}_{1\to k+1,k+1}$. We have to perform 3 operations:

- account for the scattering at x_k, by evaluating C for $\tilde{\mathbf{p}}_{1\to k,k}$ as an estimator of the parameters *after* the scattering. In our model, we just need to account for an additional error on a, so we compute:

$$\tilde{C}'_{1\to k,k} = \tilde{C}_{1\to k,k} + \begin{pmatrix} 0 & 0 \\ 0 & \tau^2 \end{pmatrix} \quad \text{and} \quad \tilde{W}'_{1\to k,k} = (\tilde{C}'_{1\to k,k})^{-1}$$

  The value of $\tilde{\chi}^2$ is not modified.

- *propagate* the estimator, going to the local parameters at x_{k+1}: we have $y_{k+1} = y_k + a(x_{k+1} - x_k)$, and the slope a is not modified. We write this simple transformation in a matrix formalism:

$$\tilde{\mathbf{p}}_{1\to k,k+1} = D_{k\to k+1}\ \tilde{\mathbf{p}}_{1\to k,k} \quad \text{with} \quad D_{k\to k+1} = \begin{pmatrix} 1 & x_{k+1} - x_k \\ 0 & 1 \end{pmatrix}$$

$$\tilde{C}'_{1\to k,k+1} = \tilde{C}'_{1\to k,k}\left[D^T_{k\to k+1}\right]$$

$$W'_{1\to k,k+1} = \tilde{W}'_{1\to k,k}\left[D_{k+1\to k}\right] \quad \text{using} \quad (D_{k\to k+1})^{-1} = D_{k+1\to k}$$

- *combine* $\tilde{\mathbf{p}}_{1\to k,k+1}$ with the *independent* information given by Y_{k+1}. The combined χ^2 is:

$$\chi^2(\mathbf{p}_{k+1}) = \tilde{\chi}^2_{1\to k,k} + \tilde{W}'_{1\to k,k+1}\left[\mathbf{p}_{k+1} - \tilde{\mathbf{p}}_{1\to k,k+1}\right] + \frac{(Y_{k+1} - \tilde{y}_{1\to k,k+1})^2}{\sigma_k^2}$$

We introduce now the 2-vector of *local measurement* $\mathbf{P}_{k+1} = (A, Y_{k+1})$ and its *weight matrix*

$$W^m_{k+1} = \begin{pmatrix} 1/\sigma_k^2 & 0 \\ 0 & 0 \end{pmatrix}$$

Actually A is not measured, but the expression of χ^2 does not depend on it; we can set it to 0 by convention. With these definitions we have to minimize:

$$\chi^2(\mathbf{p}_{k+1}) = \tilde{\chi}^2_{1\to k,k} + \tilde{W}'_{1\to k,k+1}\left[\mathbf{p}_{k+1} - \tilde{\mathbf{p}}_{1\to k,k+1}\right] + W^m_{k+1}\left[\mathbf{p}_{k+1} - \tilde{\mathbf{P}}_{k+1}\right]$$

The solution may be written as:

$$\tilde{\mathbf{p}}_{1\to k+1,k+1} = (\tilde{W}'_{1\to k,k+1} + W^m_{k+1})^{-1}(\tilde{W}'_{1\to k,k+1}\tilde{\mathbf{p}}_{1\to k,k+1} + W^m_{k+1}\tilde{\mathbf{P}}_{k+1})$$

which is an extension of the weighted mean found in the simplest model (here the weights are matrices). We still have the property of *additivity of weights*: the weight matrix of the combination is the sum of the weight matrices of the independent estimators.

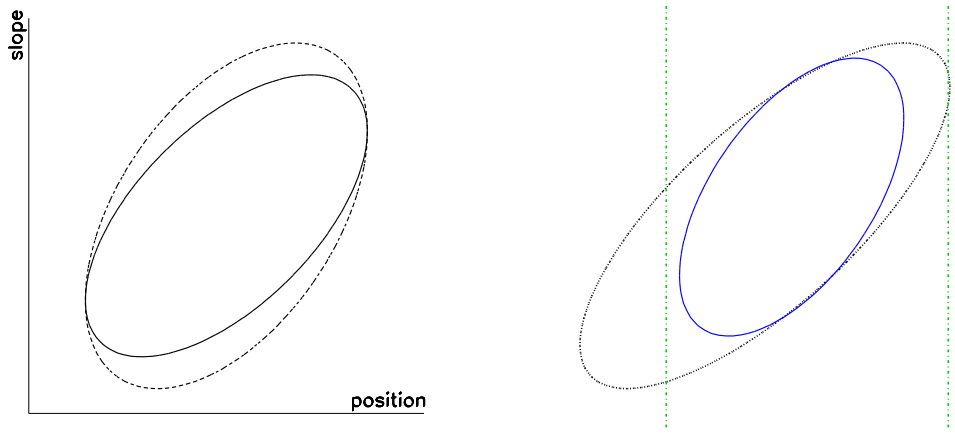


Figure 7.6: The operations of one step $k \to k+1$ of the filter (as in Fig.7.4) applied to a 2-parameter model (position and slope of a trajectory). The ellipses are the contours associated to one standard deviation around the central value. Left: solid: the "initial" estimator $\tilde{\mathbf{p}}_{1\to k,k}$; dashed: including the scattering. Right: black,dotted: propagated to point $k+1$; green, dash-dotted: measurement $\mathbf{P}_{k+1}$ at point $k+1$ (position only $\to$ vertical strip); blue: combination.

The formalism above may be completely expressed with elementary operations on "atoms" of information (one "atom" = vector of local parameters + weight matrix + minimal χ^2), but we have evaded some technical problems for a practical coding of the algorithm, especially: how to begin this recursive procedure. With the first measurement we have not enough information to define both parameters: we have seen that we can manage this situation by taking a singular weight matrix $diag(1/\sigma^2, 0)$ and a measurement vector with an arbitrary value for the first element (0 for example): the χ^2 valley may be seen a strip in the (a, b) plane, along the a direction. This "atom" may be propagated to the next position with the formalism above: the valley is still infinite, by now in an oblique direction w.r.t. the local parameters; when combined to a strip in the a direction (local measurement) it results in a finite region, and both $\tilde{C}$ and $\tilde{W}$ are regular.

The handling of the scattering remains to be clarified, because the operation $\tilde{C}' = \tilde{C} + C^{scat}$ cannot be

done if $\tilde{C}$ is not yet defined (first point). However we can rewrite this operation as $\tilde{W}' = (\tilde{W}^{-1}+C^{scat})^{-1} = (1+\tilde{W}C^{scat})^{-1}\tilde{W}$, which can be performed with one measurement only.

7.3.2 Further comments

The forward filter and the interpolators may be defined in the same way as in the simplest model. The tools to detect measurement outliers or abnormal scatterings may be built using χ^2 differences, with the same principles.

In the previous model the scattering occurred at the positions x_k where measurements were done. The formalism may be extended to any configuration: we just need to define the set of positions x_k where something happens (measurement or scattering) and to make the corresponding operations on $(\mathbf{p}, W, \chi^2)$ by increasing x (forward filter) or decreasing x (backward filter), with propagation operations in between. In this way forward of backward extrapolations may be done to account for material outside the range of measurements. An important application is the material between the vertex of origin and the first measurement (e.g. the beam pipe), to be accounted for in the analysis of the primary interaction.

A thick scattering region may be described, either as a set of elementary layers handled as above, or globally by computing the (2×2) matrix C^{scat}: at first order, the scattering through a thick material is fully described by terms to be added to the covariance matrix of y and a at a given reference position x_0, including a non-diagonal one to account for their correlation.

7.3.3 Curved planar trajectory (3 parameters)

Here we want to introduce a good approximation of a detector making measurements in a plane xy perpendicular to a magnetic field, for trajectories with a moderate angle w.r.t. the x axis, and a large radius of curvature R compared to the size of the detector. In that case we can describe the trajectory with a linear function of 3 parameters a, b, c:

$$y = ax + b + x^2/2R = ax + b + cx^2/2$$

We assume as above that y is measured at x_k $(k = 1, 2, \cdots n)$ with a variance σ_k^2. The 3-vector of local parameters is $\mathbf{p} = (y, \mathrm{d}y/\mathrm{d}x, \mathrm{d}^2y/\mathrm{d}x^2)$. In this model we can account for scatterings (random variations of $\mathrm{d}y/\mathrm{d}x$) and also for both deterministic and random variations of energy, which are expressed as variations of the curvature $\mathrm{d}^2y/\mathrm{d}x^2$. The formalism of the filters is exactly the same as in the previous model (plus a specific operation to modify the curvature parameter in case of energy loss), with a few technical differences:

- The matrix W is regular only when at least 3 points are included in the fit. In practice, it is of rank 1 with one point (the χ^2 valley is a slice in 3D space), 2 with 2 points (the χ^2 valley is a tube). But the same formalism as above may be used: a measurement is represented by a vector $(Y_k, 0, 0)$ with weight matrix $diag(1/\sigma^2, 0, 0)$ and in the initial vector of parameters undefined values are set to 0.
- In this model the momentum may be estimated from the fit itself so the variance of the scattering angles may be computed without external information; more precisely: the relevant curvature parameter c is proportional to the inverse of the momentum, with a geometrical sign depending on the physical sign of the charge, which is also to be determined. However, the curvature is not supposed to be known at the beginning of the filter. As a consequence, an iteration is needed: for example, the trajectory is fitted first ignoring the scattering, and the curvature found is injected in a second pass; if the curvature is significantly modified, a third pass may be needed. If the measurement range is too short to provide a significant estimation of the curvature, an external information is needed to use the noise formalism in the filters.

7.3.4 Realistic trajectory in space: using a linear approximation

In real detectors, no linear model (as the parabolic one) may represent the trajectories with the accuracy requested by the precision of the measurements. However, if the magnetic field is regular, one can choose

initial parameters such that the position and the direction of the trajectory in any measurement layer depends smoothly on them[3]. In these conditions a *reference trajectory* determined by initial parameters $\mathbf{p}_0$ may be defined as a first approximation, such that the functions F_k introduced in Sect.7.1.3 may be replace by a *linear expansion*:

$$F_k(\mathbf{p}_0 + \delta\mathbf{p}) = F_k(\mathbf{p}_0) + (\nabla F)_0.\delta\mathbf{p}$$

Let us take an example to illustrate a practical application of this method (and possible limitations): a circular trajectory in a (xy) plane (see Fig.7.7).

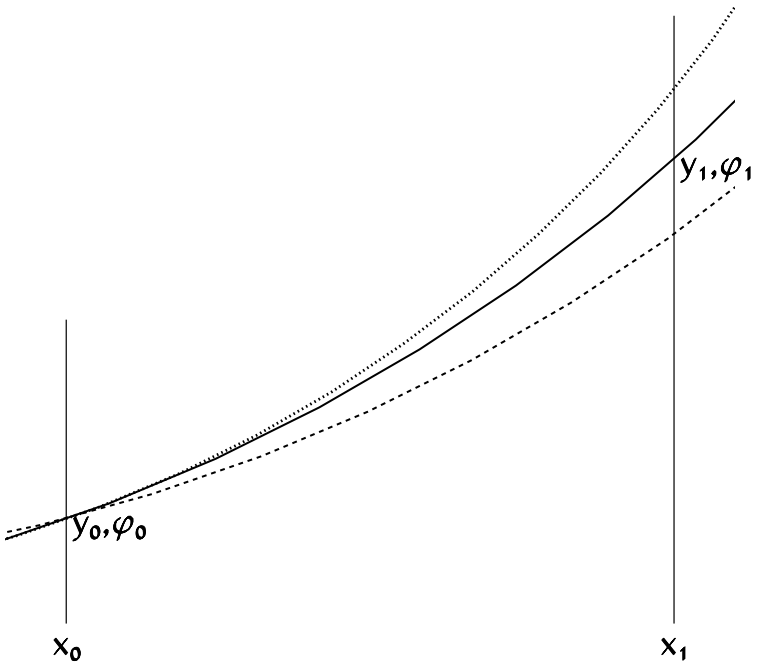


Figure 7.7: A planar circular trajectory measured at fixed x. Solid: the reference trajectory; dashed: with a variation of the initial direction φ_0; dotted: with a variation of the curvature c (changing the initial position y_0 gives a simple translation). The parameters at x_1 (y_1, φ_1) depend linearly on small variations of the initial parameters at x_0 .

The initial parameters (at $x = 0$) are $y_0, \varphi_0, c = 1/\mathcal{R}$, where φ is the local direction ($\tan\varphi = \mathrm{d}y/\mathrm{d}x$) and $\mathcal{R}$ the radius R with a *geometrical sign*: by convention we take + for an anticlockwise trajectory; in this model c is constant. With our convention we can write:

$$x = x_0 + \mathcal{R}(\sin\varphi - \sin\varphi_0) \qquad\qquad y = y_0 + \mathcal{R}(\cos\varphi - \cos\varphi_0)$$

First we want to evaluate the parameters y_1, φ_1, c at a fixed abscissa x_1. The first equation gives two solutions for φ_1 (or no one !), and we have to choose one of them. In general it is the nearest one to φ_0 (but an actual measurement can correspond to the second solution if it is within the detector...). Once the "right" solution (y_1, φ_1) is found, we want compute the derivatives of of y_1, φ_1 w.r.t. y_0, φ_0, c. Differentiating the first equation we obtain the derivatives of φ_1 (with $\Delta x = x_1 - x_0$):

$$\mathrm{d}\varphi_1 = \frac{\cos\varphi_0}{\cos\varphi_1}\mathrm{d}\varphi_0 - \frac{\Delta x}{\mathcal{R}^2\cos\varphi_1}\mathrm{d}\mathcal{R} \quad \rightarrow \quad \frac{\partial\varphi_1}{\partial\varphi_0} = \frac{\cos\varphi_0}{\cos\varphi_1} \;\; ; \;\; \frac{\partial\varphi_1}{\partial c} = \frac{\Delta x}{\cos\varphi_1}$$

Let us now differentiate the second one:

$$\mathrm{d}y_1 = \mathrm{d}y_0 - (\cos\varphi_1 - \cos\varphi_0)\mathrm{d}\mathcal{R} + \mathcal{R}(\sin\varphi_1\mathrm{d}\varphi_1 - \sin\varphi_0\mathrm{d}\varphi_0)$$

Injecting the expression of $d\varphi_1$ found above in this equation, we can extract after some algebra the

[3] this may be wrong for low energy particles at the end of their range, but in that case the contribution of the noise is so large that the precision of the measurement is not really significant, and a precise prediction of the trajectory is not needed.

derivatives of y_1:

$$\frac{\partial y_1}{\partial y_0} = 1 \qquad \frac{\partial y_1}{\partial \varphi_0} = \frac{\sin(\varphi_1 - \varphi_0)}{c \cos\varphi_1} \qquad \frac{\partial y_1}{\partial c} = \frac{1 - \cos(\varphi_1 - \varphi_0)}{c^2 \cos\varphi_1}$$

It is interesting to note that when R is large, c is a more convenient parameter than R or $\mathcal{R}$: in that case $\varphi_1 - \varphi_0$ is small and proportional to c, so all derivatives go to a finite limit when $R \to \infty$; moreover, the value $c = 0$ (straight line, infinite R) is not a singularity, and the geometrical sign may freely change during a progressive fit.

These expressions give us the expression of the 3×3 propagation matrix D similar to the 2×2 matrix defined in Sect.7.3.1. We give in parentheses the approximation for weakly curved trajectories (small $\Delta\varphi$), introducing the length of the arc between the two points $\ell = \Delta\varphi/c$

$$D = \begin{pmatrix} 1 & \frac{\sin(\Delta\varphi)}{c\cos\varphi_1}\ (\simeq \frac{\ell}{\cos\varphi_1}) & \frac{1-\cos(\Delta\varphi)}{c^2\cos\varphi_1}\ (\simeq \frac{\ell^2}{2\cos\varphi_1}) \\ 0 & \frac{\cos\varphi_0}{\cos\varphi_1}\ (\simeq 1) & \frac{\Delta x}{\cos\varphi_1}\ (\simeq \ell) \\ 0 & 0 & 1 \end{pmatrix}$$

This formalism should be used with care when the trajectory is close to a real singularity (quasi tangent to the measurement layer, that is $\cos\varphi_1 \simeq 0$): then the linear approximation is no longer valid, and moreover the actual meaning of the measurement may not be the coordinate y, and depend on the internal structure of the detection layer. In such a situation, it may help to redefine the parameters (e.g. take x at fixed y) and to express specifically the response of the detector under a skimming incidence; if this is not possible, the best solution is to ignore the measurement.

Once a convenient parametrization and a reference trajectory are found, the linear formalism using weight matrices may be applied to the vector $\delta\mathbf{p}$. The C and W matrices are the same for $\mathbf{p}$ and $\delta\mathbf{p}$. If the deviations from the reference are too large, it may be iteratively redefined until a satisfactory one is found. It is also possible to use different references for different parts of the trajectory; to go from one part to the next one, the parameters and their weight matrix need to be transformed through an operation similar to the propagation described in Sect.7.3.1 (see below).

7.3.5 Convenient parameters in usual detector configurations

We consider here two main categories of detectors: fixed target experiment or collider. In the first case the detection layers are mainly planes perpendicular to the beam axis (z coordinate) in the forward region, and possibly planes parallel to the beam around the target; in the second one there is a "barrel" part (cylinders of axis along z) and endcaps (planes perpendicular to z). Other configurations are possible, for example with "oblique" layers; this will be discussed later. If there is a magnetic field, we will use S/p to describe the curvature (p is the momentum, S the physical sign). In some cases (e.g. roughly uniform field along z) it is more convenient to use S/p_t (p_t is the transverse momentum).

Cartesian parameters

When the detectors are planes (e.g. at fixed z), a natural choice is x, y to describe the position within the plane, two slopes ($u = \mathrm{d}x/\mathrm{d}z$, $v = \mathrm{d}y/\mathrm{d}z$) or two angles to describe the local direction; for example a polar angle θ w.r.t. the z axis, and a azimuthal angle φ in projection onto the xy plane.

Cylindrical parameters

If the detection layers are cylinders around the beam axis, cylindrical coordinates (r, Φ, z) are natural parameters for the position: more precisely, the position at fixed r (in a detector layer) is defined by Φ, z (optionally $r\Phi, z$ for homogeneity). The direction may be given by θ, φ[4]. If the field is uniform and parallel to z axis, the trajectory is a helix of radius R. As in Sect.7.3.4, we use $\mathcal{R}$ with a geometrical sign

[4] To avoid confusions it is important to use different notations for Φ and φ for these two independent parameters.

(+ if the trajectory is anticlockwise in xy projection). Using a point x_0, y_0, z_0 on the trajectory and φ as a running parameter, the trajectory is defined by:

$$\begin{aligned} x &= x_0 + \mathcal{R}(\sin\varphi - \sin\varphi_0) = r_0\cos\Phi_0 + \mathcal{R}(\sin\varphi - \sin\varphi_0) \\ y &= y_0 - \mathcal{R}(\cos\varphi - \cos\varphi_0) = r_0\sin\Phi_0 + \mathcal{R}(\cos\varphi - \cos\varphi_0) \\ z &= z_0 + \mathcal{R}\cot\theta(\varphi - \varphi_0) \end{aligned}$$

The "perigee" parameters

It may be useful to summarize the information about the trajectory in one set of intrinsic parameters instead of using an arbitrary reference surface. In the case of quasi-uniform magnetic field along the beam axis (by convention the z axis), we can use the "perigee", point of closest approach to the z axis: if the particle originates from the main vertex, this point will be close to this vertex, so it will give a good approximation of the particle momentum. Another advantage is that a propagation of the trajectory and its error matrix to this point includes most of the material actually crossed by the particle (all material if the perigee is within a vacuum region, e.g. the beam pipe), so if this material is taken into account properly, the perigee parameters may be used in a further step of vertex fitting in a purely geometrical way, without accounting for noise: this will be exploited in Sect.7.6.

The trajectory is defined by 5 parameters (see Fig.7.8): the cylindrical coordinates of the perigee $(\varepsilon, \Phi_p, z_p)$, the signed curvature $c = 1/\mathcal{R}$ and θ. To avoid discontinuities around the origin when extrapolating a trajectory towards the interaction region, it is convenient to give a geometrical sign to ε: by convention it is positive if the origin O is on the right hand side of the trajectory, and Φ_p is defined as $\varphi_p + \pi/2$. With this convention, we have always $x_p = \varepsilon\cos\Phi_p$, $y_p = \varepsilon\sin\Phi_p$, and the trajectory may be parametrized as:

$$\begin{aligned} x &= \varepsilon\cos\Phi_p + \mathcal{R}(\sin\varphi - \sin\varphi_p) = \varepsilon\cos\Phi_p + \mathcal{R}(\sin\varphi + \cos\Phi_p) \\ y &= \varepsilon\sin\Phi_p - \mathcal{R}(\cos\varphi - \cos\varphi_p) = \varepsilon\sin\Phi_p - \mathcal{R}(\cos\varphi - \sin\Phi_p) \\ z &= z_p + \mathcal{R}\cot\theta\,(\varphi - \varphi_p) = z_p + \mathcal{R}\cot\theta\,(\varphi - \Phi_p + \pi/2) \end{aligned}$$

In the vertex fitting procedure, we need in principle short range extrapolations from the perigee, so we can use the second order approximation in $\ell = \mathcal{R}(\sin\varphi - \sin\varphi_p)$ (distance from perigee in xy projection):

$$\begin{aligned} x &= \varepsilon\cos\Phi_p + \ell\sin\Phi_p + c\,\ell^2\cos\Phi_p \\ y &= \varepsilon\sin\Phi_p - \ell\cos\Phi_p + c\,\ell^2\sin\Phi_p \\ z &= z_p + \ell\cot\theta \end{aligned}$$

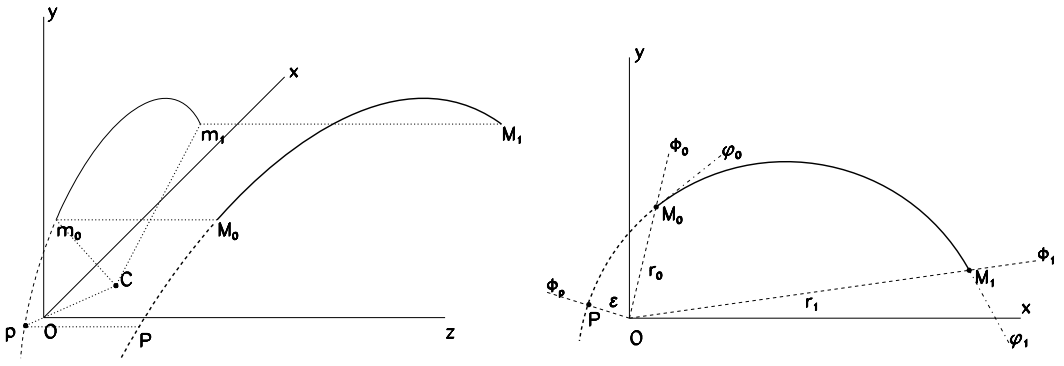


Figure 7.8: Left: the trajectory in 3D-space (helix of axis along z). Solid: the measured portion M_0M_1, dashed: the extrapolation to the perigee P. Right: The projection onto the xy (circle). The position of a point along the trajectory is defined by r, Φ; the direction of the tangent is defined by φ. In this example, the signed radius $\mathcal{R}$ is negative (φ decreases with increasing r), and the perigee distance ε is positive.

The perigee parameters will also be used as a technical tool to compute the derivative matrices needed to propagate the error matrix in cylindrical coordinates (see Appendix).

Propagation of error matrices

In the helix model for trajectories, the analytical computation of the matrix of derivatives D was done in Sect.7.3.4 for cartesian parameters. Using similar techniques, on can obtain analytical expression for cylindrical parameters, as function of the parameters at the initial and the final point. The computation is developped in Appendix.

If the magnetic field is not perfectly uniform, the trajectory has to be propagated with a precision better than the measurements, so a numerical computation (or a perturbative expansion) may be needed. However, the derivatives may be taken from the analytical expressions, because they give a sufficient approximation for the propagation of errors.

Local change of parametrization

To follow the disposition of the detector layers, it may be convenient to modify the parametrization at a certain point of the trajectory. For example, with the helix model in cartesian coordinates, we use parameters $(x, y, \theta, \varphi, c)$ at fixed z in the forward region, and $(x, z, \theta, \varphi, c)$ at fixed y in the lateral region. Here using the same notation x for a parameter with a different meaning may be a source of confusion: the transformation of the error matrix should account for a non trivial transformation on the position parameters: an elementary variation δy of y in a plane $z = z_0$, at given values of x, θ, φ, c, is a translation which results in a displacement of the intersection of the trajectory with a plane at fixed y; using the notation $a|_b$ for "a at fixed b", and noting $\mathbf{u}$ the unit vector along the trajectory, the variations of the coordinates $x|_y, z|_y$ are:

$$\delta x|_y = -\frac{u_x}{u_y}\,\delta y|_z = -\cot\varphi\,\delta y|_z \quad ; \quad \delta z|_y = -\frac{u_z}{u_y}\,\delta y|_z = -\frac{\cot\theta}{\sin\varphi}\,\delta y|_z$$

For the same reason, if the curvature is not negligible, a variation of $y|_z$ will affect the direction (actually, only φ) at fixed y; if we call $\delta\ell$ the displacement inxy projection, we have from the helix model:

$$\delta\varphi = c\,\delta\ell = -\frac{c}{\sin\varphi}\,\delta y$$

The jacobian matrix of the transformation from $(x, y, \theta, \varphi, c)|_z$ to $(x, z, \theta, \varphi, c)|_y$ is then:

$$D_{\rm loc} = \begin{pmatrix} 1 & -\cot\varphi & 0 & 0 & 0 \\ 0 & -\frac{\cot\theta}{\sin\varphi} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -\frac{c}{\sin\phi} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The covariance and the weight matrix are modified as $C^{\rm new} = C^{\rm old}\left[D_{\rm loc}^T\right] = D_{\rm loc}\,C^{\rm old}\,D_{\rm loc}^T$ and $W^{\rm new} = W^{\rm old}\left[D_{\rm loc}^{-1}\right] = (D_{\rm loc}^{-1})^T\,W^{\rm old}\,D_{\rm loc}^{-1}$.

For any other change of local parameters, a similar study has to be done to define the jacobian matrix.

7.3.6 Indirect measurements of parameters ("oblique" projection)

Up to now we have represented a layer of by a simple surface (e.g. a plane of wires). The quantity actually measured by a detector is supposed to depend on the position of the intersection of the trajectory with this surface, but it may also depend on the direction of incidence. Let us take two examples:

- In a plane $z = z_0$, we mesure the distance of closest approach of the trajectory to a wire at $x = x_0$ (see Fig.7.9, left). Using the parameters x and $a = {\rm d}x/{\rm d}z$, and assuming that the curvature is

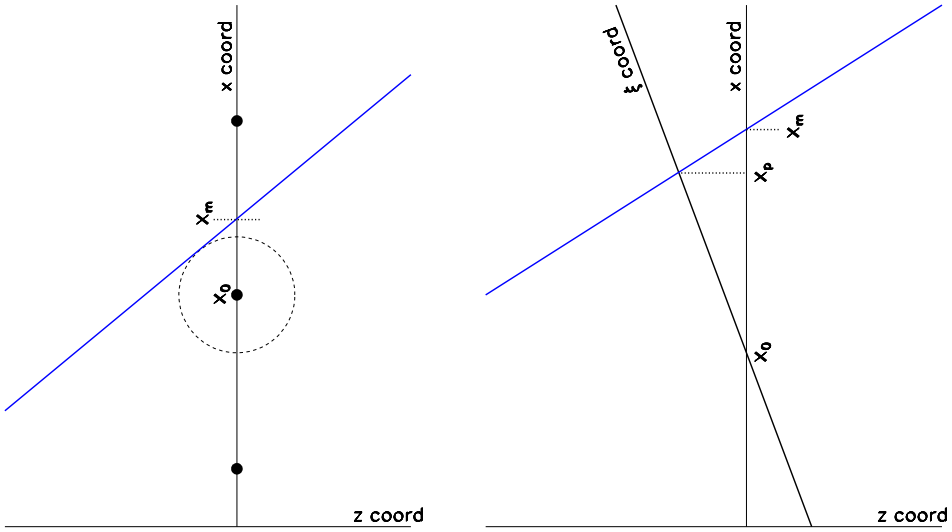


Figure 7.9: Examples of "oblique" measurements. The trajectory (blue line) has a slope $a = \mathrm{d}x/\mathrm{d}z$. In both cases, x_m represents the coordinate effectively measured in the plane $z = z_0$ through a raw measurement in the detector. Left: the raw measurement is the distance to a line parallel to y axis (e.g. a wire). Right: the raw measurement ξ is taken in a detector inclined by α (thick black line); we see on the figure that $x_p - x_0 = \xi \cos\alpha$ and $x_m - x_p = a\,\xi \sin\alpha$.

negligible at this scale, this means that we measure $d = |x - x_0|/\sqrt{1+a^2}$, with a precision σ. Within a good approximation, we can take the reference value $a_{\rm ref}$ of the slope, and consider that we measure $x = x_0 \pm d\sqrt{1+a_{\rm ref}^2}$ with a precision $\sigma_x = \sigma\sqrt{1+a_{\rm ref}^2}$ (at this level there may be an ambiguity if the extrapolation provided by the filter is not precise enough).

- The detector surface is not exactly perpendicular to z axis (see Fig.7.9, right). For example, we measure a coordinate ξ in a plane inclined by α on the xy plane, intersecting the plane $z = z_0$ at $x = x_0$ ($\xi = 0$ at the intersection). We obtain $x - x_0 = (\cos\alpha + a_{\rm ref}\sin\alpha)\xi$. Here again we apply to the measurement a factor depending on the local direction.

The real situation may be more complex. For example, in a drift chamber, the relation between the measured time and the local parameters depends on the position (close to the wire or far away). Or we may have to consider in a barrel detector (with cylindrical parameters) detector elements which are planar. In any case, the prescription is to write the local parameter to be measured as a linear function of the quantity which is actually measured, with coefficients depending on the local direction of the trajectory.

7.3.7 Composite measurements

Some detectors (e.g. chambers with tilted wires) provide a measurement of a "composite" coordinate, e.g. a quantity $u = \alpha x + \beta y$ at fixed z, measured with a precision σ. The formalism of "atoms" introduced in Sect.7.3.1 is very convenient to account for such measurement u_m: it is equivalent to a vector $\mathbf{P} = (x_m, y_m, \cdots)$, where x_m, y_m are any values such that $u_m = \alpha x_m + \beta y_m$, the other components being arbitrary, with a weight matrix of rank 1 (written here with 5 parameters):

$$W = \frac{1}{\sigma^2}\begin{pmatrix} \alpha^2 & \alpha\beta & 0 & 0 & 0 \\ \alpha\beta & \beta^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

More generally, let us suppose that we measure in a detector surface a set of n quantities $\mathbf{U} = (u_1, u_2 \cdots u_n)$ ($n \leq 5$) that can be expressed locally as a linear combination of the $\delta\mathbf{p}$:

$$\mathbf{U} = \mathbf{U}_0 + M\delta\mathbf{p}$$

The errors on the measurements of $u_1, u_2 \cdots u_n$ may be independent or not. Let us call C_U their covariance matrix, and $W_U = C_U^{-1}$ their weight matrix. We can introduce in the filter formalism an "atom" made with $\delta\mathbf{p}_m$ (any values compatible with the n measurements) and a weight matrix $W_m = M^T W_U M$ of rank n. The result of the weighted means does not depend on the arbitrary choices made to build $\mathbf{p}_m$.

7.3.8 Exogenous measurements

We call "exogenous" a measurement coming from a detector which does not belong to the trajectometer, or an information coming from an element of the trajectometer, but of a different nature. In the first category we can include a calorimetric measurement of the energy (if a matching can be established): this may be useful to compute an initial curvature parameter when starting the backward filter (but the ambiguity on the sign has to be solved). In the second category we may have a timing information, or an evaluation of the ionization rate, that can constrain the momentum, or solve the mass ambiguity. When written as a linearizable function of the local parameters, these measurements can be handled in the same way as the composite measurements above.

7.3.9 Comments on practical implementation

A big advantage of the Kalman Filter formalism is to rely on linear operations on vectors and matrices of *fixed dimension* (number of parameters needed to describe the trajectory), whatever the number of measurements and noise sources. The implementation is computationally efficient if these operations are explicitely coded, without calling functions from a general matrix package. Moreover useful approximations may result in sparse matrices, reducing even more the computations needed. As a consequence, such procedures could even be introduced in high level triggers.

7.4 Coupling the pattern recognition to the track fit

In the previous section, we have supposed that the measurements to be affected to a given trajectory were unambiguously defined in a previous step of *pattern recognition*. In practice, for a complex event including many particles, this preliminary task is far from being easy, and in most cases it cannot be achieved without any ambiguity. The progressive fitting procedure can help to resolve these ambiguities, for example using the probabilities of χ^2. For a better discriminating power, it may also be used within the pattern recognition procedure itself, to perform a progressive collection of points along the trajectory. The basic procedure [5, 6] is as follows:

- build tentative "segments" using points from a few layers, with loose criteria of compatibility; in this step ambiguities are freely accepted.
- apply a forward and/or backward filter to these segments.
- extrapolate to the next and/or previous layer and try to add a measurement found in this layer, and apply a χ^2 criterion to accept or reject this measurement; at this level ambiguities are still accepted (and possibly extended if several measurements are compatible with the extrapolation).
- iterate the procedure. In principle the χ^2 is more and more selective with more and more points included.
- at the end, resolve the remaining ambiguities if any (or keep some of them open for a final analysis).

In any case, the strategy should be adapted to the context on the following points: chosing the best starting region, tuning the criteria at each step, defining tolerance for missing points, using approximations in the filter (for example: ignoring the noise, assuming a small curvature), etc. In some cases, an external

measurement (e.g. from a calorimeter or a muon chamber) may be provide a good starting segment; if the first layers are very precise (as in usual "vertex detectors"), it can be used to define clean segments to be extrapolated forwards, because at this level there are few parasitic tracks produced in the material, and the trajectories are quasi straight lines. Hybrid strategies may also be efficient; there is no general rule on this subject.

7.5 Beyond the gaussian approximation

The measurement errors and the perturbations on the trajectory are never exactly gaussian. Some deviations from "gaussianity" are not worrying, because the convolution of different errors tend to "gaussianize" the combination. For example, let us imagine a series of hodoscopes which just provide an interval (x_1, x_2) for a coordinate x at different position in y along a straight trajectory in xy plane, described by parameters a, b: in the absence of multiple scattering, each measurement gives slice in the a, b plane, and the global information is a polygon which is more or less extended depending on the position of the trajectory, while the gaussian model gives an ellipse with a shape depending only on the coordinates y_k of the hodoscopes; on the contrary, accounting for the multiple scattering results in a smoothing of the distribution of errors. Modern detectors are generally not hodoscopes, and the distribution of errors is often smooth and nearly gaussian. In the case of precise measurements, the non-gaussianity is wiped out by the "noise" along the trajectory.

More serious is the problem of errors with long tails, especially in the energy loss of electrons or positrons, which may be large even through a moderate amount of matter. These tails are propagated throughout the fitting procedure, so that the fitted values do not follow a gaussian distribution, and their variance is underestimated in the gaussian model. We have described above tools to detect abnormal deviations, but we want to go further and try to use explicitly the shape of the error distribution in the case where it is known, or predictable from the parameters of the trajectory. In practice we have to find a reasonable compromise between an *ideal* procedure (complete description and propagation of the errors), which will appear to be extremely heavy with several parameters, and the available computing power; we also want to have an idea of what we can gain with respect to the gaussian procedure, which is very fast.

7.5.1 The ideal procedure

The fitting procedure is still a forward or backward chain of basic operations (measurement, noise, propagation) along the trajectory, but now the "atom" of information is a density function $F(\mathbf{p})$ in the space of parameters, which express the likelihood of the subset of measurements included from the beginning of the chain. The previous considerations on the *independence* of the errors are still valid, so the mathematical transformations of F corresponding to the basic operations are:

- **measurement**: combination of independent informations, that is a **product**:

$$F^{\text{new}}(\mathbf{p}) = F^{\text{old}}(\mathbf{p})\, f^{\text{meas}}(m(\mathbf{p}))$$

  where m is the expression of the local measurement as a function of the local parameters, and f the distribution of the error on m.

- **noise**: addition of independent errors, that is a **convolution**:

$$F^{\text{new}}(\mathbf{p}) = F^{\text{old}}(\mathbf{p}) * g^{\text{noise}}(\mathbf{p})$$

- **propagation**: going from one layer to the next one consists in a transformation of the local parameters, that is a **composition**:

$$F^{\text{new}}(\mathbf{p}) = F^{\text{old}}(\mathcal{P}^{-1}(\mathbf{p}))/J(\mathcal{P}) \quad \rightarrow \quad F^{\text{new}} = F^{\text{old}} \circ \mathcal{P}^{-1}/J(\mathcal{P})$$

  where $\mathcal{P}$ is the transformation from the local parameters in the initial layer to the local parameters in the final one, and $J(\mathcal{P})$ its jacobian determinant.

None of these operations can be performed in a reasonable computing time in a multidimensional space (5 parameters in the standard implementation), without an adequate parametrization of F, $f^{\rm meas}$ and $g^{\rm noise}$.

7.5.2 The Gaussian Sum Filter

One practical solution is to replace all functions involved in the different steps by a *sum of gaussian functions* [8]. The main advantage is that such functions are defined by a small set of values (the mean $\mathbf{p}_0$ and the weight matrix W); both their product and their convolution are gaussian, and the mean value and the weight matrix of the result have simple expressions. We summarize here the algebra of gaussian functions in a N-dimensional space:

$$\begin{aligned}
\text{normalized density} \quad G_{\mathbf{p}_0,W}(\mathbf{p}) &= \sqrt{\frac{\det(W)}{(2\pi)^N}}\,\exp\left(-\frac{W\,[\mathbf{p}-\mathbf{p}_0]}{2}\right)\\
\text{product} \quad G_{\mathbf{p}_1,W_1}\,G_{\mathbf{p}_2,W_2} &= C_{12}\,G_{\bar{\mathbf{p}},W_1+W_2}\\
\text{with} \quad \bar{\mathbf{p}} &= (W_1+W_2)^{-1}(W_1\mathbf{p}_1+W_2\mathbf{p}_2) \quad \text{(weighted mean of } \mathbf{p}_1 \text{ and } \mathbf{p}_2\text{)}\\
\text{and} \quad C_{12} &= \sqrt{\frac{\det(W_1)\,\det(W_2)}{(2\pi)^N\det(W_1+W_2)}}\,\exp\left(-\frac{W_1\,[\mathbf{p}_1-\bar{\mathbf{p}}]+W_2\,[\mathbf{p}_2-\bar{\mathbf{p}}]}{2}\right)\\
\text{convolution} \quad G_{\mathbf{p}_1,W_1} * G_{\mathbf{p}_2,W_2} &= G_{\mathbf{p}_1+\mathbf{p}_2,\left(W_1^{-1}+W_2^{-1}\right)^{-1}}
\end{aligned}$$

In the linear approximation, the propagation may be expressed as in Sect.7.3: when going from $\mathbf{p}_i$ to $\mathbf{p}_f$, with a jacobian matrix $D_{i\to f}=\partial\mathbf{p}_f/\partial\mathbf{p}_i$, we obtain:

$$W_f = W_i\left[D_{i\to f}^{-1}\right] = W_i\,[D_{f\to i}]$$

If we can approximate the measurement and the noise density functions as linear combinations of normalized gaussians, with *positive* coefficients:

$$f^{\rm meas}=\sum_j a_j\,G_j \qquad g^{\rm noise}=\sum_k b_k\,G_k \qquad \text{with} \quad \sum_j a_j=\sum_k b_k=1$$

we obtain at each step of the procedure F as a combination of gaussian terms, which is automatically *positive* in the whole space of parameters. Of course, the main problem is that after n steps including each a sum with m_i coefficients, F is expressed as a sum of $\prod m_i$ terms, so the complexity may be too high if the detector has many layers. This can be partly cured by reducing the number of terms after each step, for example, suppressing the terms with low coefficients, or grouping similar terms into one. The strategy should be tuned for a given detector configuration.

To illustrate the method and the possible gain, we come back to our simplest model with one parameter (Sect.7.2.1), with one difference: the displacement η between two measurements is no longer gaussian. We adopt here an asymmetric superposition of gaussian functions, with mean value 0:

$$g(\eta)=\frac{a_1\,G_{\mu_1,\tau_1}+a_2\,G_{\mu_2,\tau_2}+a_3\,G_{\mu_3,\tau_3}}{a_1+a_2+a_3} \quad \text{with} \quad a_1\mu_1+a_2\mu_2+a_3\mu_3=0$$

The variance of the displacement is then $\tau^2(\eta)=\left(a_1(\mu_1^2+\tau_1^2)+a_2(\mu_2^2+\tau_2^2)+a_3(\mu_3^2+\tau_3^2)\right)/(a_1+a_2+a_3)$ In the following, we take for the triplets (a,μ,τ): (10,-1,0.3), (3,0,3) and (1,10,10), which give $\tau(\eta)=4.122$; the measurements are gaussian with variance 1. We perform a series of trials of 6 measurements with 5 intermediate displacements; we apply to each sample the standard gaussian filter and the gaussian sum filter (keeping all 3^5 terms), to find an estimator of the initial position. The gaussian sum may be used through its mean value and its standard deviation. Alternatively, one can search for its maximum and the deviations giving a decrease of 1/2 for its logarithm; in this example, there is no significant difference between the two methods. Note that the estimated error on the gaussian sum depends on the actual configuration of the displacements, while the standard filter gives always the same value.

In Fig.7.10 we summarize the results on this example: the gaussian sum gives a slightly narrower

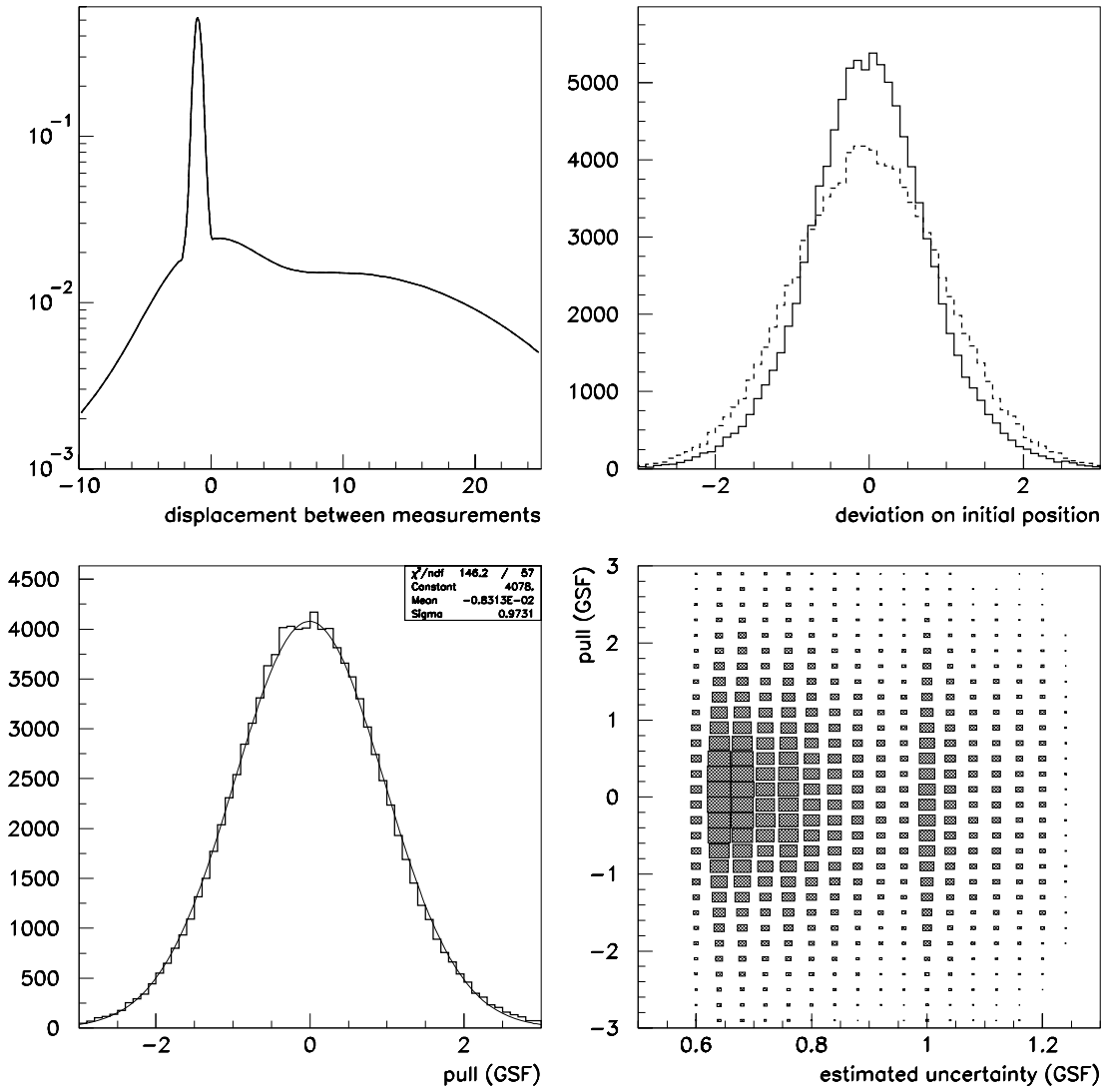


Figure 7.10: Standard Filter vs Gaussian Sum Filter(GSF) in the simple 1D model (with measurement error = 1. Top left: the distribution of the random displacements between the measurements. Top right: error of the GSF estimator (solid) vs the Standard one. Bottom left: the distribution of pulls for the GSF (global). Bottom right: the pulls for GSF vs the estimated uncertainty.

distribution of errors. More important, it provides an estimation of the variance for each realization, which extends over a large range: computing the "pulls" (deviation/error) with this estimation gives the right spread for any value of the variance. In brief, the average error is not greatly improved, but we have a distinction between more or less precise evaluations, with a reliable error for every configuration.

7.5.3 Comments

The main application of a fitting procedure extended beyond the gaussian approximation is the reconstruction of electrons/positrons, accounting for the tail in the distribution of energy loss. We can try to understand intuitively what can be gained. The trajectory is measured over a given segment: if a large energy loss occurs close to the end of this segment, it has no significant effect, whatever the procedure; if it occurs close to the beginning, both the gaussian and the beyond-to-gaussian methods will suffer the same bias on the energy. There may be a significant difference if the large loss occurs in the central region of the segment: the beyond-to-gaussian backward filter includes a tail towards lower curvature (larger energy), so is has more flexibility to modify the curvature when including the points before the large loss, and hence to obtain a better evaluation of the initial energy.

In principle, the formalism may be used outside the measurement range, for example, when including a calorimetric measurement: it can be transformed through an extrapolation to the trajectometer, accounting for the material in between, to give a non-gaussian distribution for the curvature at the beginning of the backward filter (even if was roughly gaussian within the calorimeter). Similarly, the backward extrapolation to the vertex region may be beyond-to-gaussian, including the material crossed before the trajectometer. The non-gaussian features can be introduced in subsequent kinematical reconstructions; in practice, this may be difficult to implement, especially if the trajectometry and the kinematics are handled in independent modules, in the spirit of "hidden boxes" in an Object Oriented framework.

The vertex procedure (Sect.7.6) may be coupled to the track fitting to improve the reconstruction. For example, if an electron/positron is supposed to come from a given vertex, the position of this vertex can be used as a "virtual measurement" constraining the initial part of the trajectory and improving the reconstruction of the energy. But this is not possible if one wants to decide whether this electron/positron comes from the main interaction or from a secondary decay; in any case, such a decision is more ambiguous than for a heavy particle.

7.6 Fitting a vertex

Once the trajectories have fitted, we have for each one a 5-vector of parameters $\mathbf{p}_i$ (intersection with the initial surface) with their weight matrix W_i. Assuming that a given sample of trajectories comes from the same vertex of interaction, we want to reconstruct this vertex (and possibly check this assumption). This way be done at two levels:

- find the best estimator of the 3 coordinates of the vertex (and evaluate errors on them). This may also provide a criterion of quality, e.g. a χ^2 providing a probability for the hypothesis of convergence; if possible, we also want to define a criterion for each individual particle to belong to the vertex. Hereafter we call "simple vertex fit" such a procedure.

- exploit the fact that the trajectories come from this point to improve their reconstruction, that is, add to each trajectory a virtual measurement given by the other ones. This is interesting in view of the kinematical reconstruction of the event. All trajectories are improved, and particularly those measured over a short range, because their parameters (especially the curvature) are poorly defined: an additional point may give a very useful information; but, of course, the criterion to decide whether such a trajectory should be attached to the common vertex is loose. This "full vertex fit" is *a priori* a complex procedure, because we want to fit $3N+3$ parameters: the coordinates of the vertex and 3 quantities to define the initial state of N particles (e.g. p_x, p_y, p_z or better $1/p$ and two angles). We will see how the problem can be simplified using a linearization.

7.6.1 The simple vertex fit

The procedure is conceptually simple. From the initial parameters one can deduce the parameters and their error matrix on any surface: this defines a "tube" of probability around the trajectory. When extrapolating the trajectory backwards from the initial surface to the region of the vertex, the errors on the position increase, so the tube gets broader, but over a short range it may be considered as a cylinder, as illustrated in Fig.7.11 . In other terms, if the position of the vertex is approximately known, each trajectory provides an information on the vertex coordinates that may be summarized in a position with a weight matrix of rank 2 (the position may be arbitray chosen along the axis of the tube): as in Sect.7.3, combining these informations amounts to make their weighted mean. If at least two non parallel tubes are combined, the degeneracy of the position is removed.

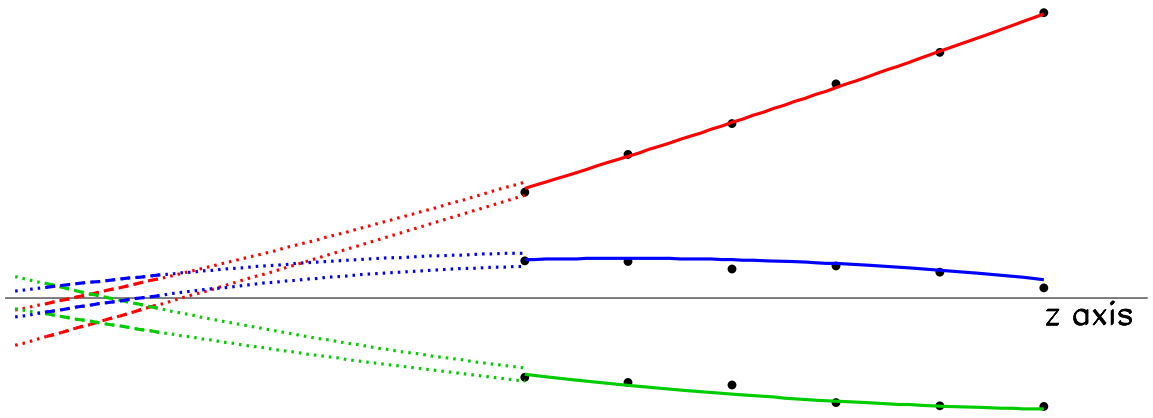


Figure 7.11: Principle of the "simple" vertex fit. Each trajectory fitted to measurements (black points) provides an initial position with its 2×2 error matrix, which is extrapolated backwards to the vertex region (dotted lines). The errors increase with the range of the extrapolation, but in the region of interest (dashed lines) they can be represented by a cylinder, that is, an arbitrary position along a straight line and a constant error matrix on two coordinates (e.g. here x, y at fixed z). Finding the vertex consists in making the *weighted mean* of these tubes, in the sense defined in Sect.7.3.1.

We describe now the mathematical procedure. Let us suppose that the parameters are x, y at fixed z, and three more for the direction and the curvature. First, an approximate position (x_0, y_0, z_0) of the vertex is found from the intersections of the extrapolated trajectories in xz and yz projections. Then, the error matrix C_i of each trajectory is propagated to z_0, using the matrix of derivatives $\mathcal{D}_i = \partial \mathbf{p}_0 / \partial \mathbf{p}_i$; actually, we just need the 2×5 submatrix D_i of the derivatives of x, y (at fixed $z = z_0$) w.r.t. $\mathbf{p}_i$ to compute the 2×2 error matrix $C_{i0} = D_i C_i D_i^T$ and $W_{i0} = C_{i0}^{-1}$. If we approximate locally the trajectory as $x = x_0 + a_x(z - z_0)$, $y = y_0 + a_y(z - z_0)$, we can describe the probability of presence of the vertex at $\mathbf{v} = (x, y, z)$ by saying that the 2-vector $(x - a_x(z - z_0), y - a_y(z - z_0)) = \mathbf{u} - (z - z_0)\mathbf{a}$ has a mean position $\mathbf{u}_0$ with a weight matrix W_{i0}. In the gaussian approximation, this gives a density of probablitity $\exp(-W_{i0}\left[\mathbf{u} - (z - z_0)\mathbf{a} - \mathbf{u}_0\right]/2)$, that is a 3×3 weight matrix for $\mathbf{v}$ (writing W for W_{i0} in the right hand side):

$$W_{iv} = \begin{pmatrix} W_{xx} & W_{xy} & -a_x W_{xx} - a_y W_{xy} \\ & W_{yy} & -a_x W_{xy} - a_y W_{yy} \\ & & a_x^2 W_{xx} + 2a_x a_y W_{xy} + a_y^2 W_{yy} \end{pmatrix}$$

Let $\mathbf{v}_{i0} = (\mathbf{u}_{i0}, z_0)$ be the mean position of the extrapolation of the trajectory i. Its weight matrix W_{iv} has rank 2, but if the extrapolations of trajectories at z_0 are not all parallel, the weighted mean of the extrapolated positions of the trajectories is defined as $(\sum_i W_{iv})^{-1} \sum_i W_{iv} \mathbf{v}_{i0}$: this is an estimator of the vertex position (the best one in the gaussian case), with an error matrix $(\sum_i W_{iv})^{-1}$.

7.6.2 The full vertex fit as a "hierarchical" fit

This procedure ([3, 7, 9]) aims to fit $3N + 3$ parameters ($\mathbf{V} = (x_v, y_v, z_v)$ and $\mathbf{q}_i = (1/p_i, \theta_{vi}, \varphi_{vi})$ for each of the N particles) to a set of N 5-fold measurements, e.g. $\mathbf{p}_k = (x_i, y_i, 1/p_i, \theta_i, \varphi_i)$ at a fixed value of z for particle i, with a weight matrix W_k. The equation of propagation expresses $\mathbf{p}_k$ as a function of V and $\mathbf{q}_k$, and we see that the parameters to be fitted do not play the same role: for any k, $\mathbf{p}_i$ depends on $\mathbf{V}$, but not on $\mathbf{q}_j$ if $j \neq i$. So we can distinguish 3 *global* parameters and N *individual* subsets of 3 parameters, which are related to only one measurement: we call "hierarchical" such a fit.

If we want to perform the fit by minimizing a function F (χ^2 or negative log-likelihood), we can write it as a sum over the particles (measured independently):

$$F(\mathbf{V}, \mathbf{q}_1, \cdots \mathbf{q}_N) = \sum_i f(\mathbf{p}_i(\mathbf{V}, \mathbf{q}_i)) \quad \rightarrow \quad \min F = \min_{\mathbf{V}} \sum_i \min_{\mathbf{p}_i} f(\mathbf{p}_i(\mathbf{V}, \mathbf{q}_i))$$

So, if we want to use a minimizing package, we can use an embedded structure of minimizers, where the function of $3N + 3$ parameters to be minimized is itself computed at each step through N calls to a minimizer of a function of 3 parameters. In this procedure, the correlations between all parameters are taken into account to find the best path towards the minimum.

We can imagine another solution, using an iterative alternate procedure: fit V with fixed $\mathbf{q}_k$, then fit each $\mathbf{q}_k$ with fixed V, and so on. In practice, if the parameters are correlated, the convergence may be very slow[5]. It may be accelerated if F is quasi quadratic around its minimum: in this favourable case, the differences between the parameters at step i and their final values decrease exponentially with i, so after a certain number of steps, one can evaluate a good approximation of the limits, restart the alternate fit, redo the computation of limits, and restart the overall procedure if needed. An advantage of this method is to offer a better control of the convergence, and a way to remove tracks during the iteration, while the first one uses the minimizer package as a "black box" where the internal strategy cannot be modified.

Whatever the strategy, the computing time grows rapidly with N. If the particles produced in the initial interaction may produce secondary vertices (decays or interactions in the material), one has to make trials to choose to best association of tracks to vertices, and the computation may become very heavy. Fortunately, in most detectors, the $\mathbf{p}_k$ depend linearly on the variations of $\mathbf{V}$ and $\mathbf{q}_k$ within a few standard deviations around the central values: with this approximation, minimizing a global χ^2 depending

[5] a similar situation occurs when going to the minimum of $g(x, y)$ by alternate downgoing steps at fixed x and y: if the valley has elliptical contours with oblique axes, it is easy to see that the convergence is slow when the ellipses have a large length/width ratio.

on $3N+3$ parameters will result in a sparse linear system of equations, that can be solved through a number of operations proportional to N, and moreover adding a removing a track to/from a vertex will be simple.

7.6.3 Linearization of the problem

Let $\mathbf{V}_0$ be an approximate position of the vertex, and $\mathbf{q}_{i0}$ approximate parameters of track i at the vertex (e.g. the ones obtained by a backward extrapolation of $\mathbf{p}_i$ to z_0). In the linear approximation we differentiate the propagation from $\mathbf{V}, \mathbf{q}_i$ to $\mathbf{p}_i$:

$$\mathbf{p}_i(\mathbf{V}_0 + \delta\mathbf{V}, \mathbf{q}_{i0} + \delta\mathbf{q}_i) = \mathbf{p}_i(\mathbf{V}_0, \mathbf{q}_{i0}) + D_i\,\delta\mathbf{V} + E_i\,\delta\mathbf{q}_i$$

where D_i and E_i are (5×3) matrices of derivatives[6]. So, if each track was fitted individualy as $(\mathbf{p}_i^f, W_i)$ at the initial point, we can rewrite $\chi^2 = \sum_i W_i \left[\mathbf{p}_i - \mathbf{p}_i^f\right]$ as:

$$\chi^2 = \sum_i W_i \left[\mathbf{p}_i(\mathbf{V}_0, \mathbf{q}_{i0}) - \mathbf{p}_i^f + D_i\,\delta\mathbf{V} + E_i\,\delta\mathbf{q}_i\right] = \sum_i W_i \left[\Delta\mathbf{p}_i - D_i\,\delta\mathbf{V} - E_i\,\delta\mathbf{q}_i\right]$$

where we have introduced the 5-vector of deviations of the individual fits from the predictions of the first approximation at the vertex: $\Delta\mathbf{p}_i = \mathbf{p}_i^f - \mathbf{p}_i(\mathbf{V}_0, \mathbf{q}_{i0})$.

Using the image of "tubes" in the space of parameters , we can say that the fit of the trajectory i defines a tube of rank 5 in the 6D space $(\mathbf{V}, \mathbf{q}_i)$, and we have to combine N such tubes in a $(3N+3)$-D space. The χ^2 is here approximated by a quadratic function of the parameters, so the minimum is given by a linear system of $3N+3$ equations on $\delta\mathbf{V}$ and the $\delta\mathbf{q}_i$. This system may be split in $N+1$ blocks of 3 equations; the first block contains terms for all parameters:

$$\sum_i (D_i^T W_i D_i)\,\delta\mathbf{V} + \sum_i (D_i^T W_i E_i)\,\delta\mathbf{q}_i = \sum_i (D_i^T W_i)\,\Delta\mathbf{p}_i \quad \text{in short} \quad \mathcal{A}\,\delta\mathbf{V} + \sum_i \mathcal{B}_i\,\delta\mathbf{q}_i = \mathcal{U}$$

the N following ones contain each $\delta\mathbf{V}$ and only one of the $\delta\mathbf{q}_i$:

$$(E_i^T W_i D_i)\,\delta\mathbf{V} + (E_i^T W_i E_i)\,\delta\mathbf{q}_i = (E_i^T W_i)\,\Delta\mathbf{p}_i \quad \text{in short} \quad \mathcal{B}_i^T\,\delta\mathbf{V} + \mathcal{C}_i\,\delta\mathbf{q}_i = \mathcal{T}_i$$

The last blocks of equations give expressions of the $\delta\mathbf{q}_i$ as a function of $\delta\mathbf{V}$:

$$\delta\mathbf{q}_i = \mathcal{C}_i^{-1}(\mathcal{T}_i - \mathcal{B}_i^T\,\delta\mathbf{V})$$

which can be injected into the first one to obtain an equation giving $\delta\mathbf{V}$:

$$\mathcal{A}\,\delta\mathbf{V} + \sum_i \mathcal{B}_i\,\mathcal{C}_i^{-1}(\mathcal{T}_i - \mathcal{B}_i^T\,\delta\mathbf{V}) = \mathcal{U} \quad \rightarrow \quad \delta\mathbf{V} = \left(\mathcal{A} - \sum_i \mathcal{B}_i\mathcal{C}_i^{-1}\mathcal{B}_i^T\right)^{-1}\left(\mathcal{U} - \sum_i \mathcal{B}_i\mathcal{C}_i^{-1}\mathcal{T}_i\right)$$

Then each $\delta\mathbf{q}_i$ follows from $\delta\mathbf{V}$.

The left hand side of the linear system written above may be expressed with a sparse matrix $\mathcal{W}$ of 3×3 blocks, where only the first line, the first raw and the diagonal are non-zero, easily solved by a substitution method:

$$\begin{pmatrix} \mathcal{A} & \cdots & \mathcal{B}_i^T & \cdots \\ \vdots & \ddots & & \\ \mathcal{B}_i & & \mathcal{C}_i & \\ \vdots & & & \ddots \end{pmatrix} \begin{pmatrix} \delta\mathbf{V} \\ \vdots \\ \delta\mathbf{q}_i \\ \vdots \end{pmatrix} = \begin{pmatrix} \mathcal{U} \\ \vdots \\ \mathcal{T}_i \\ \vdots \end{pmatrix}$$

[6] We use here the same notation D_i as in Sect.7.6.1, but the definition and the meaning are different. Note that here D_i goes from vertex to the initial point of the trajectory.

The global covariance matrix on the $3N+3$ parameters is the inverse of the global weight matrix $\mathcal{W}$, that is, written by 3×3 blocks:

$$\begin{aligned} cov(\mathbf{V},\mathbf{V}) &= \left(\mathcal{A}-\sum_i \mathcal{B}_i\mathcal{C}_i^{-1}\mathcal{B}_i^T\right)^{-1} \\ cov(\mathbf{V},\mathbf{q}_i) &= -cov(\mathbf{V},\mathbf{V})\,\mathcal{B}_i\mathcal{C}_i^{-1} \\ cov(\mathbf{q}_i,\mathbf{q}_j) &= \delta_{ij}\mathcal{C}_i^{-1}+\mathcal{C}_i^{-1}\mathcal{B}_i^T\,cov(\mathbf{V},\mathbf{V})\,\mathcal{B}_j\mathcal{C}_j^{-1} \end{aligned}$$

This provides an error matrix on the position of the vertex and re-evaluated error matrices on the indivdual particles. In addition, it should be noted that the procedure introduces a correlation between the different particles: this means that, in principle, this correlation should be taken into account when evaluating the uncertainty on physical quantities built with several particles of a vertex, like the total momentum and energy, equivalent masses, etc.

The system has a unique solution for $N>1$, provided the trajectories are not all parallel at the vertex. The total number of operations needed to solve the system grows proportionally to N if N is large. In the gaussian approximation, the minimum of χ^2, as found with the solution of the linear system, follows a law of χ^2 with $5N-(3N+3)=2N-3$ degrees of freedom. The associated probability gives a criterion to decide whether the set of N tracks are actually compatible with the hypothesis of a common origin.

7.6.4 Flexibility and iterative procedures

In many cases, the bundling of tracks in vertices is ambiguous, either because there are short lived particles giving a secondary vertex close to the main one, or because some secondary particles from remote decays (e.g. K^0 or Λ) are not obviously distinguished. In these conditions, it may be useful to add or remove a track from a vertex to perform a new trial. Let us suppose that we have fitted a vertex with N tracks, giving $\chi^2_{\min}$. We take the fitted parameters as new starting values; then the χ^2 including a $(N+1)^{th}$ track may be written as:

$$\chi^2=\chi^2_{\min}+\sum_{i=1}^{N} W_i\left[\Delta D_i\,\delta\mathbf{V}+E_i\,\delta\mathbf{q}_i\right]+W_{N+1}\left[\Delta\mathbf{p}_{N+1}-D_{N+1}\,\delta\mathbf{V}-E_{N+1}\,\delta\mathbf{q}_{N+1}\right]$$

Using $\mathcal{A}$, $\mathcal{B}_i$, $\mathcal{C}_i$, $\mathcal{T}_i$, $\mathcal{U}$ as defined above, and introducing $\mathcal{A}_{N+1}=D_{N+1}^T W_{N+1} D_{N+1}$ and $\mathcal{U}_{N+1}=D_{N+1}^T W_{N+1}\,\Delta\mathbf{p}_{N+1}$ the minimization gives:

$$\begin{aligned} (\mathcal{A}+\mathcal{A}_{N+1})\,\delta\mathbf{V}+\sum_{i=1}^{N+1}\mathcal{B}_i\,\delta\mathbf{q}_i &= \mathcal{U}_{N+1} \\ \text{for } i=1 \text{ to } N \qquad \mathcal{B}_i^T\,\delta\mathbf{V}+\mathcal{C}_i\,\delta\mathbf{q}_i &= 0 \quad\rightarrow\quad \delta\mathbf{q}_i=-\mathcal{C}_i^{-1}\mathcal{B}_i^T\,\delta\mathbf{V} \\ \mathcal{B}_{N+1}^T\,\delta\mathbf{V}+\mathcal{C}_{N+1}\,\delta\mathbf{q}_{N+1} &= \mathcal{T}_{N+1} \quad\rightarrow\quad \delta\mathbf{q}_{N+1}=\mathcal{C}_{N+1}^{-1}(\mathcal{T}_{N+1}-\mathcal{B}_{N+1}^T\,\delta\mathbf{V}) \end{aligned}$$

Injecting in the first equation the expressions of $\delta\mathbf{q}_i$ found in the second and in the third one, we obtain an equation which gives $\delta\mathbf{V}$:

$$\left(\mathcal{A}+\mathcal{A}_{N+1}-\sum_{i=1}^{N+1}\mathcal{B}_i\mathcal{C}_i^{-1}\mathcal{B}_i^T\right)\delta\mathbf{V}=\mathcal{U}_{N+1}-\mathcal{B}_{N+1}\,\mathcal{C}_{N+1}^{-1}\,\mathcal{T}_{N+1}$$

hence the $\delta\mathbf{q}_i$. The matrices $\left(\mathcal{A}-\sum_{i=1}^{N+1}\mathcal{B}_i\mathcal{C}_i^{-1}\mathcal{B}_i^T\right)$ and $\mathcal{C}_i^{-1}\mathcal{B}_i^T$ were already computed in the fit with N tracks, so the amount of computation needed is much less than doing a fit with $N+1$ tracks *ab initio*.

The same algorithm can be applied to remove a track from a vertex fit: it is equivalent to add this track with a negative weight matrix $-W_i$. Note also that all operations (multiplications, inversions) involve always 3×5 and 3×3 matrices and may be coded explicitly in a very efficient way, avoiding any use of indexed arrays.

The tools described above can be inserted in an iterative vertex building. In most detectors the trajectories around the vertex may be described by smooth functions, so the linear approximation is quite adequate, especially when using the "perigee" parameters (no iteration is needed for a given set of tracks). With quality criteria based for example on the probability of χ^2, a strategy may be defined to determine the best repartition of the tracks in vertices; this strategy may be driven by physical considerations, for example finding the decay of a heavy flavour particle, starting from a "seed" (a large p_t lepton, a combination identified by equivalent masses, etc).

7.6.5 Fitting a vertex with constraints

Beam profile

The simplest case of constraint for the main vertex is the beam profile, which may be considered as a particular trajectory entering the vertex, except that its parameters should not be re-evaluated in the procedure. If the z axis is chosen along the beam, the lateral profile is usually summarized by the lateral standard deviations σ_x and σ_y, and the constraint may be expressed by just adding a term $(x_v/\sigma_x)^2 + y_v/\sigma_y)^2$ to the χ^2. In the "simple vertex" procedure (Sect.7.6.1), we just need to add a trajectory with $x_0 = y_0 = a_x = a_y = 0$ and a diagonal weight matrix $(1/\sigma_x^2, 1/\sigma_y^2)$. In the "full vertex" fit within the linear approximation (Sect.7.6.3), the additional term results in adding the diagonal matrix $W_b = (1/\sigma_x^2, 1/\sigma_y^2, 0)$ to $\mathcal{A}$ and $-W_b \mathbf{V}_0$ to $\mathcal{U}$.

This formalism may be applied to both fixed target experiments and colliders, but it improves the results only if σ_x and σ_y are comparable to, or smaller than the errors on extrapolated tracks: in practice, this is true in colliders, where the beam constraint is very useful.

Kinematical and geometrical constraints

A typical example is the reconstruction of the so called V^0's: remote decay of a neutral object (γ, K^0, Λ, etc) in two charged particles. The constraint consists in the value of the equivalent mass of the pair with given mass assumptions for the charged particles i and j. The fit may be performed by removing one of the free parameters and replacing it by a function of the other ones; in the case of $\gamma \rightarrow e^+e^-$, we can force $\vec{p}_i$ and $\vec{p}_j$ to be parallel by making a fit with 7 parameters instead of 9: x_v, y_v, z_v, a common value for θ and φ, and the curvatures c_i and c_j. More generally, the constraint may be expressed as $F(\mathbf{p}_i, \mathbf{p}_j) = 0$ and handled through the generic method of Lagrange multipliers (see below).

As a result of the fit, we obtain estimators for the trajectory of the neutral particle (a straight line), which can be inserted in a primary or a nearby secondary vertex, through one of the procedures described above: the introduction of 4-vectors instead of 5-vectors in the formalism is straightforward. Conversely, we can introduce an additional constraint in the remote vertex fit, if the neutral particle comes from a previously fitted vertex with a given error matrix.

General procedure with Lagrange multipliers in the linear approximation

The constraint may be applied as a further step after the "standard" fit (with the pure constraint of convergence). Let us call now $\mathbf{p}^f$ the global vector of values fitted without the constraint $F(\mathbf{p}) = 0$. We can suppose that the constraint is nearly satisfied by $\mathbf{p}^f$, that is, F can be linearly expanded in the neighbourhood of $\mathbf{p}^f$:

$$F(\mathbf{p}) = F(\mathbf{p}^f) + \nabla F.(\mathbf{p} - \mathbf{p}^f)$$

where ∇F is the gradient of F at $\mathbf{p}^f$. On the other hand, the χ^2 may be approximated by a quadratic expansion around $\mathbf{p}^f$:

$$\chi^2 = \chi^2_{\min} + \mathcal{W}\left[\mathbf{p} - \mathbf{p}^f\right]$$

The Lagrange multiplier method consists in finding the minimum of:

$$G(\mathbf{p}) = \frac{1}{2}\left(\chi^2_{\min} + \mathcal{W}\left[\mathbf{p} - \mathbf{p}^f\right]\right) + \lambda\left(F(\mathbf{p}^f) + \nabla F.(\mathbf{p} - \mathbf{p}^f)\right)$$

when varying both the components of $\mathbf{p}$ and λ. This is a quadratic function, so the solution can be obtained easily in two steps:

- canceling the gradient of G w.r.t. $\mathbf{p}$, which provides $\mathbf{p}$ as a linear function of λ:

$$\nabla G(\mathbf{p}) = \mathcal{W}\left[\mathbf{p} - \mathbf{p}^f\right] + \lambda \nabla F = 0 \qquad \rightarrow \qquad \mathbf{p} - \mathbf{p}^f = -\lambda \mathcal{W}^{-1}.\nabla F$$

- canceling the derivative w.r.t. λ and introducing the previous expression to solve a linear equation in λ:

$$\frac{\partial G}{\partial \lambda} = F(\mathbf{p}^f) + \nabla F.(\mathbf{p} - \mathbf{p}^f) = F(\mathbf{p}^f) + \lambda \mathcal{W}^{-1}\left[\nabla F\right] \qquad \rightarrow \qquad \lambda = \frac{F(\mathbf{p}^f)}{\mathcal{W}^{-1}\left[\nabla F\right]}$$

  This value of λ gives $\mathbf{p}$ through the first equation.

This procedure is easily extended to the case of several simultaneous constraints $F_k(\mathbf{p})$, $k = 1 \cdots N_c$:

$$G(\mathbf{p}) = \frac{1}{2}\left(\chi^2_{\min} + \mathcal{W}\left[\mathbf{p} - \mathbf{p}^f\right]\right) + \sum_k \lambda_k \left(F(\mathbf{p}^f) + \nabla F_k.(\mathbf{p} - \mathbf{p}^f)\right)$$

We obtain $\mathbf{p} - \mathbf{p}^f$ as a linear function of the λ_k and a linear system of equations on the λ_k, where the expressions of $\mathbf{p} - \mathbf{p}^f$ may be injected to eliminate them. The values of the λ_k obtained from this linear system provide the wanted solution for $\mathbf{p}$.

7.7 Appendix: derivative matrix for propagation in the helix model

With the notations defined in Sect.7.3.5, the trajectory is defined through a running parameter φ:

$$\begin{aligned} x &= x_0 + \mathcal{R}\,(\sin\varphi - \sin\varphi_0) = r_0\,\cos\Phi_0 + \mathcal{R}\,(\sin\varphi - \sin\varphi_0) \\ y &= y_0 - \mathcal{R}\,(\cos\varphi - \cos\varphi_0) = r_0\,\sin\Phi_0 + \mathcal{R}\,(\cos\varphi - \cos\varphi_0) \\ z &= z_0 + \mathcal{R}\,\cot\theta\,(\varphi - \varphi_0) \end{aligned}$$

We define the signed curvature $c = 1/\mathcal{R}$. To simplify expressions we use $t = \cot\theta$ as parameter instead of θ. We want to obtain analytical expression for the derivatives of the non-constant parameters (i.e. other than t, c) on a surface, as functions of the parameters on another one.

7.7.1 Propagation between planes

Planes perpendicular to the magnetic field

The initial parameters are x, y, φ, t, c at fixed z_0. To obtain the parameters at $z = z_1$, we can write, with $\Delta z = z_1 - z_0$ and $\ell = \mathcal{R}\,\Delta\varphi = \Delta z/t$ (length of the arc in xy projection):

$$\varphi_1 = \varphi_0 + \frac{c\Delta z}{t}$$

hence the derivatives:

$$\frac{\partial\varphi_1}{\partial\varphi_0} = 1 \quad ; \quad \frac{\partial\varphi_1}{\partial t} = -\frac{c\Delta z}{t^2} = -\frac{c\ell}{t} \quad ; \quad \frac{\partial\varphi_1}{\partial c} = \frac{\Delta z}{t} = \ell$$

The expressions for x_1, y_1 and their derivatives follow immediately from φ_1, using the notation $\Delta A = A_1 - A_0$ for any quantity A depending on the local parameters:

$$\begin{aligned} \frac{\partial x_1}{\partial x_0} &= \frac{\partial y_1}{\partial y_0} = 1 \quad ; \quad \frac{\partial x_1}{\partial y_0} = \frac{\partial x_1}{\partial t} = \frac{\partial y_1}{\partial x_0} = \frac{\partial y_1}{\partial t} = 0 \\ \frac{\partial x_1}{\partial\varphi_0} &= \mathcal{R}\left(-\cos\varphi_0 + \cos\varphi_1\,\frac{\partial\varphi_1}{\partial\varphi_0}\right) = \frac{\Delta(\cos\varphi)}{c} \quad ; \quad \frac{\partial x_1}{\partial c} = -\frac{\Delta(\sin\varphi)}{c^2} \\ \frac{\partial y_1}{\partial\varphi_0} &= \mathcal{R}\left(-\sin\varphi_0 + \sin\varphi_1\,\frac{\partial\varphi_1}{\partial\varphi_0}\right) = \frac{\Delta(\sin\varphi)}{c} \quad ; \quad \frac{\partial y_1}{\partial c} = \frac{\Delta(\cos\varphi)}{c^2} \end{aligned}$$

In the approximation of weak curvature ($|\Delta\varphi| \ll 1$) we obtain "quasi straight line" approximations for the last two:

$$\frac{\partial x_1}{\partial\varphi_0} \simeq -\ell\sin\varphi_0 \quad ; \quad \frac{\partial x_1}{\partial c} \simeq -\frac{\ell\cos\varphi_0}{c} \quad ; \quad \frac{\partial y_1}{\partial\varphi_0} \simeq \ell\cos\varphi_0 \quad ; \quad \frac{\partial x_1}{\partial c} \simeq -\frac{\ell\sin\varphi_0}{c}$$

Planes parallel to the magnetic field

The parameters are now y, z, φ, t, c at fixed x; we want to go from x_0 to x_1. As previously, we first determine φ_1 and its derivatives, using similar notations (assuming that the right solution φ_1 of the trigonometric equation is chosen; usually it is the closest one to φ_0); some of the computations are the same as in Sect. 7.3.4.

$$\sin\varphi_1 = \sin\varphi_0 + c\,\Delta x \quad \rightarrow \quad \frac{\partial\varphi_1}{\partial\varphi_0} = \frac{\cos\varphi_0}{\cos\varphi_1} \quad , \quad \frac{\partial\varphi_1}{\partial c} = \frac{\cos\varphi_0}{\cos\varphi_1}\,\Delta x$$

(the other derivatives vanish)

Hence the derivatives of the other parameters which are not 1 or 0:

$$\begin{aligned}
\frac{\partial y_1}{\partial \varphi_0} &= \frac{\sin(\Delta\varphi)}{c\cos\varphi_1} \simeq \frac{\ell}{c\cos\varphi_0} \quad ; \quad \frac{\partial y_1}{\partial c} = \frac{1-\cos(\Delta\varphi)}{c^2\cos\varphi_1} \simeq \frac{\ell^2}{2\cos\varphi_1} \\
\frac{\partial z_1}{\partial \varphi_0} &= \frac{t\,\Delta(\cos\varphi)}{c\cos\varphi_1} \simeq -t\tan\varphi_0\,\ell \quad ; \quad \frac{\partial z_1}{\partial t} = \frac{\Delta\varphi}{c} = \ell \quad ; \quad \frac{\partial z_1}{\partial c} = -\frac{t\,\Delta\varphi}{c^2} \simeq -\frac{t\,\ell}{c}
\end{aligned}$$

7.7.2 Propagation between cylinders

In the following, we will use as trajectory parameters at fixed r: Φ, ξ and $\mathcal{R}$ (or $c = 1/\mathcal{R}$) in xy projection, and $z, t = \cot\theta$ to complete the description in 3D space. The coordinates of the center C of the projection onto the xy plane may be expressed from any point (r, Φ, φ) on the trajectory as:

$$x_c = r\cos\Phi - \mathcal{R}\sin\varphi \tag{7.1}$$
$$y_c = r\sin\Phi + \mathcal{R}\cos\varphi \tag{7.2}$$

so the polar coordinates of C are r_c, Φ_c such that:

$$r_c^2 = r^2 + \mathcal{R}^2 - 2\,\mathcal{R}\,r\sin\xi \qquad \Phi_c = \mathrm{atan2}(y_c, x_c)$$

where we define $\xi = \varphi - \Phi$ (deviation from the radial direction).

As intermediate parameters, we use also r_c defined above and the z coordinate of the perigee $z_p = z - \mathcal{R}\,t\,(\varphi - \varphi_p)$. Note that with our convention on the geometrical sign S of the curvature (the sign of $\mathcal{R}$), we have $\Phi_c = \varphi_p + S\pi/2$. To simplify some expressions we introduce for any point on the trajectory $\psi = \varphi - \varphi_p$ (rotation fom the perigee to this point). Introducing $\rho_c = Sr_c$, we can rewrite (7.1) and (7.2) as:

$$-\rho_c\sin\varphi_p = r\cos\Phi - \mathcal{R}\sin\varphi \tag{7.3}$$
$$\rho_c\cos\varphi_p = r\cos\Phi + \mathcal{R}\cos\varphi \tag{7.4}$$

Then the combinations $\cos\varphi(7.3) + \sin\varphi(7.4)$ and $\cos\varphi(7.4 - \sin\varphi(7.3)$ give:

$$\cos\psi = \frac{\mathcal{R} - r\sin\xi}{\rho_c} \quad ; \quad \sin\psi = \frac{r\cos\xi}{\rho_c}$$

The notations used in this Section are illustrated in Fig.7.12.

We want to obtain the derivatives of the parameters at $r = r_1$ with respect to the parameters at $r = r_0$: to do so we will compute the derivatives of the intermediate parameters (r_c, Φ_c, z_p) with respect to the initial ones and to the final ones, and use the inversion and the multiplication of jacobian matrices. For convenience, we compute derivatives w.r.t. $\mathcal{R}$ instead of c in the intermediate steps. First we consider the transformation from $(\Phi_0, \xi_0, \mathcal{R}, z_0, t)$ to $(\rho_c, \Phi_c, \mathcal{R}, z_p, t)$. From the expression of $\rho_c^2 = r_c^2 = r_0^2 + \mathcal{R}^2 - 2\,\mathcal{R}\,r_0\sin\xi_0$, we have immediately:

$$\begin{aligned}
\frac{\partial \rho_c}{\partial \Phi_0} &= \frac{\partial \rho_c}{\partial z_0} = \frac{\partial \rho_c}{\partial t} = 0 \\
\frac{\partial \rho_c}{\partial \xi_0} &= -\frac{\mathcal{R}\,r_0\cos\xi_0}{\rho_c} = -\mathcal{R}\sin\psi_0 \quad ; \quad \frac{\partial \rho_c}{\partial \mathcal{R}} = -\frac{r_0\sin\xi_0}{\rho_c} = \cos\psi_0 - \frac{\mathcal{R}}{\rho_c}
\end{aligned}$$

For the derivatives of Φ_c, we use $\mathrm{d}(\mathrm{atan2}(y_c, x_c)) = (x_c\mathrm{d}y_c - y_c\mathrm{d}x_c)/(x_c^2 + y_c^2)$, hence for any parameter α: $\partial\Phi_c/\partial\alpha = (x_c\,\partial y_c/\partial\alpha - y_c\,\partial x_c/\partial\alpha)/\rho_c^2$. This gives:

$$\begin{aligned}
\frac{\partial \Phi_c}{\partial \Phi_0} &= 1 \quad \text{(obvious by rotational invariance)} \quad ; \quad \frac{\partial \Phi_c}{\partial z_0} = \frac{\partial \Phi_c}{\partial t} = 0 \\
\frac{\partial \Phi_c}{\partial \xi_0} &= \frac{\mathcal{R}(\mathcal{R} - r_0\sin\xi_0)}{\rho_c^2} = \frac{\mathcal{R}\cos\psi_0}{\rho_c} \quad ; \quad \frac{\partial \Phi_c}{\partial \mathcal{R}} = \frac{r_0\cos\xi_0}{\rho_c^2} = S\,\frac{\sin\psi_0}{\rho_c}
\end{aligned}$$

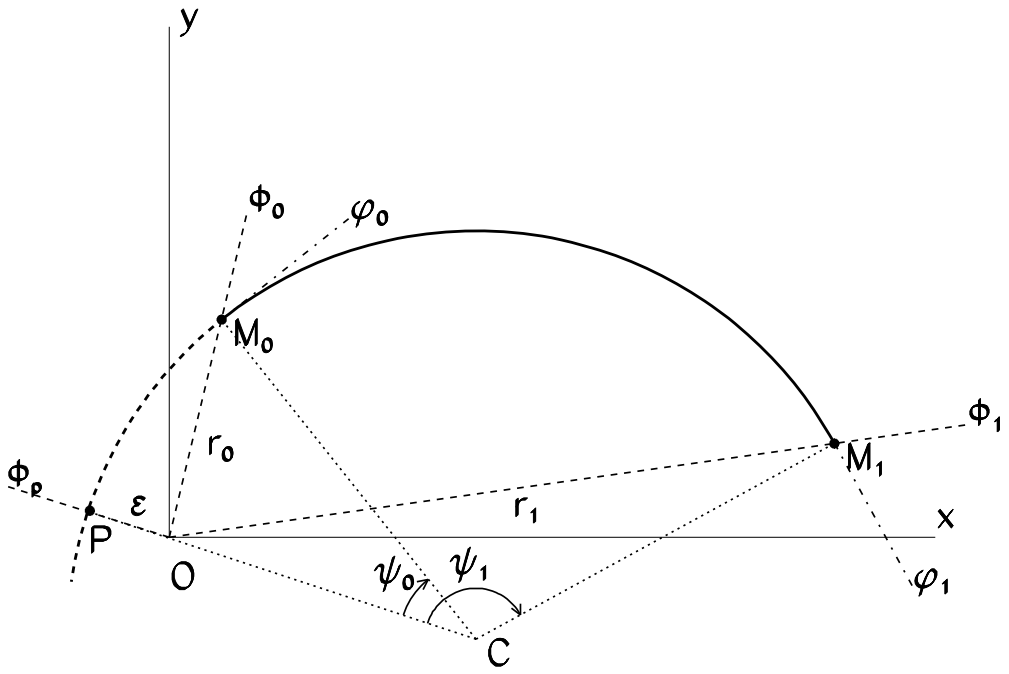


Figure 7.12: Parameters used to express the propagation of cylindrical parameters at fixed r.

The derivatives of z_p are obtained from $z_p = z_0 - \mathcal{R}\, t\, \psi_0 = z_0 + \mathcal{R}\, t\, (\Phi_c - \Phi - \xi - S\pi/2)$; to shorten one expression, we also introduce $\Psi = \Phi - \Phi_c$, with $\cos\Psi = (r - \mathcal{R}\sin\xi)/r_c$:

$$\begin{aligned}
\frac{\partial z_p}{\partial \Phi_0} &= 0 \quad ; \quad \frac{\partial z_p}{\partial z_0} = 1 \\
\frac{\partial z_p}{\partial \xi_0} &= \mathcal{R}\, t \left(\frac{\partial \Phi_c}{\partial \xi_0} - 1 \right) = \frac{\mathcal{R}\, r_0\, t\, (\mathcal{R}\sin\xi_0 - r_0)}{\rho_c^2} = -\frac{\mathcal{R}\, r_0\, t \cos\Psi_0}{r_c} \\
\frac{\partial z_p}{\partial t} &= -\mathcal{R}\, \psi_0 \\
\frac{\partial z_p}{\partial \mathcal{R}} &= t \left(\psi_0 + \mathcal{R} \frac{\partial \Phi_c}{\partial \mathcal{R}} \right) = t \left(\psi_0 + \frac{R \sin\psi_0}{\rho_c} \right)
\end{aligned}$$

The derivatives form the jacobian matrix

$$D_{0\to c} = \frac{\partial(\rho_c, \Phi_c, z_p, t, \mathcal{R})}{\partial(\xi_0, \Phi_0, z_0, t, \mathcal{R})} = \begin{pmatrix} -\mathcal{R}\sin\psi_0 & 0 & 0 & 0 & \cos\psi_0 - \mathcal{R}/\rho_c \\ \mathcal{R}\cos\psi_0/\rho_c & 1 & 0 & 0 & \sin\psi_0/\rho_c \\ -\mathcal{R} r_0 t \cos\Psi_0/r_c & 0 & 1 & -\mathcal{R}\psi_0 & t(\psi_0 + \mathcal{R}\sin\psi_0/\rho_c) \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The matrix $D_{1\to c} = \partial(\rho_c, \Phi_c, z_p, t, \mathcal{R})/\partial(\xi_1, \Phi_1, z_1, t, \mathcal{R})$ has exactly the same expression, replacing the index 0 by 1. To obtain matrix $D_{0\to 1}$ of the derivatives of $(\xi_1, \Phi_1, z_1, t, \mathcal{R})$ w.r.t. $(\xi_0, \Phi_0, z_0, t, \mathcal{R})$ we just need to invert $D_{1\to c}$ and then:

$$D_{0\to 1} = D_{c\to 1}\, D_{0\to c} = (D_{1\to c})^{-1}\, D_{0\to c}$$

Note that the sparse structure of $D_{1\to c}$ is conserved by inversion; for any values of (a, b, c, d, e, f, g) we

have:

$$\begin{pmatrix} a & 0 & 0 & 0 & e \\ b & 1 & 0 & 0 & f \\ c & 0 & 1 & d & g \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1/a & 0 & 0 & 0 & -e/a \\ -b/a & 1 & 0 & 0 & be/a - f \\ -c/a & 0 & 1 & -d & ce/a - g \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

This expression makes the algebraic computations easy:

$$\begin{pmatrix} a_1 & 0 & 0 & 0 & e_1 \\ b_1 & 1 & 0 & 0 & f_1 \\ c_1 & 0 & 1 & d_1 & g_1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} a_0 & 0 & 0 & 0 & e_0 \\ b_0 & 1 & 0 & 0 & f_0 \\ c_0 & 0 & 1 & d_0 & g_0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{a_0}{a_1} & 0 & 0 & 0 & \frac{e_0-e_1}{a_1} \\ b_0 - b_1\frac{a_0}{a_1} & 1 & 0 & 0 & f_0 - f_1 + (e_1-e_0)\frac{b_1}{a_1} \\ c_0 - c_1\frac{a_0}{a_1} & 0 & 1 & d_0 - d_1 & g_0 - g_1 + (e_1-e_0)\frac{c_1}{a_1} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

We give the derivatives which are not 1 or 0, using $\Delta\psi = \Delta\varphi$, with some useful approximations as function of $\ell = \mathcal{R}\Delta\varphi$ (length of the arc in xy projection) in the case of low curvature (more precisely: $\Delta\varphi \ll 1$):

$$\begin{aligned}
\frac{\partial\Phi_1}{\partial\xi_0} &= -\frac{R\sin(\Delta\varphi)}{r_c\sin\psi_1} = \frac{\mathcal{R}\sin(\Delta\varphi)}{r_1\cos\xi_1} \simeq \frac{\ell}{r_1\cos\xi_1} \\
\frac{\partial\Phi_1}{\partial\mathcal{R}} &= \frac{\cos(\Delta\varphi)-1}{\rho_c\sin\psi_1} = \frac{\cos(\Delta\varphi)-1}{r_1\cos\xi_1} \rightarrow \frac{\partial\Phi_1}{\partial c} = \frac{R^2(1-\cos(\Delta\varphi))}{r_1\cos\xi_1} \simeq \frac{\ell^2}{2r_1\cos\xi_1} \\
\frac{\partial\xi_1}{\partial\xi_0} &= \frac{\sin\psi_0}{\sin\psi_1} \simeq 1 \\
\frac{\partial\xi_1}{\partial\mathcal{R}} &= \frac{\Delta(\cos\psi)}{\mathcal{R}\sin\psi_1} \rightarrow \frac{\partial\xi_1}{\partial c} = -\frac{\mathcal{R}\Delta(\cos\psi)}{\sin\psi_1} \simeq \ell \\
\frac{\partial z_1}{\partial\xi_0} &= \frac{\mathcal{R}\,t}{r_c}\left(\frac{\sin\psi_0}{\sin\psi_1}\,r_1\cos\Psi_1 - r_0\cos\Psi_0\right) \\
\frac{\partial z_1}{\partial t} &= \mathcal{R}\,\Delta\psi = \ell \\
\frac{\partial z_1}{\partial c} &= \mathcal{R}^2 t\left(-\Delta\varphi + \frac{\mathcal{R}\,\Delta(\sin\psi)}{\rho_c} + \frac{r_1}{r_c}\frac{\Delta(\cos\psi)\cos\Psi_1}{\sin\psi_1}\right)
\end{aligned}$$

Some approximations may be applied to the first and last derivative of z_1 if the impact parameter $|R - r_c|$ is small.

References

[1] R.E. Kalman, "A New Approach to Linear Filtering and Prediction Problems", Transactions of the ASME - Journal of Basic Engineering **82** (1960) 35
R.E. Kalman, R. S. Bucy, "New Results in Linear Filtering and Prediction Theory", Transactions of the ASME - Journal of Basic Engineering **83** (1961) 95

[2] P. Billoir, "Track fitting with multiple scattering: a new method", Nucl. Inst. Meth., **225** (1984) 352

[3] P. Billoir, R. Früwirth, M. Regler, "Track element merging strategy and vertex fitting in complex modular storage ring detectors", Nucl. Inst. Meth., **241** (1985) 115

[4] R. Früwirth, "Application of Kalman filtering to track and vertex fitting" Nucl. Inst. Meth., **262** (1987) 444

[5] P. Billoir, "Progressive track recognition with a Kalman-like fitting procedure", Comp. Phys. Comm. **57** (1989) 390

[6] P. Billoir, S.Qian, "Simultaneous pattern recognition and track fitting by the Kalman Filtering method", Nucl. Inst. Meth. **A294** (1990) 219

[7] P. Billoir, S.Qian, "Fast vertex fitting with a local parametrization of tracks", Nucl. Inst. Meth. **A311** (1990) 219

[8] R. Früwirth, "Track fitting with non-gaussian noise", Comp. Phys. Comm. **110** (1997) 1
R. Früwirth, S. Früwirth-Schnatter, "On the treatment of energy loss in track fitting", Comp. Phys. Comm. **110** (1998) 80

[9] Tools for hierarchical fits in ROOT framework: `http://gentitfx.fr/Hierarchal`

Unfolding in particle physics: a window on solving inverse problems

Francesco Spanò
Royal Holloway, University of London
Egham, Surrey, TW20 0EX, United Kingdom

Abstract

Unfolding is the ensemble of techniques aimed at resolving inverse, ill-posed problems. A pedagogical introduction to the origin and main problems related to unfolding is presented and used as the the stepping stone towards the illustration of some of the most common techniques that are currently used in particle physics experiments.

8.1 Introduction

The problem of recovering the "true", untarnished distribution of the values for a given variable from "smeared", biased and inefficient observations is common to a variety of fields.

Figure 8.1 shows an example from medical imaging [1]. Positron Emission Topography (PET) aims at visualizing the blood flow and metabolic activity in an organ by introducing a positron-emitting radioactive material (tracer) and by detecting X-ray photons from electron-positron annihilation ($e^+e^- \rightarrow \gamma\gamma$) when positrons emitted by the tracer annihilate with electrons from the surrounding organic matter. The reconstruction the photon emission spatial density from the detected counts provides the organ's image and it requires inverting the process outlined in Figure 8.1 .

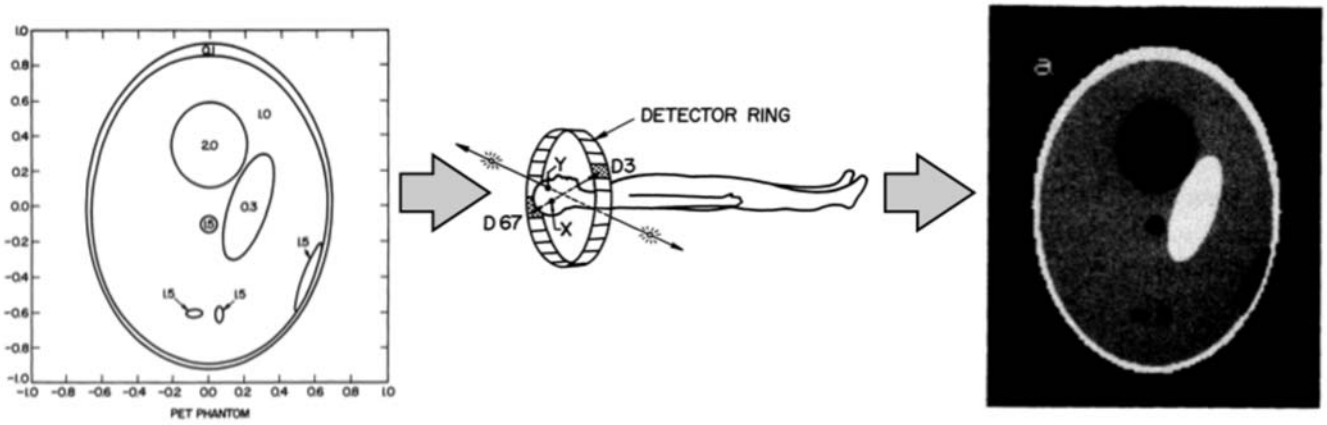


Figure 8.1: Scheme for the evolution of the medical imaging process using figures from Ref. [1]. The simulated photon pair emission density representing the brain (left, Figure 2) is passed to a simulation of the Positron Emission Tomography (center, Figure 1a) that produces the "observed" count distribution from the photon detector (right, Figure 3a). The names of the figures are as they appear in the reference.

Images are often blurred by detector effects and corrupted by the presence of additional random noise [2]. Inverting the process shown in figure 8.2 is necessary to recover the details of initial image.

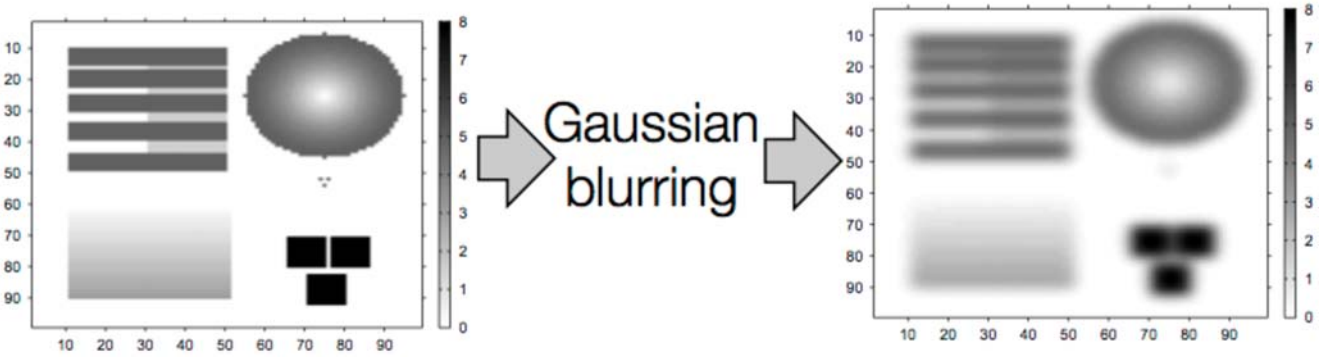


Figure 8.2: Scheme for the evolution of blurring and degradation of a two dimensional image using figures from Ref. [2]. The "true" simulated two-dimensional image (left, Figure 4a) is degraded by convoluting it with a Gaussian "spread" function with the addition of random Gaussian noise (see Section 4 in Ref. [2]) to produce the "observed" image (right, Figure 5A). The names of the figures are as they appear in the reference.

A variety of particle physics measurements share the same "imaging" goal. An illustrative example is the invariant mass of the pair of top-antitop quarks produced in proton-proton (pp) collisions at a center-of-mass energy ($\sqrt{s}$) of 7 TeV at the Large Hadron Collider (LHC) [3] ($pp \rightarrow t\bar{t} + X$) . This is reconstructed by measuring a complex final state involving jets of hadrons and leptons with a multi-purpose detector (in this example the ATLAS detector [4]). The inversion of the measuring process, shown in the cartoon of figure 8.3, is required to correct for detector (and sometimes acceptance) effects and recover the distribution of the underlying physical property. In particle physics *unfolding* is the ensemble

of statistical techniques used to solve what is defined as the *inverse problem*: infer an unknown distribution $f(y)$ for a variable y from the measured distribution $g(s)$ by using knowledge and/or assumptions on the probability distribution that links the observation to the "true" value. Other terms are used that give somewhat more emphasis to recovering one specific feature of the degraded information so the techniques are also named *unsmearing* or *deconvolving*. The proper mathematical formulation of the inverse problem is the crucial step to understand the challenges involved in the unfolding procedure.

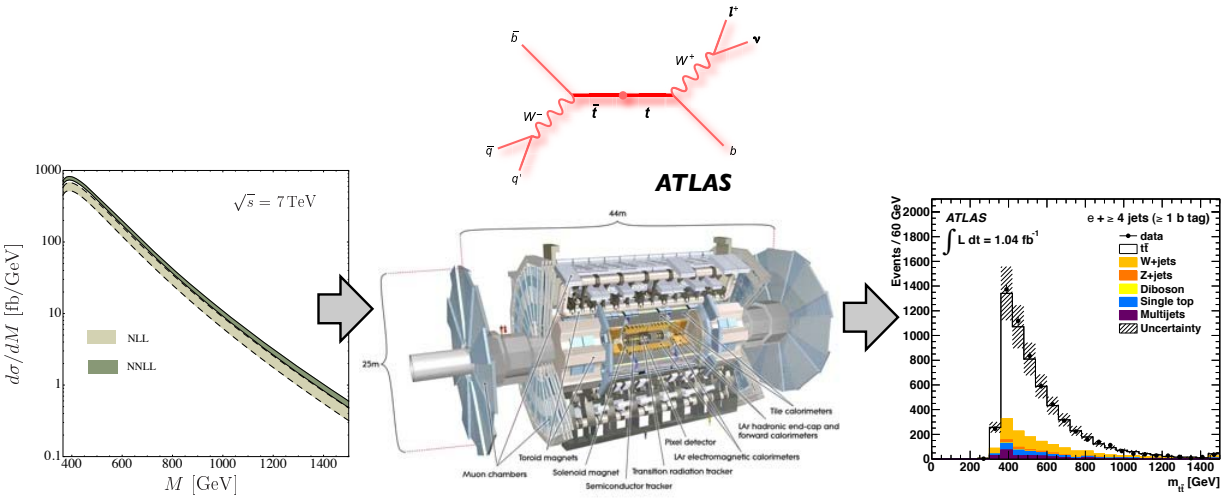


Figure 8.3: Scheme of evolution of the measurement of the invariant mass of the top-antitop quark system. The predicted mass distribution (left, Figure (10, right) in Ref. [5] (shown with the inclusion of theoretical uncertainties) for events featuring top-antitop quarks produced in $\sqrt{s} = 7$ TeV pp collisions at the LHC is reconstructed (right, Figure 1(a) from Ref. [6]) after the top quark decay products are measured by the ATLAS detector (middle, image from Ref. [7]). A diagram from Ref. [8] shows the final state partons from the top quark decay at leading order. The names of the figures are those by which they appear in the references.

8.2 Unfolding foundations

The mathematical foundations of unfolding are intimately related to the description of the inverse problem [10] provided by the Fredholm integral equation of the first type

$$g(\mathbf{s}) = \int_{\Omega} K(\mathbf{s},\mathbf{y}) f(\mathbf{y}) d\mathbf{y} \tag{8.1}$$

where the true $f(\mathbf{y})$ distribution of the variable $\mathbf{y} = (y_1,..,y_J)$ is related to the measured or observed distribution $g(\mathbf{s})$ of the variable $\mathbf{s} = (s_1,..,s_L)$ by the convolution with the *kernel* function $K(\mathbf{s},\mathbf{y})$ [11]. In general the variables $\mathbf{y}$ and $\mathbf{s}$ belong to multidimensional spaces with different dimensions so the two integers J and L are different, in principle. The "volume" Ω represents the support of $f(\mathbf{y})$ i.e. the subspace of the multidimensional space where $\mathbf{y}$ is defined. The distribution $f(\mathbf{y})$ is transformed into the reconstructed distribution $g(\mathbf{s})$ generally because of limitations in the reconstruction of the data (biases), non-unitary and non-uniform efficiency in their collection and resolution effects.

Given the random nature of both the values of the variables to be observed and of the effects that limit their observation, retrieving $f(\mathbf{y})$ is a statistical estimation problem and the estimator needs to be assessed for consistency, bias, efficiency, and robustness [11, 12]. If $f(\mathbf{y},\mathbf{a})$ exists such that it is a prediction for $f(\mathbf{y})$ expressed as a function of a set of parameters $(a_1,..,a_P)$, the convolution of $f(\mathbf{y},\mathbf{a})$ with $K(\mathbf{s},\mathbf{y})$ can be compared with the observed distribution $g(\mathbf{s})$ to extract the parameter vector $\mathbf{a}$ from the data (for instance by means of a fit) and provide a complete description of $f(\mathbf{y})$. However if no parametrized prediction for $f(\mathbf{y})$ exists (as it is often the case), different techniques need to be used to estimate $f(\mathbf{y})$

from $g(\mathbf{s})$. Operatively the measurements that sample $g(\mathbf{s})$ are limited in number and affected by biases, inefficiency and imperfect resolution, so a discretized version of the integral equation 8.1 is used and a limited number of ingredients define the unfolding problem [13].

In the very common one dimensional case where both y and s are real variables, the measured distribution is approximated by the histogram representing the values ν_i, the expected number of counts in a given interval of s according to the definition

$$\nu_i = \int_{s_{i-1}}^{s_i} g(s)ds \tag{8.2}$$

where the interval of definition for s is divided in N sub-intervals by a set of (s_1,...,s_N) values and any integral of $g(s)$ over a specified sub-interval provides the total number of observed events in that sub-interval.

In a similar manner the true distribution is approximated by a histogram. The range of the allowed values for y is also divided in M sub-intervals by a set of (y_1,...,y_M) values and the expected number of counts in one of the sub-intervals is defined as

$$\mu_j = \int_{y_{j-1}}^{y_j} f(y)dy \tag{8.3}$$

The integral *kernel* $K(s,y)$ from Equation 8.1 is approximated by a response matrix $R(i,j)$ representing the probability that an event with a value of the y variable in bin j is observed as an event with a value of s in bin i. So Equation 8.1 is transformed in

$$\nu_i = \sum_{j=1}^{M} R_{i,j}\mu_j \tag{8.4}$$

where ν_i and μ_j are the expected number of reconstructed and "true" events in bins i and j respectively.

Consequently the first ingredient for the unfolding problem described by Equation 8.4 is the knowledge of the response matrix R. In general R is a rectangular matrix and by combining Equation 8.1 with Equation 8.2, it is connected to the *kernel* by the equation

$$R_{i,j} = \frac{\int_{s_{i-1}}^{s_i}\int_{y_{j-1}}^{y_j} K(s,y)f(y)dyds}{\int_{y_{j-1}}^{y_j} f(y)dy} \tag{8.5}$$

If the analytical formulation of the *kernel* is available, R can be determined directly from Equation 8.5. However most frequently R is obtained by running detailed simulation of the measuring apparatus including as many effects as possible. Monte Carlo events are generated with the best available prediction for the true distribution $f(y)$ and fully simulated with the most accurate model of the detector to produce our best guess of $g(s)$, the distribution of measured values. For some cases it is possible to measure the response to δ-like (unit-impulse) inputs that can allow to determine the *kernel* in a certain range of values, like the response of a calorimeter to a beam of particle of known energy and nature. This is equivalent to the integral $K(s,y_0) = \int_a^b K(s,y)\delta(y-y_0)dy$.

The second ingredient is the the vector of expected bin contents $\boldsymbol{\nu}$. The vector $\boldsymbol{\nu}$ is approximated by the vector $\mathbf{n} = (n_1,...,n_N)$ representing the number of observed events in each histogram bin for the variable s. By definition $\boldsymbol{\nu}$ is such that $E[n_i] = \nu_i$ where $E[n_i]$ indicates the expectation value of n_i.

Two additional ingredients are necessary to make the model built in 8.4 closer to reality.

First some interesting events are not observed due to inefficiencies in the detection or to the requirements imposed on the events properties. Such inefficiency is included in the estimate of the response matrix $R(i,j)$ with a proper normalization by defining

$$\sum R_{i,j} = \sum_{j=1}^{M} P(\text{observed in bin } i|\text{true value in bin } j) = P(\text{observed anywhere}|\text{true value in bin } j) = \epsilon_j \tag{8.6}$$

where the vector $\boldsymbol{\epsilon}$= (ϵ_1,..,ϵ_M) describes the detection efficiency as a function of the histogram bin.

Secondly some of the observed events are not interesting for the measurement one wants to perform as they are due to backgrounds (events that look like the ones of interest, but have different origin) and they modify the observed distribution. Such events have their own distribution $b(s)$ in terms of the values of the observed variable s. The vector $\boldsymbol{\beta}$ of the expected number of background events in each bin of the histogram of s can be defined as

$$\beta_i = \int_{s_{i-1}}^{s_i} b(s)ds \tag{8.7}$$

Examples of histograms [13] featuring the vectors $\boldsymbol{\mu}$, $\boldsymbol{\epsilon}$ and the corresponding vectors $\mathbf{n}$ and $\boldsymbol{\nu}$ are shown in figure 8.4.

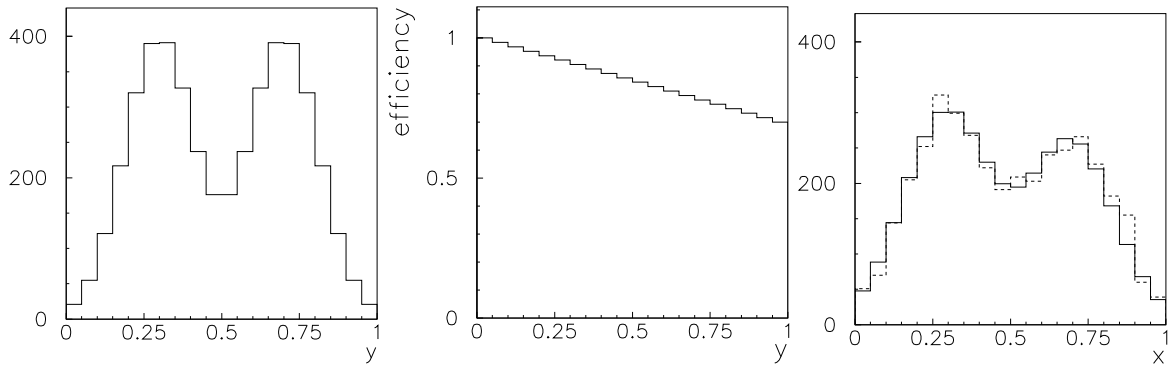


Figure 8.4: Examples of "true" distribution (left) ($\boldsymbol{\mu}$), a given set of efficiencies including resolution effects (center) ($\boldsymbol{\epsilon}$) and the corresponding observed (dashed, right) ($\mathbf{n}$) and expected observed distribution (solid, right)($\boldsymbol{\nu}$) [13]. The vectors $\boldsymbol{\mu}$, $\boldsymbol{\epsilon}$, $\mathbf{n}$ and $\boldsymbol{\nu}$ are defined in the text.

In general the model described in Equation 8.1 is then extended to

$$g(\mathbf{s}) = \int_{\Omega} K(\mathbf{s},\mathbf{y})f(\mathbf{y})d\mathbf{y} + b(\mathbf{s}) \tag{8.8}$$

and its discretized one-dimensional form described in Equation 8.4 is consequently extended [13] to

$$E[n_i] = \nu_i = \sum_{j=1}^{M} R_{i,j}\mu_j + \beta_i \tag{8.9}$$

whose vectorial compact form is

$$E[\mathbf{n}] = \boldsymbol{\nu} = R\boldsymbol{\mu} + \boldsymbol{\beta} \tag{8.10}$$

8.3 The maximum likelihood solution

Given the problem described by Equation 8.10, the formal solution is written as

$$\boldsymbol{\mu_{est}} = R^{-1}(\boldsymbol{\nu} - \boldsymbol{\beta}) \tag{8.11}$$

where R^{-1} is the inverse of R. This estimate for $\boldsymbol{\mu}$ can also be derived from the principle of maximum likelihood (ML) [14]. If one assumes (fairly generally) that events are being counted in each histogram bin and that the data are consequently independent Poisson observation distributed according to

$$P(n_i|\nu_i) = \nu_i^{n_i}\frac{e^{-\nu_i}}{n_i!} \tag{8.12}$$

the logarithm of the global likelihood $\mathcal{L} = \prod_{i=1}^{N} P(n_i|\nu_i)$ resulting from the Poisson assumption is

$$\log\mathcal{L}(\mu) = \sum_{i=1}^{N}(n_i \log \nu_i - \nu_i - \log n_i!) \tag{8.13}$$

where $\boldsymbol{\nu} = \boldsymbol{\nu}(\boldsymbol{\mu})$ because of equation 8.10. Consequently the maximum likelihood estimator for $\boldsymbol{\nu}$ obtained by imposing $\partial\log\mathcal{L}(\mu_i)/\partial\mu_i = 0 \ \forall \ i$ is given by

$$\boldsymbol{\nu}_{ML} = \mathbf{n} \tag{8.14}$$

and consequently the estimate of $\boldsymbol{\mu}$ is obtained as

$$\boldsymbol{\mu}_{ML} = R^{-1}(\boldsymbol{\nu}_{ML} - \boldsymbol{\beta}) = R^{-1}(\mathbf{n} - \boldsymbol{\beta}) = \boldsymbol{\mu}_{est} \tag{8.15}$$

Is this solution always working ? An example shown in Ref. [13] reports a double-peaked true distribution for which the resulting ML estimate, derived according to equation 8.15, shows a multi-peaked shape with extremely large variances and very large anticorrelation between neighbouring bins: the estimate turns out to be very different from known input. The response matrix R for this example has sizeable non-diagonal elements and the bin size of the histogram to be "inverted" is smaller than the detector resolution encoded in the model for event migrations. Figure 8.5 shows the generated "true" histogram $\boldsymbol{\mu}$, the observed histogram (dashed) and the corresponding expectation values (solid) and the estimator $\boldsymbol{\mu_{est}}$.

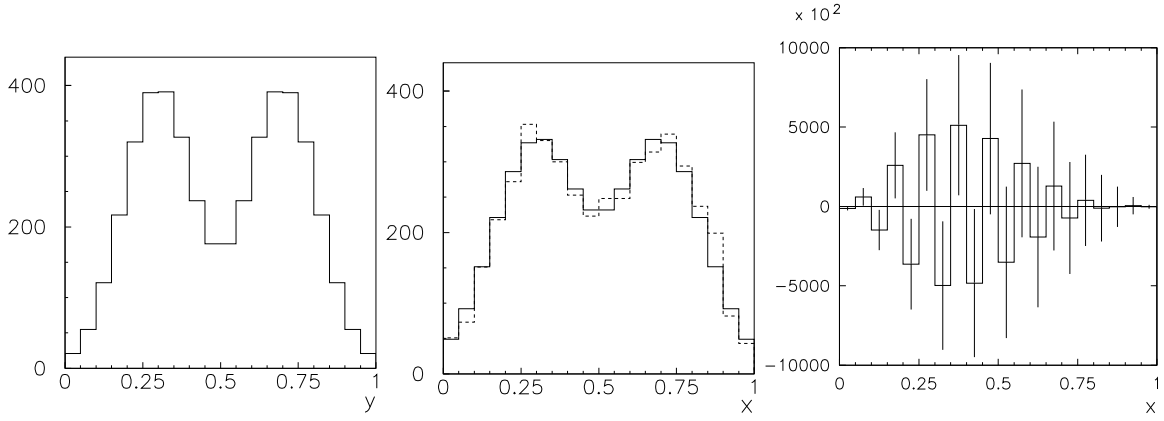


Figure 8.5: Examples of "true" distribution (left) ($\boldsymbol{\mu}$), the observed (dashed, middle) ($\mathbf{n}$) and the expected observed distribution (solid, middle) ($\boldsymbol{\nu}$) assuming imperfect resolution and perfect detection efficiency, the resulting estimate for $\boldsymbol{\mu}_{est}$ using the ML solution (right) [13]. The vectors $\boldsymbol{\mu}$, $\boldsymbol{\nu}$, $\mathbf{n}$ and $\boldsymbol{\mu}_{est}$ are defined in the text.

What is happening? Insight into the reasons for the ML result can be obtained by considering an instance where the true $\boldsymbol{\mu}$ have a fine structure and the detection effects, represented by the response matrix R, dilute the true information while allowing residual structure to be present [13]. This is shown in figure 8.6. The application of R^{-1} aims at restoring the original histogram, according to Equation 8.15. If the migrations are properly modelled, the inversion returns the correct values if the input data are the expectation vector $\boldsymbol{\nu}$ of the reconstructed bin contents. However the matrix inversion is applied to one instance of the vector $\mathbf{n}$, it is not applied to its expectation value ν. As a consequence, in a suggestively descriptive way, R "assumes" that the fluctuations in $\mathbf{n}$ are the residual of a real original structure diluted by the detection effects (and not of statistical origin) and uses the given input and the available model for migrations to reconstruct μ i.e. it magnifies the fluctuations back into the result.

Independently of the large fluctuations induced by the application of the matrix inversion the maximum likelihood solution is an unbiased estimator of $\boldsymbol{\mu}$ because

$$E[\boldsymbol{\mu}_{ML}] = E[R^{-1}(\mathbf{n} - \boldsymbol{\beta})] = R^{-1}(E[\mathbf{n}] - \boldsymbol{\beta}) = R^{-1}(\boldsymbol{\nu} - \boldsymbol{\beta}) \tag{8.16}$$

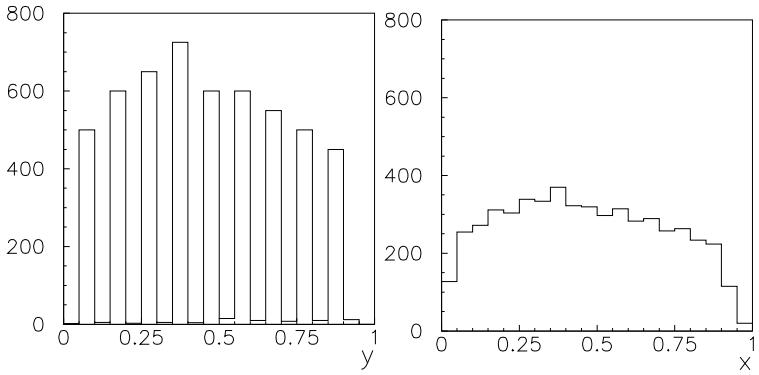


Figure 8.6: Examples of "true" distribution with fine structure (left) ($\boldsymbol{\mu}$) and the expected observed distribution (right) ($\boldsymbol{\nu}$) [13]. The vectors $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ are defined in the text.

with a covariance given by

$$U_{ML,i,j} = cov[\mu_{ML,i}, \mu_{ML,j}] = \sum_{k,l=1}^{N} R^{-1}_{i,k} R^{-1}_{j,l} cov(n_k, n_l) = \sum_{k,l=1}^{N} R^{-1}_{i,k} R^{-1}_{j,l} \delta_{k,l} \nu_k = \sum_{k=1}^{N} R^{-1}_{i,k} R^{-1}_{j,k} \nu_k \quad (8.17)$$

where $\delta_{k,l}$ is the Dirac δ symbol [1] and the equality $cov[n_k, n_k] = \nu$ uses the property of Poisson distributed data according to equation 8.12.

Under rather general condition the variance of unbiased estimators has a minimum value (effectively a lower bound) determined by the Cramér-Rao-Frechet bound [14]:

$$U^{-1}_{min,k,l} = -E[\frac{\partial^2 log\mathcal{L}}{\partial\mu_k \partial\mu_l}] = \sum_{i=1}^{N} R_{ik} R_{i,l}/\nu_i \quad (8.18)$$

If this equation is inverted [2] the minimum variance equals the ML variance obtained in Equation 8.17 i.e. $U_{i,j} = U_{min,i,j}$. Consequently the ML solution provides the unbiased estimator with the smallest variance. As a consequence estimators providing an additional reduction in variance with respect to the ML estimator will necessarily introduce a bias in the estimate of the true distribution. The balance between bias and variance is a crucial item in the unfolding procedure. Understanding the origin of the large fluctuations in the ML estimator allows to develop techniques to reduce the fluctuations (and consequently the variance of the estimator) while understanding the limitations in terms of bias of the estimator.

8.4 Correction factors: a "diagonal" ML

A simple step towards a small variance estimator consists in a simplification of Equation 8.15 derived by taking the same binning for $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ and assuming R to be diagonal (no migrations of events between bins when transforming the true distribution into the measured one). The resulting estimate for $\boldsymbol{\mu}$ is

$$\mu_{i,est} = C_i(n_i - \beta_i) \quad (8.19)$$

where C_i are correction factors (often called "bin-by-bin corrections") and they are usually derived from full simulation of the process under investigation. This provides an estimate for the expected number of reconstructed events μ_i^{MC} and true events ν_i^{MC} and the correction factors are simply derived as

$$C_i = \frac{\mu_i^{MC}}{\nu_i^{MC}} \quad (8.20)$$

[1] $\delta_{k,l} = 1$ only if k=l, it is zero otherwise.

[2] One can use equations 8.17 and 8.18 to verify that $UU^{-1}_{min} = \mathbb{1}$.

The corresponding covariance matrix is estimated [13] to be

$$U_{C,i,j} = cov[\mu_i^{MC}, \mu_j^{MC}] = C_i^2 cov[n_i, n_j] \tag{8.21}$$

The correction factor C_i is often of order unity so the variance of the estimators is not much larger than the Poisson statistical uncertainty in the data and it is typically reduced with respect to the ML estimator uncertainty. In relation to the uncertainties in Equation 8.21 a simple example due to R. Cousins and reported in Ref. [15]) points out their limitations. If one assumes that, for a given bin i of the distribution to be corrected, the values are C_i= 0.1, $\beta_i = 0$ and $n_i = 100$, the estimate $\mu_{i,C}$ for the expected number of events in this bin is obtained by $C_i n_i$= 10 and the associated standard deviation is $C_i\sqrt{n_i}$=1. However this estimate maintains that only 10 of the 100 events that are observed in the bin are actually belonging to the bin, while the remaining 90 events migrated in from other bins. It is then contradictory to have a measurement with a 10% uncertainty when there are in fact only 10 events that are actually carrying information about the bin content.

The bias corresponding to this technique, defined as E[$\mu_{i,est}$]- μ_i, is estimated [13] to be

$$b = (\frac{\mu_i^{MC}}{\nu_i^{MC}} - \frac{\mu_i}{\nu_i^{sig}})\nu_i^{sig} \tag{8.22}$$

where ν_i^{sig}= ν_i - β_i. The bias b is zero only if the simulation provides a proper description of the (unknown) true distribution and the bias pulls the result towards the values derived by the model that is used to determine the correction factor.

Ultimately the values of C_i depend circularly on the assumed true distribution one is trying to find. In addition the bin-to-bin correlations are completely neglected and uncertainties are only diagonal. The sum of the estimated events can be different from the sum of the observed number of events, differently from the ML estimator. The reduction in statistical uncertainty is obtained in exchange for a bias on the estimated result and the actual estimate of the bias is not simple. The bias is reduced if the migration between bins are a small fraction of the bins contents i.e. if the non-diagonal elements of the response matrix R are much much smaller than unity. Another visualization of this reduction is the requirement for the bin width to be large compared to the measurement resolution. Given its limitations in terms of possibly large biases, the technique of correction factors is a good tool for an initial approximation of the results, but it is generally advisable to avoid it for general use [3]

8.5 Back to basics: where to from the maximum likelihood solution?

The sensitivity to fluctuations associated with the ML solution stems from the nature of equation 8.15 : the original Fredholm equation 8.1 is an intrinsically *ill-posed* or *improper* problem [10] i.e. a problem where *"large and sometimes infinite changes in the solution could correspond to small changes in the input data"* [16] [4] In this light the stability of the solution of Equation 8.15 with respect to fluctuations can be quantified by how the uncertainties on the inputs are propagated to the output: a quantitative figure of merit for this propagation is the maximum ratio of relative precision of the estimated solution $\boldsymbol{\mu}_{est}$ of Equation 8.15 to the relative precision of the measured input vector $\mathbf{d} = \mathbf{n}$ - $\boldsymbol{\beta}$, defined as

$$c(R) = max_{\mathbf{d},\delta\mathbf{d}} \frac{\delta\boldsymbol{\mu}_{est}/\boldsymbol{\mu}_{est}}{\delta\mathbf{d}/\mathbf{d}} \tag{8.23}$$

The quantity $c(R)$ is called the *condition* of the R matrix and it is the upper bound on the magnification factor for the uncertainties on the input to the inversion. A large value for $c(R)$ implies instability under small fluctuations in the input i.e. a significant sensitivity to "noise" in the measurement.

[3] A possible exception can be some very well behaved cases with nearly diagonal response matrices where migrations effects are minimal, the expected uncertainties are well understood and the expected bias is found to be negligible in comparison to the total final uncertainties on the unfolded results (see also Section 8.13).

[4] A simple and powerful visualization of the ill-posed problem is also given in Ref. [10]: given that the *kernel* integration in Equation 8.1 tends to smooth out $f(\mathbf{y})$ and to reduce its high frequency components (edges, cusps and the like), the inversion of such a procedure will inevitably enhance the high frequency features of the input.

A deeper analysis of equation 8.15 illustrates the link between fluctuations and instability and exposes the origin of instability in a quantitative manner [17] by making a connection with the *condition* of the matrix to be inverted.

The first step is to perform a transformation of variables in equation 8.15 such that the covariance matrix $V_{\mathbf{d}}$ of the vector $\mathbf{d}$ becomes the identity matrix. In general $V_{\mathbf{d}}$ can be non-diagonal as there can be correlations between the observations in the different bins: the Poisson-based likelihood for independent observations described by Equation 8.12 is consequently extended to be

$$\mathcal{L} \propto e^{-\frac{1}{2}\chi^2(\boldsymbol{\mu},\mathbf{d})} = e^{-\frac{1}{2}(R\boldsymbol{\mu}-\mathbf{d})^T V_{\mathbf{d}}^{-1}(R\boldsymbol{\mu}-\mathbf{d})} \tag{8.24}$$

and the estimates deriving from its maximization coincide with the least squares estimate [5]. The reduction of $V_{\mathbf{d}}$ to the identity matrix allows to write the generalized likelihood of Equation 8.24 in terms of significances i.e. variables normalized to their uncertainties. The transformation of variables is a rotation in $\mathbb{R}^N$ followed by rescaling. The matrix $V_{\mathbf{d}}$ is symmetric and positive definite so there exists an $N \times N$ orthogonal matrix Q ($QQ^T = \mathbb{1}$) such that $V_{\mathbf{d}} = QV'_{\mathbf{d}}Q^T$ and $V'_{\mathbf{d}}$ is an $N \times N$ diagonal matrix such that $V'_{\mathbf{d},i,i} = v_i^2 \neq$ zero and $V'_{\mathbf{d},i,j} = 0$ for $i \neq j$. The new vector $\mathbf{d}'$ is obtained by a rotation with Q and a rescaling based on v_i as follows

$$d'_i = \frac{1}{v_i}\sum_{j=1}^{N} Q_{i,j}d_j \tag{8.25}$$

The new rotated and normalized $\mathbf{d}'$ vector encapsulates the statistical significance of the inputs (i.e. their size in units of their uncertainty) : it takes into account the different statistical power of the equation associated to each of the N input values (see Equation 8.9) . The new R' matrix is also redefined accordingly

$$R'_{i,j} = \frac{1}{v_I}\sum_{k=1}^{N} Q_{i,k}R_{k,,j} \tag{8.26}$$

so that equation 8.11 is reformulated in terms of the new variables as

$$\boldsymbol{\mu}_{est} = (R')^{-1}\mathbf{d}' \tag{8.27}$$

and the sum of squares to be minimized equivalent to the maximum likelihood is simplified to

$$\frac{1}{2}\chi^2(\boldsymbol{\mu},\mathbf{d}) = (R'\boldsymbol{\mu} - \mathbf{d}')^T(R'\boldsymbol{\mu} - \mathbf{d}') \tag{8.28}$$

The second step is to expose the decomposition of the ML solution in terms of parameters that measure the sensitivity to fluctuations in the input [10]. Such parameters can also be related to the size of the migrations described by R' (see Section 4 of Ref. [19]) i.e. the resolution and acceptance performance of the available instruments. This is done by performing a *singular value decomposition* [20] (SVD) of R' . In general a matrix R' of dimensions $M \times N$ can be decomposed as

$$R' = U\Sigma V^T \tag{8.29}$$

where U and V are unitary matrices ($U^TU = V^TV = \mathbb{1}$)) respectively of dimensions $M \times M$ and $N \times N$ and $\Sigma = U^TR'V$ is a diagonal matrix of dimensions $M \times N$ i.e. such that $\Sigma_{i,j} = \sigma_i$ if and only if $i = j$ otherwise it is zero. The σ_i values are called *singular values* of the matrix R', they are non not negative and can always be arranged in non-increasing order [10]. Both matrices U and V can be written in terms of their column vectors: $U = (\mathbf{u}_1,..,\mathbf{u}_N)$ and $V = (\mathbf{v}_1,..,\mathbf{v}_N)$. If R' is replaced by its singular

[5] In the limit of large expected number of events each independent Poisson variable described in Equation 8.12 tends to a Gaussian with the same mean and variance so the resulting likelihood $\mathcal{L}$ will tend to the diagonal multivariate Gaussian distribution $\mathcal{L} \propto e^{-(R\boldsymbol{\mu}-\mathbf{d})^T D_{\mathbf{d}}^{-1}(R\boldsymbol{\mu}-\mathbf{d})}$ where $D_{\mathbf{d},i,i} = \sigma(d_i)^2$, the uncertainly on y_i, and $D_{\mathbf{d},i,j} = 0$ for $i \neq j$ (see chapter 4 of [18]). A non-diagonal multivariate Gaussian likelihood will include correlations. An example of correlated variables is given in the case where the total number of events is a fixed quantity and the bin contents of a histogram are correlated and are distributed according to a multinomial distribution. In the limit of large number of observed and expected events in each bin, the multivariate generalization is a multivariate Gaussian [18].

value decomposition in the inversion and $\sigma_j \neq 0 \ \forall \ j$ the result is

$$\boldsymbol{\mu}_{est} = (R')^{-1}\mathbf{d}' = (R')^{-1}(\mathbf{n}' - \boldsymbol{\beta}') = V\Sigma^{-1}U^T\mathbf{d}' = \sum_{i=1}^{N}\frac{1}{\sigma_i}(\mathbf{u}_i^T\mathbf{d}')\mathbf{v_i} = \sum_{i=1}^{N}\frac{1}{\sigma_i}c_i\mathbf{v_i} \tag{8.30}$$

The singular values σ_i have important properties to characterize the unfolded result. The smoother the *kernel* corresponding to R' (i.e. the higher order continuous partial derivatives it has), the faster the decay to zero of the singular values σ_i is found to be; the smaller the value of σ_i becomes, the larger the frequency turns out to be for the component σ_i corresponds to (i.e. the more oscillations are present in the functions the corresponding *kernel* is decomposed in) [10]. The coefficients $c_i = \mathbf{u}_i^T\mathbf{d}$ can be ordered by decreasing value and they decrease rapidly with the increasing index i [21]. In addition the vector $\mathbf{c} = (c_1,..,c_N)$ has unitary covariance matrix $V_\mathbf{c} = \mathbb{1}$ because it is obtained by multiplying the unit-covariance $\mathbf{d}'$ by the orthogonal matrix U^T. These normalized coefficients encode the significance of the corresponding contribution to the ML result. The contribution of each c_i is weighted with the inverse of the corresponding singular value σ_i: small singular values can generate large fluctuations in the final ML result [21].

The quantitative connection between the singular value decomposition and the magnification of uncertainties in the unfolded result can be found in the condition $c(R')$: this can be re-written as

$$c(R') = ||(R')^{-1}\delta\mathbf{d}||/||(R')^{-1}\mathbf{d}||/||\delta\mathbf{d}||/||\mathbf{d}|| \tag{8.31}$$

and it can be shown [22] that

$$c(R') = ||R'|| \cdot ||(R')^{-1}|| = \sigma_{max}/\sigma_{min} \tag{8.32}$$

where $||\mathbf{d}||$ is the norm of the vector $\mathbf{d}$ resulting from the Euclidean positive definite metric in $\mathbb{R}^N$. For the matrix R', the norm $||R'||$ is induced by the Euclidean norm. If $A{:}\mathbb{R}^N \rightarrow \mathbb{R}^N$ is a linear application with the Euclidean norm for a vector $||\mathbf{x}|| = (\sum_i x_i^2)^{\frac{1}{2}}$ defined for both $\mathbb{R}^N$ and $\mathbb{R}^M$, the norm of the matrix A is defined as $\sqrt{\text{max eigenvalue of } A^TA}$. So the *condition* of the matrix R' can be read off from its singular value decomposition that is connected to the sensitivity to fluctuations in the unfolding problem.

The overall picture is now clearer. The singular value decomposition gives insight into the unfolding problem: ML estimators are sensitive to small effects that can lead to large changes in their values. Once the problem is described in terms of uncertainty normalized variables, the large sensitivity to small fluctuations (i.e. high frequency components, in Fourier-like language) can be derived from the high condition number $c(R)$ for the response matrix that describes the unfolding problem. In order to pose the problem more properly, it is then necessary to reduce the the impact of the low significance, highly oscillating input components while preserving the information available in the remaining high significance, more stable components. The problem is then said to have been "regularized". As the ML estimator is unbiased according to the discussion of Section 8.3, regularization inevitably leads to accepting a certain level of bias in exchange for a reduced variance. The bias is defined as the difference between the expected value of the unfolded result and the true unmeasured expected value. The heart of unfolding problems lies in understanding the balance between bias and uncertainty.

8.6 Regularized unfolding: a general view

The likelihood formulation of the unfolding problem in Equations 8.13 and 8.24 quantifies the distance between the data vector $\mathbf{n}$ and the expectation vector $\boldsymbol{\nu}$. According to that distance, in a neighbourhood of the ML solution in $\mathbb{R}^N$ the values of $\boldsymbol{\mu}$ are such that

$$\log\mathcal{L}(\boldsymbol{\mu}) \geq \log\mathcal{L}_{max} - \Delta\log\mathcal{L} \tag{8.33}$$

In order to filter out a certain amount of the high frequency components of the input and alleviate the sensitivity to large fluctuations, this distance definition can be modified with the goal to single out a modified solution that is still "close" to the unbiased ML estimate, but less sensitive to fluctuation. A

transparent way to carry out such modification is to impose constraints on the initial likelihood by adding Lagrange multipliers and describing the regularization as a maximization procedure for a new likelihood ϕ.

The logarithm of the new likelihood to be minimized then becomes

$$\phi = \alpha \log\mathcal{L}(\boldsymbol{\mu}) + S(\boldsymbol{\mu}) \tag{8.34}$$

or

$$\phi = \log\mathcal{L}(\boldsymbol{\mu}) + \tau S(\boldsymbol{\mu}) \tag{8.35}$$

where $\mathcal{L}(\boldsymbol{\mu})$ is the initial likelihood (for instance from either Equation 8.13 or Eq. 8.24), $S(\boldsymbol{\mu})$ is called regularization function, α and τ are the regularization parameters that allow to tune the strength of the constraints (equivalent a special choice of $\Delta\log\mathcal{L}$). In addition, it is possible to add the constraint that $n_{tot} = \sum_{i=1}^{N} \nu_i$ if the solution is required to provide an unbiased estimate of the total number of events. This results in the maximization of

$$\phi = \alpha \log\mathcal{L}(\boldsymbol{\mu}) + S(\boldsymbol{\mu}) + \lambda(n_{tot} - \sum_{i=1}^{N} \nu_i) \tag{8.36}$$

as a function of λ and $\boldsymbol{\mu}$. It should be noted that $\sum_{i=1}^{N} \nu_i$ is a function of μ_i as $\nu_i = \sum_{i=1}^{N} R_{i,j}\nu_j + \beta_i$. The regularization function is often perceived as a measure of the level of "smoothness" required of the maximum likelihood solution. In this light, taking for instance the formulation of Equation 8.34, if α is set to zero, the solution is set to the smooth function encoding all the constrains (i.e. available pre-existing information): the shape of $S(\boldsymbol{\mu})$ is imposed as the correct one and the data are ignored. If α tends to infinity (i.e. α is much larger than any of the other coefficients) $S(\boldsymbol{\mu})$ carries no weight in the maximization and the ML solution is re-obtained.

In the explicit formalism the ingredients for the regularization of a given likelihood $\mathcal{L}(\boldsymbol{\mu})$ are the regularization function $S(\boldsymbol{\mu})$ and a prescription for α to tune the level of filtering for the high frequency components of the input.

8.7 Regularized unfolding: the Tikhonov scheme

An analytic and quantitative measure of the smoothness of the unfolding solution is the mean square of the k^{th} derivative proposed by Tikhonov and Arsenin in Ref. [23]. The proposed form for the regularization function S is then

$$S[f(y)] = \int (\frac{d^k f(y)}{dy^k})^2 dy \tag{8.37}$$

with k in an integer number. If $k = 2$ is chosen, Equation 8.37 can be approximated by a sum over the numerical estimate of second derivative [24]

$$S(\boldsymbol{\mu}) = -\sum_{i=1}^{M-2} [(\mu_{i+2} - \mu_{i+1}) - (\mu_{i+1} - \mu_i)]^2 \tag{8.38}$$

where M is the number of values used to describe the regularization function or the number of bins used to provide its discrete description. In matrix notation it is possible to re-write $S(\boldsymbol{\mu})$ as

$$S(\boldsymbol{\mu}) = (C\boldsymbol{\mu})^T (C\boldsymbol{\mu}) \tag{8.39}$$

where C is the $M \times M$ matrix that encodes the definition of the second order numerical derivatives (see Section 6 in [19]) [6].

In the limit of large expected and observed number of events for the distribution of interest the

[6] In general a different form for C allows to use a different regularization function that is also quadratic in $\boldsymbol{\mu}$.

logarithm of the likelihood to be maximized results from combining Equations 8.24, 8.34 and 8.33 into

$$\phi(\boldsymbol{\mu},\tau) = -\frac{1}{2}\chi^2(\boldsymbol{\mu}) + \tau S(\boldsymbol{\mu}) \tag{8.40}$$

The combined likelihood $\phi(\boldsymbol{\mu},\tau)$ is a quadratic function of μ given the definition of χ^2 in Equation 8.24 and of $S(\boldsymbol{\mu})$ in Equation 8.38. Consequently the first partial derivatives with respect to μ and τ to be solved to minimize $\phi(\mu,\tau)$ return a system of linear equations.

Similarly to Section 8.5 the first step is a linear transformation such that the input variables $\mathbf{d}$ are normalized to their uncertainties and their new covariance matrix is $\mathbb{1} \in \mathbb{R}^N$ (diagonal with unitary elements). Consequently the value of $-\frac{1}{2}\chi^2(\boldsymbol{\mu})$ takes the form reported in Equation 8.28. The minimization of Equation 8.40 is then equivalent to finding the solution to the problem represented as

$$\begin{pmatrix} R'\mu \\ \sqrt{\tau}C(\mu) \end{pmatrix} = \begin{pmatrix} \mathbf{d}' \\ \mathbf{0} \end{pmatrix} \tag{8.41}$$

which can also be re-written as

$$\begin{pmatrix} R'C^{-1} \\ \sqrt{\tau}\mathbb{1}) \end{pmatrix} = \begin{pmatrix} \mathbf{d}' \\ \mathbf{0} \end{pmatrix} \tag{8.42}$$

As third step the product $R'C^{-1}$ can be expanded by a singular value decomposition like in Equation 8.29 and the solution of the problem can be written in terms of such expansion like in Equation 8.30. The major difference is the presence of the τ-dependent constraint. In order to incorporate τ in the solution of Eq. 8.42, a special linear transformation is performed, called Givens rotation [25] : this coordinate transformation sets the lower diagonal block proportional to τ to zero while transferring the information about the τ variable to the upper block. In this way the solution can be expressed as a function of τ and of the solution for τ=0 [19, 21]. The final result [21] is

$$\boldsymbol{\mu}_{est} = \sum_{i=1}^{N} \frac{1}{\sigma_i}(\tilde{\mathbf{u}}_i^T\mathbf{d}')\tilde{\mathbf{v}}_i = \sum_{i=1}^{N} \frac{1}{\sigma_i}\phi_i\tilde{c}_i\tilde{\mathbf{v}}_i \tag{8.43}$$

with $\phi_i(\tau)= \frac{\sigma_i^2}{\sigma_i^2+\tau}$ and $R'C^{-1}$ is SVD-decomposed as $R'C^{-1} = \tilde{U}\Sigma\tilde{V}^T$. The small values of σ_i are now regularized by the presence of τ so that they do not cause large fluctuations. The τ parameter acts like a cut off for a low pass filter to single out the highest frequencies causing the most rapid fluctuations. When σ_i is much smaller than τ the coefficient $\phi_i(\tau)$ behaves like σ_i/τ instead of behaving like $1/\sigma_i$ (see Equation 8.30) so ϕ_i tends to zero instead of tending to infinity and the impact of these "frequencies" is drastically reduced. It is the additional assumption on the smoothness of the solutions that reduces the importance of the solutions that result from highly oscillating solutions. While the C matrix is set by the assumption on the derivatives, the value of τ is optimized by the properties of the problem at hand. The choice of the value for the τ parameter is discussed in detail in Sections 4.3 and 7 of Ref. [19]. Here we report just the salient concepts. The reduction of the impact of higher-frequency, less statistically significant, more noise-like components is a powerful criterion for the choice of τ. The significance of each component can be read off from the coefficients of equation 8.43 similarly to the general discussion in Section 8.5. Reference [19] uses the components of the covariance-normalized, SVD-rotated input vector $\mathbf{w}$ defined as

$$\mathbf{w} = \tilde{U}\mathbf{d}' = \sum_{i=1}^{N} \tilde{\mathbf{u}}_i^T\mathbf{d}' \tag{8.44}$$

The components w_i of $\mathbf{w}$ that are of order unity or less are considered to represent statistically insignificant contributions and given that the impact of the components of σ_i larger than τ is suppressed according to equation 8.43, setting τ equal to the value of σ_k^2 for k such that $w_k \lesssim 1$ is one way to meet the chosen criterion. Additional optimization for the choice of the value of τ is possible [7], for instance by using the τ that gives that best least-squares-based comparison between a generated and the unfolded version fully

[7] The current implementation of the Tikhonov scheme with n=2 used in the ROOT Unfolding framework (RooUnfold) [51] only allows to select $\tau = s_k^2$ without any additional modification.

simulated model for the problem under consideration. Figure 8.7 shows an "academic" example [19] of unfolding simulated events with this instance of Tikhonov regularization: it reports the response matrix, the superposition of the true, reconstructed and unfolded distributions, the distribution of the w_i values and the difference between the true and the unfolded result.

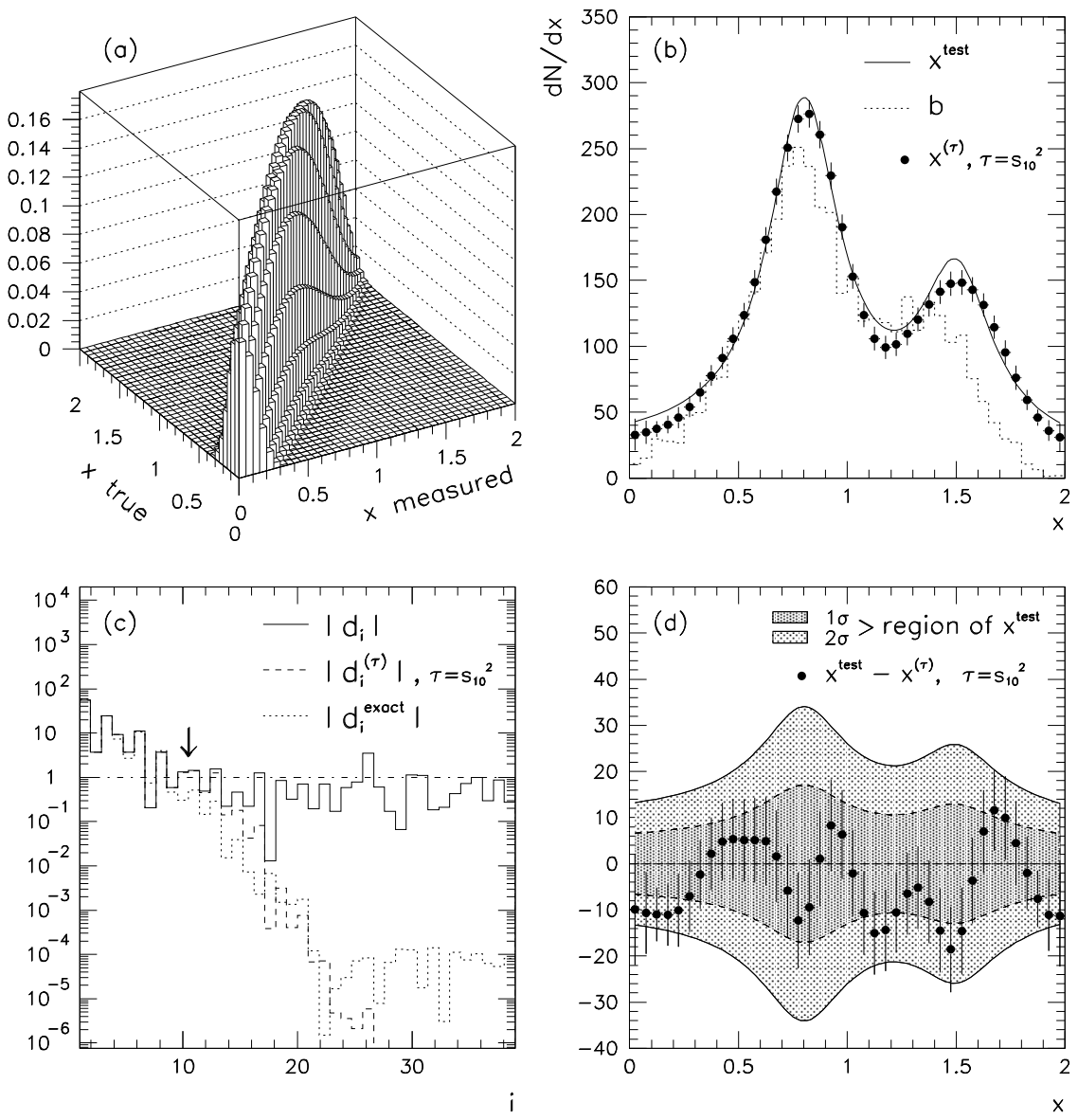


Figure 8.7: All the ingredients and results of an unfolding problem resolved with the n = 2 Tikhonov scheme from Ref. [19] (a) The response matrix connecting the true distribution to the measured one by encapsulating the model for the detection performance (b) The superposition of the true distribution (solid curve, labelled x^{test}), the measured distribution (dotted curve, labelled b) and the unfolded distribution (dots, labelled x^{tau} with the choice of $\tau = s_{10}^2$, the tenth singular value c) The superposition of three versions for the absolute value of the covariance normalized, SVD-rotated input vector called: the unregularized values (solid, labelled $|d_i|$), the regularized values(dashed, labelled d_i^τ, for $\tau = s_{10}^2$) corresponding to equation 8.44 $\mathbf{w}$, the arrow points to the boundary between significant and non-significant components of the unfolded solution. (d) the difference between the true distribution (x^{test}) and the unfolded result (x^τ) showing the one and two standard deviation statistical bands of the true distribution.

In the technical implementation of this version for Tikhonov regularization, a correction function is used to scale the simulated truth shape of the truth distribution to obtain the unfolded result. The minimization constraint is actually imposed on the curvature of this shape correction function. As a consequence the normalization information is kept in the response matrix which is no more probability-base like in Equation 8.4, but it is related to the actual number of simulated events so that the statistical uncertainties on the knowledge of the matrix elements are properly kept into account in the final result.

8.8 Iterative unfolding

A different approach to including information about the expected true distribution to counter the enhancement of fluctuations is obtained with an iterative approach.

As pointed out in Ref. [26], the initial idea of an iterative technique for solving ill-posed problems dates back to at least 40 years ago [28]. The general description outlined by Ref. [28] provides a still valid basis to understand the core concepts of a large number of the iterative schemes used at present. The inspiration is derived from Bayes' theorem [29] where the sets involved in the formulation are named *obs* and *true* to hint respectively at the observed and true number events associated to a given property [8]. In this light Bayes' theorem can be written as

$$P(true|obs) = \frac{P(obs|true)f(true)}{\int f(true)P(obs|true)dtrue} = \frac{P(obs|true)f(true)}{g(obs)} \tag{8.45}$$

where $P(x|y)$ is the conditional probability that a variable x has a certain value given the value of the variable y is between y and $y+dy$, $f(x)$ is a probability density for x and $g(obs)$ is defined as

$$g(obs) = \int f(true)P(obs|true)dtrue. \tag{8.46}$$

Inverting equation 8.45 and using the normalization properties of P(x|y), one obtains

$$f(true) = \int g(obs)P(true|obs)dobs \tag{8.47}$$

Equation 8.47 looks like the "inverse" of 8.46: it should be noted that $P(true|obs)$ is actually a function of $f(true)$ itself. The proper inverse *kernel* for f(true) needs to be a function of P(obs|true) only.

Equation. 8.45 provides the ansatz that if an initial hypothesis is made on $f(true)$, it is possible to use $P(obs|true)$ estimated from simulation to evaluate $P(true|obs)$ by defining $g(obs)$ as the convolution of $f(true)$ and $P(obs|true)$. The estimate of $f(true)$ can be re-used as initial hypothesis for an updated estimate and the procedure can be iterated. So in the the r^{th} iterative step, using Equation 8.46, $g^r(obs)$ is defined as

$$g^r(obs) = \int f^r(true)P(obs|true)dtrue \tag{8.48}$$

and consequently Equation 8.45 returns

$$P^r(true|obs) = \frac{f^r(true)P(obs|true)}{g^r(obs)} \tag{8.49}$$

Using the ansatz of Equation 8.47, the estimate $P^r(true|obs)$ is then convoluted with the observed $g(obs)_{data}$ that is estimated from data. So a new estimate for $f(true)$, the starting point for the $(r+1)^{th}$ step, is then obtained by using Equation 8.49 as follows:

$$f^{r+1}(true) = \int g(obs)_{data}P^r(true|obs)dobs = \int f^r(true)\frac{g(obs)_{data}}{g^r(obs)}P(obs|true)dobs \tag{8.50}$$

The iteration ends at the step r for which modifications to the value of $f^{r+1}(true)$ introduced by additional steps are smaller than a given tolerance value. An equivalent statement is that the iterative scheme converges if there is a step r such that $g(obs)_{data} = g^r(obs)$ [28]. The integration in Equations 8.50 will tend to remove deviations from unity in $g(obs)_{data}/g^r(obs)$ that are present on a large scale compared to the support of $P(obs|true)$. On the other hand deviations on a small length scale compared to the support of $P(obs|true)$ will be tend to be averaged out in the integration. This means that the scheme is sensitive to "long wavelength" errors in $g^r(obs)$ that are usually corrected for in the initial iterations by incorporating all the useful information on the dataset. Further iterations will take more and more into account "shorter wavelength" errors more likely deriving from statistical fluctuations in $g(obs)_{data}$:

[8] In general *obs* and *true* can be considered names of variables.

the resulting corrections will tend to match $g^r(obs)$ to those fluctuations rather than to the proper $g(obs)$ value corresponding to the best estimate of $f(true)$.

The discretized version of the iteration technique described in Ref. [26] is based on the assumption that the response matrix R is positive definite and the input binned distribution y_i in non negative so that the relation $\mathbf{d} = \mathbf{n} - \boldsymbol{\beta} = \mathrm{R}\boldsymbol{\mu}$ (from Section 8.3) can be inverted iteratively. The starting point is an hypothesis-guess on $\boldsymbol{\mu}$ called $\boldsymbol{\mu}^0$ to produce the first estimate $\mathbf{d}^0 = \mathrm{R}\,\boldsymbol{\mu}^0$. At the r^{th} step of the iteration the estimate for the vector $\mathbf{d}$ is

$$d_i^r = \sum_{j=1}^{N} R_{i,j}\mu_j^r \tag{8.51}$$

which is the discrete form of 8.48 with the correspondence $R_{i,j} \rightarrow P(obs|true)$, $\mu_j^r \rightarrow f^r(true)$. This step can be defined "folding" and it corresponds to equation 8.48. Consequently the r^{th} estimate of $\boldsymbol{\mu}$ is obtained as in Equation 8.50 by "integrating" the response matrix $R_{i,j}$ over the updating function d_i/d_i^r

$$\mu_j^{r+1} = 1/\epsilon_j \sum_{i=1}^{N} R_{i,j}\mu_j^r(d_i/d_i^r) \tag{8.52}$$

where $\mathbf{d}$ is the data input vector and corresponds to $g(obs)_{data}$ in Equation 8.50. The values of ϵ_j represent efficiencies corrections to account for experimental acceptance losses. This step can be defined as the"unfolding" one and it corresponds to equation 8.50. From Equation 8.52, the convergence of the iteration is linked to the updating function y_i/y_i^r approaching unity: in fact, given the normalization condition $\sum_i R(i,j) = 1$, $y_i/y_i^r \rightarrow 1$ implies $\mu_j^{r+1} \rightarrow \mu_j^r$. If the uncertainties are Poisson distributed such an iteration technique is empirically found to converge to the ML solution [30].

8.8.1 Iterative Unfolding: a recent example

An implementation of the general iterative approach is the technique of Ref. [31]. The standard basic iterative steps outlined above are carried out. In this approach the updating function of equation 8.52 is obtained by defining the problem in terms of the relation of "causes" to"effects". The "causes" are defined as the content C_i of the bins of the given, unknown true distribution distribution one wants to recover; the "effects" are the contents of the bins of the observed distribution E_i. The connection between the "causes" and the "effects" is obtained by the simulation-derived response matrix $P(E_J|C_i)$, the probability that a given cause C_i results into a given effect E_j (coherently with to the definition of response matrix in Section 8.2). The losses due to limitations in the observations are represented by a common"trash" bin and need to be recovered by efficiency corrections, again estimated by simulation. It is emphasized that the variables the iteration estimates are the expected contents of the bins C_i, the probability that a certain fraction of the total events are found in a given bin, not the overall probability for the distributions. So instead of writing Bayes theorem for a given distribution (spectrum) in the form of

$$P(x_C|X_E, R, I) \propto P(X_E|x_C, R, I)P(x_C|I) \tag{8.53}$$

one restarts from the probability of "causes" (i.e. the bin contents) and so each "cell" (i.e. bin content) of the discretized distribution is considered an independent cause of an effect and the probability is written for the bins as

$$P(C_i|E_J, I) \propto P(E_j|C_i, I)P(C_i|I). \tag{8.54}$$

With these definitions choosing $P(C_i|I) = p_0 = 1/M$ means that the probability content of a given cause i.e. a bin is a constant over the bins. This does not mean that all spectra have the same probably (it is not a statement on the probability of the distribution), on the contrary it implies a flat, precisely determined initial spectrum and consequently a very strong prior statement that would bias the result if no additional information were used: this is the starting point and the motivation for iterations. Due to this procedure the final estimate for the unfolded distribution is not expressed in terms of a given prior.

The iteration is then such that

$$P(C_i|E_j, I) \propto P(E_j|C_i, I)P(C_i|I). \tag{8.55}$$

The first estimate of the number of events $n(C_i)$ in the bin i of "causes" can be written as

$$n(C_i) = \frac{1}{\epsilon_i} \sum_{j=1}^{N_E} n(E_j) P(C_i|E_j) \tag{8.56}$$

This can also be written as

$$n(C_i) = \sum_{j=1}^{N_E} M_{i,j} n(E_J) \tag{8.57}$$

with

$$M_{i,j} = \frac{P(C_i|E_J, I) P_0(C_i)}{[\sum_{l=1}^{n_E} P(E_J|C_i)][\sum_{l=1}^{n_C} P(E_J|C_i) P_0(C_l)]} \tag{8.58}$$

where, in connection with Equation 8.52, $n(C_i)$ is corresponds to μ_j^1, $n(E_j)$ corresponds to d_i, $P(C_i|E_J, I)$ corresponds to $R(i, j)$, $P_0(C_l)$ corresponds to μ_i^0 and the denominator corresponds to d_i^1. The additional iterative steps follow exactly the evolution of the discrete scheme outlined in Section 8.8 (see the description of Equations 8.51 and 8.52). A distinctive feature of this approach is that the estimated distribution can be (and very often is) smoothed at each iteration step before the "unfolding" step by a customizable polynomial fit to a problem-specific function. In principle any smoothing technique can be applied. This smoothing is not applied in the last step of the iteration. The iteration is continued until the solution is considered stable. The criterion for stability is dependent on the analysis; one suggested possibility is to quantify the agreement with the previous iteration by means of a least squares test.

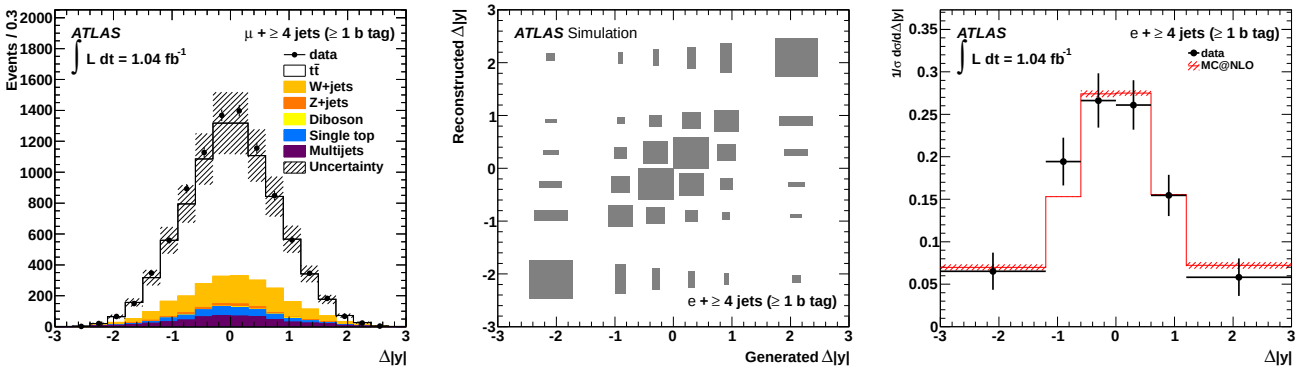
An example from particle physics of the usage of the iterative unfolding technique can be found in the measurement of the distribution of the difference between the absolute rapidities ($\Delta|y_t|$) of the reconstructed top quark and anti-top quark in a sample enhanced in $t\bar{t}$ events obtained by LHC pp collisions at $\sqrt{s} = 7$ TeV [6]. In the standard model of particle physics this distribution is expected to show a slight asymmetry (at the sub-percent level) in the amount of events with positive and negative differences [6, 32]. The asymmetry is obtained by integrating the unfolded differential $t\bar{t}$ production cross section as $\Delta|y_t|$. The migration matrix is shown in figure 8.8. The measured and unfolded distribution for one specific set of events (featuring a single electron plus jets with at least one b-tagged jet) is shown in figure 8.8. A set of basic tests are performed to choose the number of iterations, the statistical and systematic uncertainty are propagated through the unfolding scheme and the stability and robustness of the procedure is tested. The number of iterations is such that the expected variation of the value for the asymmetry is stable within 0.1% in simulated $t\bar{t}$ events. The statistical uncertainty estimate is determined with simulated experiments and the systematic uncertainty is propagated to both the response matrix and the background. Simulated $t\bar{t}$ events are re-weighted so produce different samples with different true asymmetry. The analysis is performed on each different sample and the set input asymmetries is then plotted versus the resulting reconstructed asymmetries after unfolding to check the linearity of the unfolding procedure. The small biases observed in the reconstructed distributions and the extracted asymmetry are quantified by the largest relative deviation over all the bins and the mean uncertainty-normalized relative difference between true and unfolded values from the pull distributions, respectively. Such values are used to assign additional systematic uncertainties to the unfolded distributions and the final asymmetry.

8.9 Regularization unfolding and Entropy

Information theory can provide an alternative and deeply meaningful form for the regularization function of equation 8.34. Shannon's information entropy [33] for a given distribution is defined as:

$$H = -\sum_{i=1}^{M} p_i \log p_i \tag{8.59}$$

where p_i is the probability for a given event/system to occur/be in a given subset i of the available phase



(a) (b) (c)

Figure 8.8: (a) Reconstructed distribution of the difference between the absolute rapidities of top quark and antitop quark (Δ |y|) in top quark pair events observed by the ATLAS detector in pp collisions at $\sqrt{s}$ = 7 TeV at the LHC. The observed data are represented by the dots, the predicted amount of events and their breakdown in different sources are shown in the histograms in different colours and illustrated in the legend. (b) Migration matrix from simulated top quark pair events. (c) Unfolded differential cross section for the production of top quark pair events as a function of Δ |y| (dots) compared with the prediction from the standard model (red histogram). All the plots are taken from reference [6].

space. The entropy H measures the amount of uncertainty represented by the probability distribution of a given variable and consequently determines the information content that any observation extracted from that population brings to the observer [9].

When new information about a variable is acquired the gain can be quantified by the change in uncertainty (information) between the initial estimate of the probability distribution for the variable and the new one. As the entropy H measures the information change, it is at the basis of the principle of minimum relative entropy (or cross-entropy) [34]: if there is not enough information to specify a probability distribution uniquely, a consistent estimator for it is obtained by minimizing

$$S(\boldsymbol{\mu}) = H(\boldsymbol{\mu}) = \sum_{i}^{M} \mu_i \log \frac{\mu_i}{\epsilon_i} \tag{8.60}$$

where $\boldsymbol{\mu}$ is the estimator vector for the unknown probability distribution, the index i goes from 1 to the number of M bins of the distribution and $\boldsymbol{\epsilon}$ is the reference probability distribution, representing the best knowledge about the true, unknown distribution. This method is used whenever the true distribution is known to be non-negative everywhere. When the only knowledge about the true distribution is its being non-negative and the reference distribution is taken to be a constant over all bins ($\epsilon_i = \epsilon_0 \; \forall i$), the relative entropy of Equation 8.60 is reduced to the absolute entropy of Equation 8.59 up to a constant and the principle of minimum relative entropy is equivalent to the principle of maximum entropy [35]. The axiomatic derivation [34] for the minimum relative entropy estimator defines it as the distribution μ_i that has the minimal distance from the reference, initial estimate ϵ_i in terms of information, but respects a given set of constraints.

Additional insight into the use of information entropy is provided in Ref. [36] where the minimum relative entropy estimate is interpreted as a maximum likelihood estimate. The negative logarithm of the likelihood for a given set of binned observation n_i to be compatible with a prior distribution ϵ_i and to satisfy the the response matrix constraints (see Eq. 8.24 is considered. This likelihood is shown to be proportional to the regularization function $S(\boldsymbol{\mu})$ in equation 8.60 up to a constant term (see Appendix A of [36]). The likelihood for a given set of binned observation n_i deriving from a true unknown distribution μ_i to be compatible with a prior distribution ϵ_i is represented by a multinomial distribution. The negative

[9] An outcome from a distribution with a large Shannon entropy is more useful to the observer as it is less predictable than one with small entropy (which is actually fairly predictable): the observed outcome carries more information.

logarithm of this likelihood is shown to be proportional to the cross-entropy $S(\boldsymbol{\mu})$ in equation 8.60 up to a constant term (see Appendix A of [36]). The distribution μ_i is connected to the observed data by the response matrix and the likelihood for this requirement is generally represented by the generalized $\mathcal{L}(\boldsymbol{\mu})$ of Equation 8.24. In the end in the estimate of $\boldsymbol{\mu}$ minimizing the cross-entropy $S(\boldsymbol{\mu}, \boldsymbol{\epsilon})$ with the response matrix constraint corresponds to maximizing the distribution $\phi(\boldsymbol{\mu})$ in Equation 8.34, the negative logarithm of the full likelihood for the origin and detection of the observed events, in which the cross-entropy $S(\boldsymbol{\mu})$ is interpreted as the regularization function. In addition the interpretation of $S(\boldsymbol{\mu}, \boldsymbol{\epsilon})$ as a "prior" p.d.f for μ provides the justification in a Bayesian framework [37].

8.9.1 Automatic Regularized Unfolding

An implementation of the minimum relative entropy principle to provide a the regularization function is present in the Automatic Regularized Unfolding (ARU) [38]. This scheme is presently used to perform unfolding for one-dimensional problems. The algorithm does not require any parameter to be tuned, differently from the τ parameter for the Tikhonov scheme described in Section 8.7 or the number of iterations for the iterative techniques illustrated in Section 8.8.

ARU is a regularized fit. The unfolded distribution to be found, $b(x)$, is parametrized as the sum of flexible and smooth piece-wise, non negative polynomial curves with finite support, $b_j(x)$, called B-splines [39] i.e. $b(x) = \sum_j c_j b_j(x)$ where the range of the index j is determined by number of non-zero derivatives and of grid points that characterize the B-splines chosen for the approximation [38]. This solution form is folded with the detector *kernel* $K(y, x)$ quantifying the miscalibrations, efficiencies and resolution effects to produce the function $f(y)$

$$f(y) = \int K(y,x)b(x)dx = \sum_j c_j \int K(y,x)b_j(x) = \sum_j c_j f_j(y) \tag{8.61}$$

An extended maximum likelihood fit [40] of $f(y)$ to the data is then performed by minimizing

$$L(\mathbf{c}) = L_1(\mathbf{c}) + wL_2(\mathbf{c}) \tag{8.62}$$

In this formula $L_1(\mathbf{c})$ corresponds to the negative logarithm of the overall extended likelihood function $L_1(\mathbf{c})$ = -log ($\mathcal{L}_{stand}\mathcal{L}_{norm}$). The value of $\mathcal{L}_{stand}$ is

$$\mathcal{L}_{stand} = K \prod_i \tilde{f}(y_i|\mathbf{c}) \tag{8.63}$$

where K absorbs all the normalization constants, the set of y_i are the observed values for the variable y, $\tilde{f}(y_i) = f(y_I|\mathbf{c})/v(\mathbf{c})$ and $v(\mathbf{c}) = \int dy f(y) = \sum_j c_j F_j$ with $F_j = \int dy f_j(y)$. The likelihood $\mathcal{L}_{norm}$ allows to include the variation of the normalization

$$\mathcal{L}_{norm} = v(\mathbf{c})^N \frac{e^{-v(\mathbf{c})}}{N!} \tag{8.64}$$

So, by using Equations 8.63 and 8.64 and the associated definitions, $L_1(\mathbf{c})$ can be written as

$$\begin{aligned} L_1(\mathbf{c}) = & -\log\mathcal{L}_{stand} - \log\mathcal{L}_{norm} \\ & = -\log(K\prod_i \tilde{f}(y_i|\mathbf{c})) - \log(v(\mathbf{c})^N \frac{e^{-v(\mathbf{c})}}{N!}) \\ & = -\log K - \sum_i \log\tilde{f}(y_i|\mathbf{c}) - N\log v(\mathbf{c}) + v(\mathbf{c}) + \log N! \\ & = C - \sum_i \log\frac{f(y_i)}{v(\mathbf{c})} - N\log v(\mathbf{c}) + v(\mathbf{c}) \\ & = C - \sum_i \log f(y_i) + N\log v(\mathbf{c}) - N\log v(\mathbf{c}) + v(\mathbf{c}) \\ & = C - \sum_i \log f(y_i) + \sum_j c_j F_j \end{aligned} \tag{8.65}$$

where $C = -\log K + \log N!$ includes constants that can be neglected for the purpose of minimization. $L_2(c)$ is the regularization term based on the relative entropy principle

$$L_2(c) = \int b(s)\ln\frac{b(x)}{g(x)}dx - \sum_j c_j B_j \tag{8.66}$$

where the normalization $B_j = \int b_j(x)dx$ is included. The reference distribution $g(x)$ is chosen to be uniform while the weight w is determined by minimizing the mean integrated squared error ($MISE$) on $f(y)$ (that includes an estimate of the bias)

$$MISE(f(y)) = \int dy E[(f(y) - f_{true}(y))^2] = \int dy V[f(y)] + (f(y) - f_{true}(y))^2 \tag{8.67}$$

An example of the performance of the technique is shown in figure 8.9. Unfolding is performed on a sample of one thousand events drawn from a distribution of two Gaussian convoluted with a Gaussian *kernel*. The regularized result is compared with the unregularized solution. Figure 8.10 shows the distributions obtained when performing 2000 simulated experiments with 100 or 10000 events in each experiment, respectively. The uncertainty estimate from ARU is consistent with the observed standard deviation and the average bias has the same size of the statistical uncertainty. A simulation study using 1000 pseudo-experiments of 100 and 1000 events each shows the distribution for the mean and the standard deviation of the unfolded distributions with a bias that is comparable with the statistical uncertainty of the solution.

8.10 Non-iterative Bayesian-inspired regularization

A non-iterative regularization scheme also inspired by Bayes' theorem was recently proposed [41]. The rationale is to find the probability for the spectrum of a variable as a whole, given the probability for the observed data spectrum and the migration model, according to Bayes' formula:

$$p(\mathbf{T}|\mathbf{D} \wedge \mathcal{P}) \propto P(\mathbf{D}|\mathbf{T} \wedge \mathcal{P})\pi(\mathbf{T} \wedge \mathcal{P}) \tag{8.68}$$

where $\mathbf{T} = (T_1, T_{N_t})$ is the truth level binned spectrum (event density) in an N_t-dimensional space, $\mathbf{D} = (D_1, D_{N_r})$ is the observed binned spectrum in a generally different N_r-dimensional space. D_r is assumed to follow a Poisson distribution of mean R_r where $\mathbf{R} = (R_1, R_{N_r})$ is the N_r dimensional spectrum of the expected observations. The matrix $\mathcal{P}$ is the conditional migration matrix whose element $\mathcal{P}_{t,r}$ is the conditional probability $P(r|t)$ for an event produced in the truth level bin t to be reconstructed

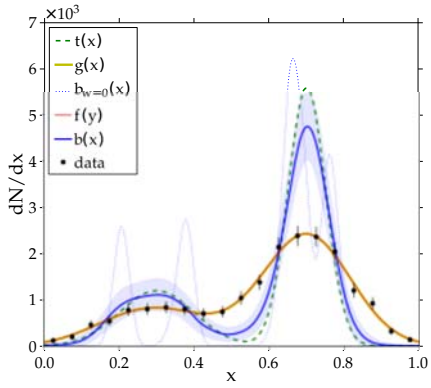


Figure 8.9: ARU-unfolded distribution of a one-dimensional variable x in a simulated experiment using a dataset of 1000 events [38]. The true distribution $t(x)$ is "smeared" into the histogram corresponding to the data points. In this case the folded distribution $f(y)$ used in equation 8.65 and the reference distribution $g(x)$ used for the regularization in Equation 8.66 are on top of each other. The regularized solution $b(x)$ is not showing undesired oscillations, differently from the unregularized solution $b_{w=0}(x)$.

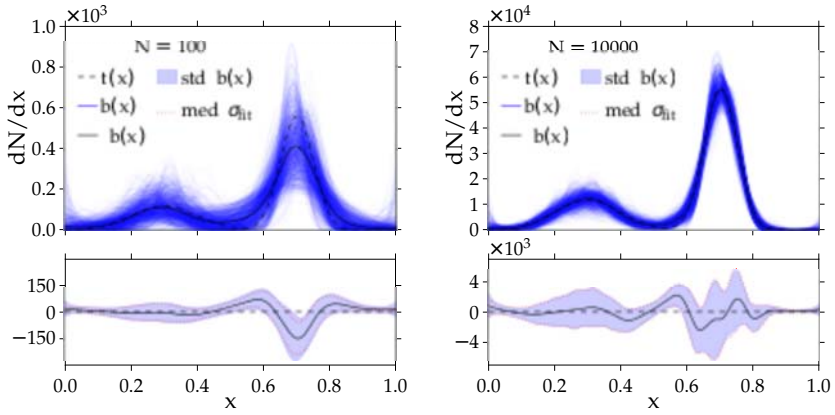


Figure 8.10: Superposed ARU-unfolded distributions $b(x)$ for a one-dimensional variable x resulting from unfolding 1000 pairs of simulated data sets randomly drawn according to the same true distribution with 100 events (left) and 1000 events (right) fir each pair respectively [38]. The upper plots also superpose the the true distribution $t(x)$ on top of the many unfolded solutions $b(x)$. The bottom plots show the bias of the unfolding defined as $b(x)$-$t(x)$, the standard deviation (std($b(x)$)) of the unfolded set of distribution and the median (med(σ_{fit}) of the estimated uncertainty on the solution.

in the reconstructed bin r. So $P(r|t)$ is defined as

$$P(r|t) = \frac{P(t,r)}{P(t)} = \frac{\mathcal{M}_{t,r}}{\epsilon_t^{-1} \sum_{k=1}^{N_r} \mathcal{M}_{t,k}} \tag{8.69}$$

where $\mathcal{M}_{t,r} = P(t,r)$ is the joint probability for an event to be produced in the truth level bin t and in the reconstructed level bin r and ϵ_t is the efficiency for reconstructing events in the bins of row t defined as

$$\epsilon_t = \frac{\sum_{r=1}^{N_r} P(t,r)}{P(t)} = \frac{\sum_{r=1}^{N_r} \mathcal{M}_{tr}}{P(t)} \tag{8.70}$$

The interpretation of Equation 8.68 is that the resulting $p(\mathbf{T}|\mathbf{D} \wedge \mathcal{P})$ is the posterior probability density function (p.d.f.) for the truth level binned spectrum $\mathbf{T}$, $P(\mathbf{D}|\mathbf{T} \wedge \mathcal{P})$ is the likelihood of the observed binned spectrum $\mathbf{D}$ as a function of $\mathbf{T}$ and $\mathcal{P}$ and $\pi(\mathbf{T} \wedge \mathcal{P})$ is the prior p.d.f of $\mathbf{T}$ and $\mathcal{P}$.

If the spectrum is such that the data are counts of events, the Poisson distribution can be used

$$P(\mathbf{D}|\mathbf{T}) = \prod_{r=1}^{N_r} Poisson(D_r|\mathbf{T}) = \prod_{r=1}^{N_r} \frac{R_r^{D_r}}{D_r!} e^{-R_r} \tag{8.71}$$

where the mean expected number of events R_r in a bin r of $\mathbf{D}$ is

$$R_r = B_r + \sum_{t=1}^{N_t} T_t P(r|t) \tag{8.72}$$

and B_r is the expected number of background in bin r.

The result of the unfolding is the posterior probability distribution function $p(\mathbf{T}|\mathbf{D})$ for the whole spectrum. The form of the posterior distribution in Equation 8.68 is the same as the regularized likelihood that generates Equation 8.35 if $\mathcal{L}(\boldsymbol{\mu})$ is identified with $p(\mathbf{T}|\mathbf{D})$ and if a function $S(\mathbf{T})$ is defined such that the prior distribution is written as

$$\pi(\mathbf{T}) = e^{\alpha S(\mathbf{T})} \tag{8.73}$$

and $S(\mathbf{T})$ is identified with $S(\boldsymbol{\mu})$. Regularization is then interpreted as the inclusion of the prior distribution i.e. the a priori degree of belief in a specified property of the "true" spectrum $\mathbf{T}$. The posterior distribution $p(\mathbf{T}|\mathbf{D})$ is used to compute the marginal posterior distributions of the content in the bins of the spectrum $p_t(T_t|\mathbf{D})$ for $t \in [1, ...N_t]$. Such distributions are defined as

$$p_t(T_l|D) = \int .. \int p(\mathbf{T}|\mathbf{D}) dT_1 .. dT_{l-1} dT_{l+1} .. dT_{N_t} \tag{8.74}$$

The estimator T_t for the content of bin t of the unfolded spectrum can then be derived by using more than one algorithm: from taking the mode or the mean of $p_t(T_l|D)$ to considering the half point of the 68% integration interval to using the mean of the Gaussian fitted to the $p_t(T_l|D)$. The uncertainty associated with the estimator is usually defined as the shorted interval in the range of T_t for which the integral of $p_t(T_l|\mathbf{D})$ amounts to 0.68. A crucial item for this technique is then the study of the convergence, stability and speed for the integration to be performed in Equation 8.74 [41]. Figure 8.11 shows an

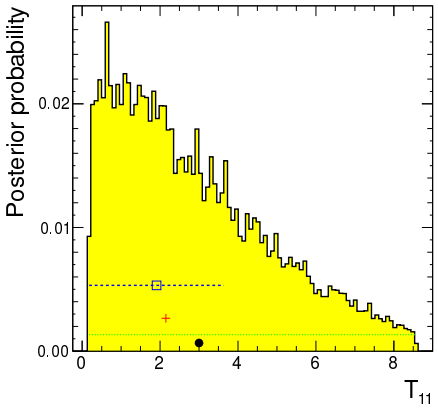


Figure 8.11: Example of one-dimensional marginal distribution for the content of one bin, $p_t(T_l|\mathbf{D})$, derived from a simulated model (see Fig 13 and section 6.4 of [41]) (yellow-filled distribution). In this specific case l=11 as it is the 11th bin (out of 14) of the overall distribution. The red cross marker represents the input truth bin content T_{11}.The black circle marker shows the simulated observed content D_{11}. The blue square marker represents the most likely value of T_{11} and the blue dashed line represents the shortest interval the integrates 68% of the marginalized distribution.The green dotted line shows the range in T_{11} that is used to sample the posterior $p(\mathbf{T}|\mathbf{D}$ to calculate the integral in equation 8.74.

example of marginalized posterior probability distribution for the content of one bin of a given"true spectrum" resulting from a simulated experiment [41]. A steeply falling distribution of a given variable

m is convoluted with an m-dependent Gaussian distribution aimed at simulating detector effects. The posterior distribution and some associated estimators are shown.

8.11 Unfolding schemes: additional examples

Other unfolding schemes tend to build on the basic techniques outlined above. The following examples are by no means an exhaustive list: its goal is to show that the problem of unfolding can be approached in a variety of manners that involve the evolution of old ideas and their merging with new schemes. Unfolding solutions vary depending on the problem at hand and unfolding schemes are used in science beyond the realms of nuclear and particle physics . These lectures should be considered a starting point to enlarge one's knowledge about the problem and an invitation to explore the available techniques and find or develop the techniques that is more suitable for the problem the reader is interested in solving.

In the realm of iterative techniques the Iterative Dinamically Stabilized (IDS) method [42] regularizes the statistical fluctuations by iterative steps in which simultaneous corrections to the normalization of the distribution and its shape are derived. Each iteration involves three stages. Initially a correction to the normalization is provided by weighting the difference between the data and the simulation (Δd) with a monotonic regularization function $f(\Delta d)$ that depends on the statistical significance of the absolute Δd. The data-simulation difference for the content in each bin k, Δd_k, is weighted with $f(\Delta d_k)$ and the "true-to-reco" migration probability estimated from simulation. This difference is then used to correct the content of bin k in the simulation of the true distribution. The second step is an improved estimate of the background subtraction and the third step is an improved estimate of the response (migration) matrix. In both steps the bin content difference between the updated unfolded result and the true simulated distribution is weighted with $f(\Delta d_k)$ (and additionally with the simulation-derived"reco-to-true" migration probability in the third step) to provide corrections to improve background treatment and migration modelling. The goal is to derive unfolding corrections that tend to preserve real new structure present in the data (but not in the simulation) while reducing the statistical fluctuations and biases due to background subtraction. An example of application of the IDS technique in particle physics is the measurement of inclusive jet and dijet production in LHC pp collisions at $\sqrt{s} = 7$ TeV using the ATLAS detector [43] where the steeply falling distribution of the transverse momentum (p_T) for jets of colourless particles is extracted by correcting the p_T distributions for jets formed by the energy depositions in the ATLAS detector.

Iterative schemes are also provided in variants that do not require binning events into histograms. Binning free methods avoid the problem of bin size optimization. They allow to transform from one variable into another and apply selection criteria, after unfolding; small size samples can be used in arbitrarily high dimensions where this becomes a problem for techniques relying on histograms. Finally the unfolded samples automatically provide the estimate of the statistical uncertainty associated to the measurement.

A first example is the Binning Free Deconvolution proposed in Ref.[44]. For each iteration two stages are proposed. The first stage is to correct the weight of each single simulated event in the distribution of the variable of interest with the ratio of the experimental local density of events to the simulated one, both estimated by counting the events corresponding to an interval around the event value for the variable under consideration. The size of the interval of values is of the order of the experimental resolution. The second stage reassigns a weight to each event by averaging, for each event, the weights of the nearest neighbours in a given region. The size of the averaging region is variable and it governs the level of smoothing in the unfolding. Events are regrouped in bins after each iteration step. The iteration stops at the step after the minimum is reached for the regularized sum of squares defined as follows: a sum of squared differences between the bin content of the data and each intermediate corrected simulated distribution is added to a Tikhonov regularization term based on a second derivative estimate (like in equation 8.40).

The second example of binning free unfolding scheme is the Satellite Method [45], a technique aiming at obtaining the deconvoluted distribution associated to a certain data set in (generally) a multi-variable space. Given the observed data sample a simulated sample with exactly the same number of the events is generated, representing the unfolded, true "positions" in the multi-dimensional space. Each true "position"

is associated to a set of "observed" positions representing the detector effects on the "true" position (de facto sampling the response *kernel*): the relative position of these "satellites" with respect to the associated "true" position is kept fixed. The number of "satellite" realizations is a tunable variable of the method. In each step of the iteration a test variable is computed to assess the quantitative compatibility of the experimental distribution and the expected density for simulated events resulting from the "satellites". The advised quantity is the statistical energy defined with the same formalism of the potential energy of two opposite charge distributions: the distributions are derived from both the experimental data and the expected simulated density and the $1/r$-law to be integrated is replaced by a general positive definite function (an exponential or a logarithm) of the distance between any two elements of the two samples. The value of the test variable is calculated at the beginning of the step and at the end, after having randomly chosen one of the "true" positions and after having moved it in the multi-variate space in a random direction, together with its satellites. The new expected density is recomputed and the test variable is recalculated: the migration is kept if the test variable has achieved a smaller value with respect to the beginning of the step, otherwise it is rejected. A new iteration starts if the test variable has not reached its minimum value. Regularization is implemented by either stopping the iteration before the results start to oscillate significantly or by varying the number of "satellite" points corresponding to a given "true" point: a larger number of satellite points implies a stronger regularization as the simulated information plays a more important role.

Finally ${}_sPlot$ [46] is a statistical technique aimed at reconstructing the unknown true distribution of a variable (control variable) for a given data set by knowing the distribution of other variables associated to the various sources of events that compose the sample (discriminating variables). The distribution of the control variable might be known for some sources of the events, but not for all. A crucial assumption is that the control variable is uncorrelated with the discriminating variables. The first step is to perform an extended maximum likelihood fit of the probability distributions for the discriminating variables to the various sources of events in the data. This fit returns the yields of the sources and determines the parameters of of the distributions. The distribution for the control variable in a given data subset (called ${}_sPlot$ for the given data subset) is then obtained by weighting all the events in the subset with a function of the distribution of the discriminating variables and their estimated covariance matrix (called sWeight). The sWeight is derived by solving a matrix equation in which the average of the control variable ${}_sPlot$ for the given data subset is expressed as a linear combination of the control variable special distributions (${}_{in}Plot$s) for all the subsets of the data and the coefficients are provided by the covariance matrix of the discriminating variables. The ${}_{in}Plot$ for a given data subset is the control variable distribution obtained by expressing the control variable as a function of the discriminating variables. The ${}_{in}Plot$ for a given data subset is obtained by weighting all the data events in a given control variable bin with the probability that each event belongs to the data subset of interest: the value of the discriminating variable for each event is fed to the known discriminating variable distribution to obtain such probability.

8.12 Unfolding software tools

Some of the most recent unfolding schemes used in particle physics are now available as updated and maintained public software tools. These make it much simpler to use the proposed techniques and to provide feed-back that can be included in new versions, with general benefit to the users' community.

A sizeable collection of the available public software (and documentation) to perform unfolding in particle physics is reported under the Unfolding Framework Project [50].

Amongst the available tools, the ROOT Unfolding framework (RooUnfold) [51] usefully collects five unfolding schemes, including a basic implementation of unregularized ML unfolding through matrix inversion (see Section 8.3), the correction factor scheme outlined in Section 8.4, the regularized Tikhonov approach using the second order derivative described in Section 8.7 and a version of the iterative Bayesian-inspired scheme illustrated in Section 8.8.1. RooUnfold is a coherent C++ framework that can be linked against the ROOT libraries [52] that are widely used in particle physics analysis.

The most updated version of the iterative scheme of Section 8.8.1 is available at [53].

A precursor for the implementation of the Tikhonov scheme outlined in Section 8.7, coupling the second derivative regularization with a B-spline-based parametrization, is available in the FORTRAN-

based program RUN [55]. The actual FORTRAN implementation of the Tikhonov scheme of Section 8.7 is available in the program GURU [56] and a C++ wrapper is also available [57] outside of ROOT.

The ARU scheme described in Section 8.9.1 is available in C++ format with a Python-based interface [58].

8.13 Optimization and choice of unfolding technique: the last two cents

Given an unfolding problem, the optimization of the balance between the bias and the total expected uncertainty, including full statistical and systematic effects, is a powerful criterion against which to scan the available degrees of freedom: unfolding scheme, binning of data, regularization parameter. Such optimization is usefully developed on large samples of simulated data so as to avoid biases induced by the specific features of the data under consideration. Quantitative figures of merit can be derived from the available information to drive the optimization [13, 47]. In the very common case of binned data, a basic figure of merit related to the total uncertainty is the mean squared averaged uncertainty over the bins of the distribution, defined as

$$MSE = \frac{1}{M}\sum_{i=1}^{M}(U_{i,i} + b_i^2) \tag{8.75}$$

where $U_{i,i}$ is the diagonal element of the covariance matrix of the unfolded estimates for the ith bin content and b_i is the estimated bias on each bin content as defined at the end of in section 8.5 and M is the number of bins in the distribution. In general different bins have different uncertainties. A modified MSE can take this into account by weighting the contribution of each bin with the inverse of the bin variance and by remembering that the mean and the variance of the Poisson-distributed content of bin i ,μ_i, are equal:

$$MSE' = \frac{1}{M}\sum_{i=1}^{M}\frac{U_{ii} + b_i^2}{\mu_i} \tag{8.76}$$

Another possible driving criterion is to require that the estimated bias squared, b_i^2, be smaller than its own variance $V_{ii} = cov[b_i, b_i]$ so that there is no statistically significant bias. In this way the prescription can be :

$$\sum_{i=1}^{M} b_i^2/V_{ii} = M \tag{8.77}$$

If the bias were statistically significant a correction for it could be considered and the uncertainty on the bias would then have to be included as systematic uncertainty on the final result.

The inclusion of systematic uncertainties in the measurement needs to take into account the correlations that the unfolding introduces on a bin-by-bin basis and to derive their combined impact on the measurement. As an example, a very powerful way to achieve this is to use pseudo-experiments [10] or toy experiments [11]: each experimental variable derived in a given pseudo- or toy experiment is the result of the sum of the effects of modifying a set of (assumed) independent modelling parameters [12]. The distributions of the modelling parameters are derived from some ancillary measurement or from an explicitly mentioned assumed distribution (usually a Gaussian or a Log-Normal distribution) and the effects are obtained by sampling such independent distributions. By calculating the observable(s) of interest in each experiment, its (their) distribution(s) can be calculated and, for instance, an estimator(estimators) for the mean(s) and the variance(s) can be obtained from the calculated distribution(s). The correlation between the various bin content can also be estimated by taking averages and variances over the pseudo- or toy experiments. This is a essentially Bayesan-inspired way of marginalizing the modelling parameters in the

[10] A set of simulated events including a break down of "signal" and "background' events" with a mixture and a total size that is aimed at reproducing the available dataset. A large number of statistically independent mixtures are realized so as to simulate many instances of the experiment.

[11] A set of simulated distributions derived from assuming a probability distribution and its parameter and generating sampled distributions corresponding to the p.d.f model.

[12] For instance the a modification in the jet energy scale as a function of jet transverse momentum will cause a variation in the reconstructed mass of the top-antitop system decaying to a lepton, missing transverse energy and jets.

likelihood through Monte Carlo integration [59, 60]. An example of the formalism for the inclusion of the modelling parameters can be found in Ref [61] (see section 2 and Appendix A). The unfolding ingredients where the variations of the parameters need to be implemented are easily read off from the general likelihood solution and its elements shown in Equations 8.11, 8.13 and 8.36: the response matrix, the estimate of non interesting events (backgrounds), the acceptance corrections all need to be varied when input to the unfolding procedure according to their relation with the independent modelling parameters. In this way the all the correlations introduced by the unfolding are automatically kept into account.

In relation to the optimization, the variables that provide crucial insight in the origin of the unfolding problem represent sensitive tools to understand the role of the balance between statistical and systematic uncertainties. An important example is the condition number $c(R)$ defined in Equation 8.23: this quantifies the general sensitivity of the analysis to incoming fluctuations, independently of their origin. When exploring the unregularized maximum likelihood solution, the condition number will change depending on the binning of the events and on the inclusion of certain systematic uncertainties. In the regularized realm an important example is the choice of τ which is based on the distribution of the input vector w_i defined in Equation 8.44 Such distribution is driven by the combination of information of the response matrix and the statistical covariance matrix of the inputs linked by the specific curvature-based regularization (as illustrated in Section 8.7): in general systematic uncertainties are not included in the input covariance matrix, so the optimal values for the τ parameter chosen according to the criteria mentioned in Section 8.7 need to be extended to reflect those effects, by considering the expected variations of w_i due to the different systematic effects.

The choice for the algorithm and the optimization of its parameters is definitely dependent on the data analysis that is being carried out. Whatever the technique chosen for unfolding, it is useful to produce (and possibly report) the unregularized maximum likelihood solution using the matrix inversion solution of Equation 8.11: the unfolding technique does not add any bias to the result (see section 8.3) and it is equivalent to using the uncorrected data to test a theory or estimating its parameters, for instance by minimizing the sum of squares [47]

$$\chi^2(\boldsymbol{\theta}) = (\boldsymbol{\mu}(\boldsymbol{\theta}) - \boldsymbol{\mu}_{ML})^T U^{-1} (\boldsymbol{\mu}(\boldsymbol{\theta}) - \boldsymbol{\mu}_{ML}) \tag{8.78}$$

where U^{-1} is the covariance matrix on the unfolded measurement and $\boldsymbol{\theta}$ is the vector of parameters to be estimated by the minimization. The important point is that the full covariance matrix for the unfolded result needs to be used.

Finally it is crucial to devise a test for stability and bias to check whether the unfolding scheme is robust with respect to possible fluctuations and what level of bias can be recovered by the unfolding. This is particularly important to build confidence that the unfolding is not diluting or cancelling important unexpected features of the data that are not foreseen by the models underlying the unfolding scheme. Typical tests are performed by unfolding simulated events whose true shapes are distorted with respect to the best knowledge encoded in the theoretical models included in the simulations. The induced distortions should be comparable with the total statistical and systematic uncertainty of the measurement. The distorted simulated test distributions are then unfolded and the deviation with respect to the input can be quantified, for instance, by using a least squares test between the input model and the unfolded result and exploiting the full covariance matrix including both statistical and systematic uncertainties. This provides a quantitative assessment of the capacity of the unfolding scheme to recover the input distortion and its ability to maintain "unforeseen" features present in the data.

Extremely useful insight into the unfolding problem is given in the lectures of Ref. [47] and Ref. [11], the chapter on introduction to statistics in Ref. [48] and the slides and proceedings of the PHYSTAT 2011 conference dedicated to unfolding [49].

8.14 Acknowledgements

The author would like to thank the organizers of SOS2012 for the invitation to contribute to the school and for the organization of a very enjoyable and enriching event. It is a pleasure to thank Glen Cowan for various enlightening discussions about the crucial aspects of unfolding and Jorgen Sjölin for many eye-opening comments about on the inclusion of systematic uncertainties in both regularized and unregularized

unfolding applications in particle physics analysis. Kyle Cranmer and Alex Read deserve thanks for a clarifying discussion on inclusion of systematic uncertainties through Monte Carlo integration.

References

[1] Y. Vardi, L. A. Shepp, L. Kaufman, *"A Statistical Model for Positron Emission Tomography"*, Journal of the American Statistical Association, Vol. 80, No. 389 (Mar., 1985), pp. 8-20 [http://www.jstor.org/stable/2288030]

[2] O. Helene, V. R. Vanin, Z. O. Guimaraes-Filho, C. Takiya, *"Variances, covariances and artifacts in image deconvolution"*, Nucl. Instr. Meth. **A**, 580 (2007) pp.1466-1473

[3] L. Evans, P. Bryant (editors), *"LHC Machine"*, 2008 JINST **3**, S08001, doi:10.1088/1748-0221/3/08/S08001 [http://iopscience.iop.org/1748-0221/3/08/S08001]

[4] the ATLAS Collaboration, *"The ATLAS Experiment at the CERN Large Hadron Collider"*, JINST **3**, 2008, S08003, doi:10.1088/1748-0221/3/08/S08003 [http://iopscience.iop.org/1748-0221/3/08/S08003]

[5] V. Ahrens, A. Ferroglia, M. Neubert, B. D. Pecjak and L. L. Yang, "Renormalization-Group Improved Predictions for Top-Quark Pair Production at Hadron Colliders", JHEP **1009**, 097 (2010) [arXiv:1003.5827 [hep-ph]].

[6] the ATLAS collaboration, *"Measurement of the charge asymmetry in top quark pair production in pp collisions at $\sqrt{s} = 7$ TeV using the ATLAS detector"*, Eur. Phys. J. C **72**, 2039 (2012) [arXiv:1203.4211 [hep-ex]].

[7] Computer generated image of the whole ATLAS detector, CERN-GE-0803012, Photograph: Joao Pequenao, [http://cds.cern.ch/record/1095924/]

[8] Figures produced by the D0 collaboration [9] available at http://www-d0.fnal.gov/Run2Physics/top/top_public_web_pages/top_feynman_diagrams.html

[9] The D0 experiment, http://www-d0.fnal.gov

[10] P C Hansen, *"Numerical tools for analysis and solution of Fredholm integral equations of the first kind"*, Inverse Problems **8** (1992) 849 doi:10.1088/0266-5611/8/6/005 [http://m.iopscience.iop.org/0266-5611/8/6/005?rel=sem&relno=7]

[11] Volker Blobel, *"Unfolding methods in high energy physics experiments"*, Report DESY 84-118, 1984 (also in Proceedings of the 1984 CERN School of Computing, CERN 85-09, pp. 88-127; see also http://www.desy.de/ blobel/).

[12] J. Beringer *et al.* (Particle Data Group), *"The Review of Particle Physics"*, Phys. Rev. **D**86, 010001 (2012) [http://pdg.lbl.gov/]

[13] G. Cowan, "A survey of unfolding methods for particle physics", Conf. Proc. C **0203181**, 248 (2002).

[14] *"Statistics"* (Revised by G. Cowan) in [12]

[15] L. Lyons, *"Unfolding: Introduction"*, in Proceedings of the PHYSTAT 2011 Workshop on Statistical Issues Related to Discovery Claims in Search Experiments and Unfolding, CERN, Geneva Switzerland, 17-20 January 2011, edited by H.B.Prosper, L.Lyons, CERN-2011-006, pp. 225-228 [http://cds.cern.ch/record/1306523] and references to unfolding therein.

[16] I. P. Nedelkov, *"Improper problems in Computation Physics"*, Com. Phys. Comm. **4** (1972) 157

[17] S. Leach, *" Singular Value Decomposition. A Primer"*, [http://people.csail.mit.edu/hasinoff/320/SingularValueDecomposition.pdf], material from CSC320S: Introduction to Visual Computing course at MIT and references therein.

[18] A. G. Frodesen, O. Skjeggestad, H. Tofte, *"Probability and Statistics in particle physics"*, Hardcover: 501 pages, Publisher: Universitetsforlaget (September 1979), ISBN-10:8200019063, ISBN-13: 978-8200019060

[19] A. Hoecker, V. Kartvelishvili, *"SVD Approach to data unfolding"*, Nucl. Instr. Meth. **A 372**, 1996 (469)

[20] A. Björck, *"Least squares methods"*, Handbook of Numerical Analysis, voI I. (1990) 465-652, ed P. G. Ciarlet and J. L. Lions (Amsterdam: Elsevier)

[21] V. Blobel, *"Unfolding methods in Particle Physics"*, in Proceedings of the PHYSTAT 2011 Workshop on Statistical Issues Related to Discovery Claims in Search Experiments and Unfolding, CERN, Geneva Switzerland, 17-20 January 2011, edited by H.B.Prosper, L.Lyons, CERN-2011-006, pp. 240-251 [http://cds.cern.ch/record/1306523] and references to unfolding therein.

[22] See for instance H. M. Antia, *"Numerical methods for scientists and Engineers"*, Birkhäuser, 2nd edition, (2002)

[23] A. N. Tikhonov ,V. Y. Arsenin *"Solutions of ill-Posed Problem"* Wiley, New York, (1977)

[24] See for instance T. M. Apostol (June 1967), *"Calculus, Vol. 1: One-Variable Calculus with an Introduction to Linear Algebra 1"* (2nd ed.), Wiley, ISBN 978-0-471-00005-1

[25] C. E. Lawson and R. J. Hanson,*"Solving Least Square Problems"*, Prentice-Hall Inc., Englewood Cliffs, 1974.

[26] G. Zech *"Regularization and error assignment to unfolded distributions"*, in Proceedings of the PHYSTAT 2011 Workshop on Statistical Issues Related to Discovery Claims in Search Experiments and Unfolding, CERN, Geneva Switzerland, 17-20 January 2011, edited by H.B.Prosper, L.Lyons, CERN-2011-006, pp. 252-259 [http://cds.cern.ch/record/1306523] and references to unfolding therein.

[27] H. N. M̀ulthei, B. Schorr, |em " On an iterative method for the unfolding of spectra", Nucl. Instr. and Meth. A257 (1987) 371-377

[28] L. B. Lucy *"An iterative technique for the rectification of observed distributions"*, Astronomical Journal 79(6) (1974) 745

[29] See section 36.1.4. in Ref [14] and references therein.

[30] See the articles in reference 1 of [26], particularly [1] and [27].

[31] G. D'Agostini, *"A multidimensional unfolding method based on Bayes' theorem"* , Nucl. Instr. Meth. **A 362** 1995 (487)

G. D'Agostini, *" Improved Iterative Bayesian unfolding"*, http://arxiv.org/abs/1010.0632

[32] J. H. Kuhn and G. Rodrigo, *"Charge asymmetries of top quarks at hadron colliders revisited"*, JHEP **1201**, 063 (2012) [arXiv:1109.6830 [hep-ph]].

[33] C. Shannon *"A mathematical Theory of Communication"* Bell System Technical Journal **27** (3) 379-423

[34] J. E. Shore , *"Relative Entropy, Probabilistic Inference and AI"* , contribution to Proceedings of the First Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-85), Corvallis, Oregon, 1985, pp 43-47, AUAI Press [http://uai.sis.pitt.edu/papers/85/p43-shore.pdf]

[35] E. T. Jaynes, *"Information Theory and Statistical Mechanics"*, Phys. Rev. 106 (1957) 620

[36] M. Schmelling, *"The method of reduced cross-entropy. A general approach to unfold probability distributions"*, Nucl. Instr. Meth. **A 340** (1994) 400-412

[37] J. Skilling, *"Quantified Maximum Entropy"*, in Maximum Entropy and Bayesian Methods, Fundamental Theories of Physics, vol. 39, 1990, pp 341-350 and ed. P.F. Fougère (Kluwer, Dordrecht, Holland, 1990).

[38] H. P. Dembinski, M. Roth, *"ARU - towards automatic unfolding of detector effects"* in Proceedings of the PHYSTAT 2011 Workshop on Statistical Issues Related to Discovery Claims in Search Experiments and Unfolding, CERN, Geneva Switzerland, 17-20 January 2011, edited by H.B.Prosper, L.Lyons, CERN-2011-006, pp. 285-291 [http://cds.cern.ch/record/1306523] and [http://aru.hepforge.org]

[39] C. de Boor, *"A Practical Guide to Splines"*, Springer Verlag (New York, Heidelberg, Berlin) (1978).

[40] R. Barlow, *"Extended maximum Likelihood"*, Nucl. Instrum. Meth. **A297**, 496 (1990) and references therein.

[41] G. Choudalakis, *"Fully Bayesian Unfolding"*, [arXiv:1201.4612[physics.data-an]]

[42] B. Malaescu, *"An iterative, dynamically stabilized method of data unfolding"*, [arXiv:0907.3791 [physics.data-an]]

[43] G. Aad *et al.* the ATLAS Collaboration,*"Measurement of inclusive jet and dijet production in pp collisions at* $\sqrt{s} = 7$ *TeV using the ATLAS detector"*, Phys. Rev. D **86**, 014022 (2012) [arXiv:1112.6297 [hep-ex]].

[44] L. Lindemann, G. Zech, *"Unfolding by Weighting Monte Carlo Events"* , Nucl. Instr. Meth **A** 354 (1995) 516-521

[45] B. Aslan and G. Zech, *"Statistical energy as a tool for binning-free, multivariate goodness- of-fit tests, two-sample comparison and unfolding"*, Nucl. Instr. and Meth. **A** 537 (2005) 626

[46] M. Pivk, F. R. Le Diberder, *" $_s$Plot a statistical tool to unfold data distributions"* , Nucl. Inst. Meth. **A** 555:356-369, (2005)

[47] G. Cowan, *"Statistics for HEP. Lecture 4:Unfolding"*, CERN Academic Training Lectures, CERN, Geneva, Switzerland, 5th April 2012, [http://indico.cern.ch/conferenceDisplay.py?confId=173729]

[48] G. Bohm and G. Zech, *"Introduction to Statistics and Data Analysis for Physicists"*, Verlag Deutsches Elektronen-Synchrotron (2010) [http://www-library.desy.de/elbook.html]

[49] Proceedings of the PHYSTAT 2011 Workshop on Statistical Issues Related to Discovery Claims in Search Experiments and Unfolding, CERN, Geneva Switzerland, 17-20 January 2011, edited by H.B.Prosper, L.Lyons, CERN-2011-006 [http://cds.cern.ch/record/1306523] and [http://indico.cern.ch/conferenceOtherViews.py?view=standard&confId=107747]

[50] The Unfolding Framework Project, [https://www.wiki.terascale.de/index.php/Unfolding_Framework_Project] (accessed on 9th May 2013) with software and references therein.

[51] T. Adye, *"Unfolding algorithms and tests using RooUnfold"* in Proceedings of the PHYSTAT 2011 Workshop on Statistical Issues Related to Discovery Claims in Search Experiments and Unfolding, CERN, Geneva Switzerland, 17-20 January 2011, edited by H.B.Prosper, L.Lyons, CERN-2011-006, pp. 313-318 [http://cds.cern.ch/record/1306523] and [http://hepunx.rl.ac.uk/ adye/software/unfold/RooUnfold.html]

[52] R. Brun and F. Rademakers, *"ROOT - An Object Oriented Data Analysis Framework"*, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86. See also http://root.cern.ch/.

[53] G. D'Agostini, *"Probabillity and Statistics - Improved iterative Bayesian unfolding"* [http://www.roma1.infn.it/∼dagos/unf2_R.tgz] written using the R Framework [54]

[54] R Development Core Team (2009), x*"R: A language and environment for statistical computing"*, R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, [http://www.r-project.org].

[55] V. Blobel, *"Unfolding"*, RUN program source files and Manual, [http://www.desy.de/∼blobel/unfold.html]

[56] V. Kartvelishvili, *"GURU"*, [http://www.hep.lancs.ac.uk/guru.tar.gz]

[57] G. Hesketh, *"Unfolding"*, [http://www-d0.fnal.gov/ ghesketh/unfolding/]

[58] H. P. Dembinski, M. Roth, *"ARU development page"*, [http://aru.hepforge.org]

[59] M. H. Kalos, P. A. Whitlock, *"Monte Carlo Methods, Volume 1"*, Wiley-VCH Publisher (John Wiley & Sons, Inc.), 2nd Edition, (2008)

[60] R. D. Cousins, V. L. Highland, *"Incorporating systematic uncertainties into an upper limit"* , Nucl. Instr. Meth. **A.320** (1992) 331-335

[61] the ATLAS Collaboration, *"Procedure for the LHC Higgs boson search combination in summer 2011"*, ATL-PHYS-PUB-2011-011 [https://cds.cern.ch/record/1375842] and references therein.

Setting limits and application to Higgs boson search

Luca Lista
INFN Napoli

9.1 Introduction

Experiments searching for rare or unknown processes have to quantify how *evident* the signal they look for is. The evidence is not always sufficient to claim a discovery, and in many cases it is interesting to quote among the published results the upper limit on the expected signal yield. From such limit, one can indirectly derive limits on the properties of the new signal that influence the signal yield, such as the mass of a new particle.

The determination of upper limits is in many cases a complex task and the computation frequently requires numerical algorithms. Several methods are adopted in High Energy Physics and are documented in literature to determine upper limits. The interpretation of the obtained limits can be, even conceptually, very different, depending on the adopted method.

This lecture summarizes the basic concept of hypothesis testing, will introduce the concepts of significance and upper limit under the frequentist and Bayesian approaches, and will discuss the benefits and limitations of the most popular approaches. Special attention will be devoted to the so-called modified frequentist approach, which is a popular method in High Energy Physics, and some application to real physics cases will be discussed.

9.2 Hypothesis testing

A key task in most of physics measurements is to discriminate between two or more *hypotheses* on the basis of the observed experimental data. One typical case is to discriminate a signal under study against background processes. This problem is addressed in statistics by the *hypothesis tests*, which defines a procedure to assign an observation to one of two or more hypothetical models considering their predicted probability distributions. One typical example is to determine whether a sample of events is composed of background only or contains a mixture of background plus signal events. The discrimination between the two hypotheses can be performed on a statistical basis looking at the observed measurements of specific discriminating variables. Another typical example in physics is the identification of a particle type (e.g.: as a muon vs pion) on the basis of the measurement of a number of discriminating variables (e.g.: the depth of penetration in an iron absorber or the energy release in scintillator crystals, etc.).

In literature typically two hypotheses are considered called *null hypothesis*, H_0, and *alternative hypothesis*, H_1. Assume that the observed data sample consists of the measurement of a number k of variables, $\vec{x} = (x_1, \cdots, x_k)$ which are randomly distributed according to some probability density function (PDF), which is in general different for the hypotheses H_0 and H_1. A measurement of whether the observed data sample better agrees with H_0, or rather with H_1 can be given by the value of a function $t(\vec{x})$, called *test statistics*, whose PDFs under the considered hypotheses can be derived from the PDFs of $\vec{x}$. One simple example is the use of a single variable x which has discriminating power between two hypotheses, as shown in Fig. 9.1, in the sense that the PDF of x under the hypotheses H_1= *signal* and H_0=*background* are appreciably different. On the basis of the observed value $\hat{x}$ of the discriminating variable x the test statistics can be defined as the measured value itself:

$$\hat{t} = t(\hat{x}) = \hat{x}\,. \tag{9.1}$$

A selection requirement (in jargon always called *cut*) can be defined by identifying a particle as a muon

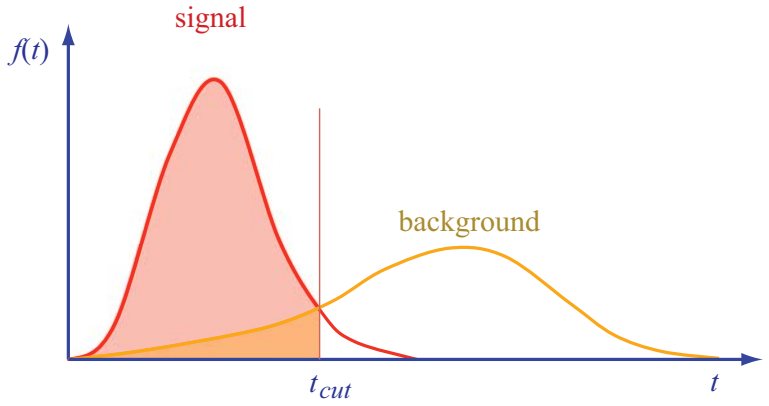


Figure 9.1: Probability distribution for a discriminating variable $t(x) = x$ which has two different PDF for the signal (red) and background (yellow) hypotheses under test. Applying a selection *cut*, in this case $t \leq t_{\rm cut}$, enriches the selected data sample of signal, reducing the fraction of background.

if $\hat{t} \leq t_{cut}$, or as a pion if $\hat{t} > t_{cut}$, where the value t_{cut} is chosen by the experimenter. Not all real muons will be correctly *identified* as muon according to this criterion, as well as not all real pions will be correctly identified as pions. The expected fraction of selected signal particles (muons) is usually called signal *selection efficiency* and the expected fraction of selected background particles (pions) is called *misidentification probability*. Misidentified particles constitute a background to positively identified signal particles.

Statistical literature defines the *significance level* α as the probability to reject the hypothesis H_1 if it is true. The case of rejecting H_1 if true is called *error of the first kind*. In our example, this means selecting a particle as a pion in case it is a muon, hence the *selection efficiency* for the signal corresponds to $1 - \alpha$. The probability β to reject the hypothesis H_0 if it is true (*error of the second kind*) is the *misidentification probability*, i.e.: the probability to incorrectly identify a pion as a muon, in our case. Varying the value

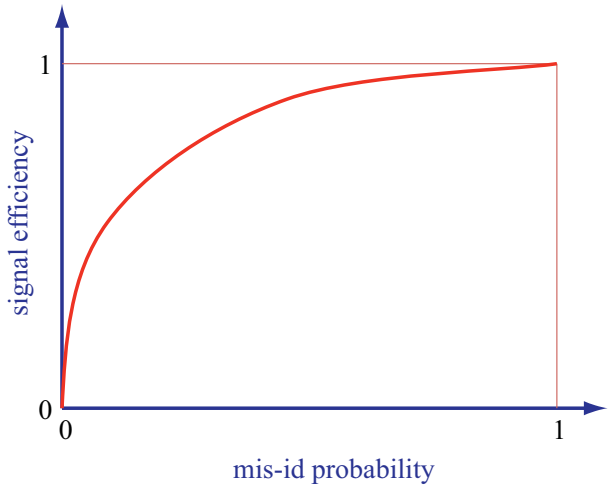


Figure 9.2: Signal efficiency versus background misidentification probability.

of the selection cut t_{cut} different values of selection efficiency and misidentification probability and the corresponding values of α and β are determined. A typical curve representing signal efficiency versus misidentification probability is shown in Fig. 9.2. A good selection should have low misidentification probability corresponding to high selection efficiency. But clearly the background rejection can't be perfect if the distributions $f(x|H_0)$ and $f(x|H_1)$ overlap, as in Fig. 9.2.

More complex examples of cut-based selections involve multiple variables, where selection requirements in multiple dimensions can be defined as regions in the discriminating variables space. Events are accepted

as "signal" or as "background" if they fall inside or outside the selection region. Finding an optimal selection in multiple dimensions is usually not a trivial task. Two simple example of selections with very different performances in terms of efficiency and misidentification probability are shown on Fig. 9.3.

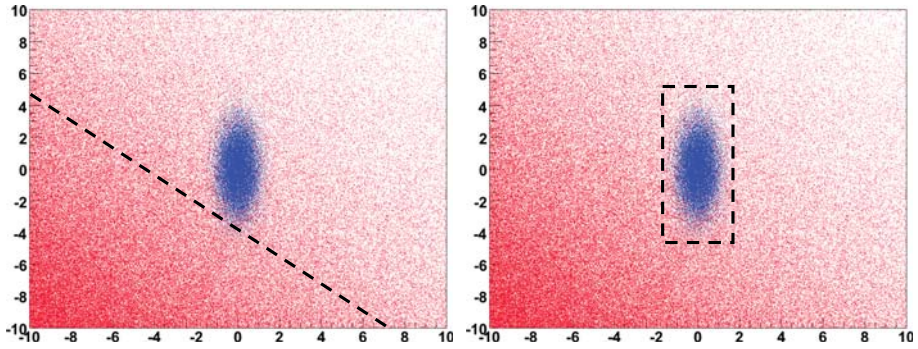


Figure 9.3: Examples of two-dimensional selections of a signal (blue dots) against a background (red dots). A linear cut is chosen on the left plot, while a box cut is chosen on the right plot.

9.2.1 The Neyman–Pearson lemma

In order to optimize the performances of a selection one has to achieve a large selection efficiency corresponding to a small misidentification probability. For a fixed signal efficiency, $\varepsilon = 1 - \alpha$, the Neyman–Pearson lemma[1] allows to determine a selection which has the lowest possible misidentification probability β based on the ratio of the likelihood functions of the observed data sample $\vec{x}$ determined under the two hypotheses H_1 and H_0. The adopted test statistics is defined as:

$$\lambda(\vec{x}) = \frac{L(\vec{x}|H_1)}{L(\vec{x}|H_0)}\,. \tag{9.2}$$

The signal selection requirement based on λ is:

$$\lambda(\vec{x}) \geq k_\alpha\,, \tag{9.3}$$

where k_α is a constant which can be determined given a fixed value of α.

If the k variables $x_1, \cdots, x_k$ that characterize our problem are independent, the likelihood function can be written as the product of one-dimensional PDFs:

$$\lambda(x_1, \cdots, x_k) = \frac{L(x_1, \cdots, x_k|H_1)}{L(x_1, \cdots, x_k|H_0)} = \frac{\prod_{j=1}^{k} f_j(x_j|H_1)}{\prod_{j=1}^{k} f_j(x_i|H_0)}\,. \tag{9.4}$$

This allows in many cases to simplify the computation of the likelihood ratio and to easily obtain the optimal selection. In concrete examples it is not always easy to find the exact functional form of λ. Numerical methods and algorithms exist to find selections in the variable space that have performances in terms of efficiency and misidentification probability close to the optimal limit given by the Neyman–Pearson lemma. There are cases in which those algorithms achieve great complexity. Among such methods some of the most frequently used in High Energy Physics are Artificial Neural Networks and Boosted Decision Trees, which are treated in this series of lectures.

In case we have a sample consisting of n events, each determined from the observation of the k variables $x_1, \cdots, x_k$, the likelihood function corresponding to the entire sample can be written as the product of PDFs evaluated at the observed variables $\vec{x}_i$, $i = 1, \cdots, n$, for each event:

$$L = \prod_{i=1}^{n} f(\vec{x}_i; \vec{\theta})\,. \tag{9.5}$$

Above, the hypotheses H_1 and H_0 are represented as two possible sets of values of the parameters $\vec{\theta} = (\theta_1, \cdots, \theta_m)$ that characterize the PDFs. Usually we want to use the number of events n as information in the likelihood definition, hence we use the *extended likelihood function* defined as the product of the usual likelihood function and a Poissonian probability corresponding to the observed number of events N:

$$L = \frac{e^{-\nu(\vec{\theta})}\nu(\vec{\theta})^n}{n!} \prod_{i=1}^{n} f(\vec{x}_i; \vec{\theta}) \,. \tag{9.6}$$

In the Poissonian term the expected number of event ν may also depend on the parameters $\vec{\theta}$: $\nu = \nu(\vec{\theta})$. Typically, we want to discriminate between two hypotheses, which are the presence of only background events in our sample ($\nu = b$) or the presence of both signal and background are present ($\nu = s + b$). The *signal strength* is usually introduced to measure the ratio of the signal yield to its theoretical prediction:

$$\nu = \mu s + b \,. \tag{9.7}$$

The hypothesis H_0 corresponding to the presence of background only is equivalent to $\mu = 0$, while the hypothesis H_1 corresponding to the presence of background plus signal is equivalent to $\mu = 1$. The PDF $f(\vec{x}_i; \vec{\theta})$ can be written as superposition of two components, one PDF for signal and another for background, weighted by the expected signal and background fractions, respectively:

$$f(\vec{x}; \vec{\theta}) = \frac{\mu s}{\mu s + b} f_s(\vec{x}; \vec{\theta}) + \frac{b}{\mu s + b} f_b(\vec{x}; \vec{\theta}) \,. \tag{9.8}$$

In this case the extended likelihood function, Eq. 9.6 becomes:

$$L = \frac{e^{-(\mu s(\vec{\theta}) + b(\vec{\theta})}}{n!} \prod_{i=1}^{n} \left(\mu s f_s(\vec{x}_i; \vec{\theta}) + b f_b(\vec{x}_i; \vec{\theta}) \right) \,. \tag{9.9}$$

The term $1/n!$ disappears when performing the likelihood ratio in Eq. (9.2).

9.2.2 Wilks' theorem

In the case of a large number of events, it is useful to have an approximation of the likelihood ratio defined in Eq. 9.2. Using Wilks' theorem [2], assuming some regularity conditions of the likelihood function, the quantity:

$$\chi^2_r = -2 \ln \frac{L(\vec{x}; \hat{\vec{\theta}}_1 | H_1)}{L(\vec{x}; \hat{\vec{\theta}}_0 | H_0)} \,, \tag{9.10}$$

where the parameter values $\hat{\vec{\theta}}_0$ and $\hat{\vec{\theta}}_1$ are taken as the *maximum likelihood estimates* of $\vec{\theta}$ corresponding to the observed data sample $\vec{x}$ in the two hypotheses H_0 and H_1 respectively, can be asymptotically approximated with a χ^2 distribution having a number of degrees of freedom equal to the difference between the number of free parameters (i.e.: not constrained from the fit) in $L(\vec{x}|H_1)$ and $L(\vec{x}|H_0)$ [3].

9.3 Claiming a discovery: significance

Given an observed data sample, claiming a discovery of a new signal requires to determine that the sample is sufficiently *inconsistent* with the hypothesis that only background is present. A test statistics can be used to measure how consistent or inconsistent the observation is with the hypothesis $\mu = 0$.

A quantitative measurement of the inconsistency with the background-only hypothesis is given by the *significance*, defined from the probability p (p-value) that the considered test statistics t assumes a value greater or equal to the observed one (large values of t corresponds to more signal-like sample) in the case of pure background fluctuation. The p-value has a uniform distribution between 0 and 1 for the background-only hypothesis, and tends to have small values in the presence of a signal. The distribution is more peaked towards zero in the presence of signal as there is better separation between signal and

background.

Instead of quoting the p-value, publications often preferred to quote the equivalent number of standard deviation that correspond to an area p under a, extreme tail of a normal distribution. So, one quotes a "$Z\sigma$" significance corresponding to a given p-value by using the following transformation:

$$p = \int_Z^\infty \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \mathrm{d}x = 1 - \frac{1}{2}\mathrm{erf}\left(\frac{Z}{\sqrt{2}}\right) . \tag{9.11}$$

By convention in literature one claims the *"observation"* of the signal under investigation if the observed significance is at least 3σ ($Z = 3$), which corresponds to a probability of background fluctuation of 1.35×10^{-3}, while one claims the *"evidence of"* the signal (*discovery!*) in case the significance is at least 5σ ($Z = 5$), corresponding to a p-value of 2.87×10^{-7}. Table 9.1 shows a number of typical significance values expressed as '$Z\sigma$" and their corresponding p-values.

Table 9.1: Significances expressed as '$Z\sigma$" and corresponding p-values in a number of typical cases.

Z (σ)	p
1.00	1.59×10^{-1}
1.28	1.00×10^{-1}
1.64	5.00×10^{-2}
2.00	2.28×10^{-2}
2.32	1.00×10^{-2}
3.00	1.35×10^{-3}
3.09	1.00×10^{-3}
3.71	1.00×10^{-4}
4.00	3.17×10^{-5}
5.00	2.87×10^{-7}
6.00	9.87×10^{-10}

Determining the significance, anyway, is only part of the process that leads to a discovery, in the scientific method. Quoting from Ref. [4]:

"It should be emphasized that in an actual scientific context, rejecting the background-only hypothesis in a statistical sense is only part of discovering a new phenomenon. One's degree of belief that a new process is present will depend in general on other factors as well, such as the plausibility of the new signal hypothesis and the degree to which it can describe the data. Here, however, we only consider the task of determining the p-value of the background-only hypothesis; if it is found below a specified threshold, we regard this as "discovery"."

In order to evaluate the "plausibility of a new signal" and other factors that give confidence in a discovery requires a judgement that cannot, of course, be replaced by the satistical evaluation only.

9.4 Excluding a signal hypothesis

For the purpose of excluding a signal hypothesis, usually the requirement applied in terms of p-value is much milder than for a discovery. Instead of the requiring a p-value of 2.87×10^{-7} or less (5σ), the upper limits for an exclusion are set requiring $p < 0.05$, corresponding to a 95% confidence level (CL) or $p < 0.10$, corresponding to a 90% CL. In this case, p indicates the probability of a signal *underfluctuation*, i.e.: the null hypothesis and alternative hypothesis are inverted with respect to the case of a discovery.

9.5 Definitions of upper limits

In the frequentist approach the procedure to set an upper limit is similar to the determination of a confidence interval for the unknown signal yield s. In the case one wants to determine au upper limit instead of a central interval, the choice of the interval with the desired CL (90% or 95%, usually) may

be fully asymmetric, becoming $s \in [0, s^{\rm up}[$. When the outcome of an experiment is an upper limit, one usually quotes:

$$s < s^{\rm up} \ \text{ at } \ 95\%\,\text{C.L (or } 90\%\,\text{CL)}\,.$$

If the Bayesian approach is adopted, the interpretation of an upper limit $s^{\rm up}$ is very different. The interval $s \in [0, s^{\rm up}[$ has to be interpreted as *credible interval*, meaning that its corresponding *posterior probability* is equal to the CL $1-\alpha$.

9.6 Poissonian counting experiments

A simple though realistic case is a counting experiment where selected events contain a mixture of signal and background events. The total number of observed events will be on average $s+b$ where s and b are the expected number of signal and background events, respectively. The main unknown parameter is s, which could also be equal to zero in case the signal is not present. The likelihood function in the case of a counting experiment is:

$$L(n; s, b) = \frac{(s+b)^n}{n!} e^{-(s+b)}\,, \tag{9.12}$$

where n is the observed number of events.

9.7 Bayesian approach

The easiest treatment of a counting experiment, at least from the technical point of view, can be done under the Bayesian approach. Assuming a uniform prior PDF for s, the Bayesian posterior PDF for s is given by:

$$P(s|n) = \frac{L(n; s)}{\int_0^\infty L(n; s)\mathrm{d}s}\,. \tag{9.13}$$

The upper limit $s^{\rm up}$ can be computed requiring that the posterior probability corresponding to the interval $[0, s^{\rm up}[$ is equal to CL, or equivalently that the probability corresponding to $[s^{\rm up}, \infty[$ is $\alpha = 1 - \text{CL}$:

$$\alpha = 1 - \text{CL} = \int_{s^{\rm up}}^\infty P(s|n)\mathrm{d}s = \frac{\int_{s^{\rm up}}^\infty L(n; s)\mathrm{d}s}{\int_0^\infty L(n; s)\mathrm{d}s}\,. \tag{9.14}$$

In the simplest case of negligible background, $b = 0$, the posterior PDF for s can be demonstrated to have the same expression as the Poissonian probability itself:

$$P(s|n) = \frac{s^n e^{-s}}{n!}\,. \tag{9.15}$$

In the case of no observed events, $n = 0$, we have:

$$P(s|0) = e^{-s}\,, \tag{9.16}$$

and:

$$\alpha = 1 - \text{CL} = \int_{s^{\rm up}}^\infty e^{-s}\mathrm{d}s = e^{-s}\,. \tag{9.17}$$

Hence, we can set the following upper limits:

$$s^{\rm up} = 3.00 \ \text{ at } \ 95\%\text{CL}\,, \tag{9.18}$$

$$s^{\rm up} = 2.30 \ \text{ at } \ 90\%\text{CL}\,. \tag{9.19}$$

The general case of a possible expected background $b \neq 0$ was treated by O. Helene [5], and Eq. 9.14

becomes:

$$\alpha = e^{-s^{\rm up}} \frac{\sum_{m=0}^{n} \frac{(s^{\rm up}+b)^m}{m!}}{\sum_{m=0}^{n} \frac{b^m}{m!}} \,. \tag{9.20}$$

The above expression can be inverted numerically to determine $s^{\rm up}$ for given α, n and b. In the case of no background ($b = 0$) Eq. (9.20) becomes:

$$\alpha = e^{-s} \sum_{m=0}^{n} \frac{s^m}{m!} \,, \tag{9.21}$$

and the corresponding upper limits are reported in Tab. 9.2.

Table 9.2: Upper limits in presence of negligible background.

	$1-\alpha = 90\%$	$1-\alpha = 95\%$
n	$s^{\rm up}$	$s^{\rm up}$
0	2.30	3.00
1	3.89	4.74
2	5.32	6.30
3	6.68	7.75
4	7.99	9.15
5	9.27	10.51
6	10.53	11.84
7	11.77	13.15
8	12.99	14.43
9	14.21	15.71
10	15.41	19.96

For different number of observed events n and different expected background b, the upper limits derived in [5] are shown in Fig. 9.4.

9.7.1 Limitations of the Bayesian approach

The derivation of Bayiesian upper limits done above assumes a uniform prior on the expected signal yield s. Assuming a different prior distribution would result in different upper limits. In general, there is no univoque criterion to chose a specific prior PDF to model the complete lack of knowledge about a variable, like in this case the signal yield. This *subjectiveness* in the choice of the prior PDF is intrinsic in the Bayesian approach, and raises criticism by supporters of the frequentist approach, which object that the obtained Bayesian results are to some extent *subjective*. Supporters of the Bayesian approach reply that the obtained result are *intersubjective* [6], in the sense that common prior choices lead to common results, and some debates are still ongoing in literature.

A frequently adopted prior distribution in physics that models one's complete lack of knowledge of a parameter is to assume a uniform distribution, as it was done for the simple Poissonian example above. This approach is anyway not unique: should we define a prior uniform in s or in $\ln s$? A typical case is the measurement of a particle lifetime τ or, alternatively, its width $\Gamma \sim 1/\tau$. Since there is no natural choice between the two quantities, should we assume a uniform prior in τ or in $1/\tau$?

An approach to find a prior distribution that is *invariant* under reparametrization of our PDF is due to H. Jeffreys [7] who suggested to chose the prior to be proportional to the square root of the determinant of the Fisher information matrix:

$$p(\vec{\theta}) \propto \sqrt{I(\vec{\theta})} \,, \tag{9.22}$$

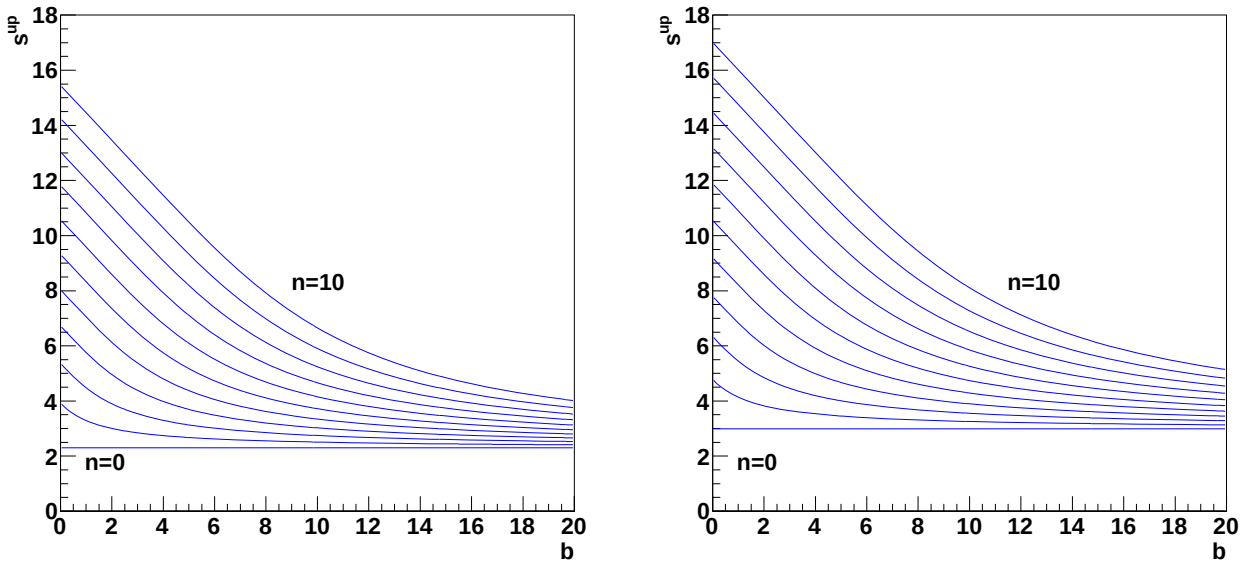


Figure 9.4: Upper limits at the 90% CL (left) and 95% CL (right) for Poissonian process using the Bayesian approach as a function of the expected background b and for number of observed events n from $n = 0$ to $n = 10$.

where

$$I(\vec{\theta}) = \det\left[\left\langle \frac{\partial \ln L(\vec{x};\vec{\theta})}{\partial\theta_i}\frac{\partial \ln L(\vec{x};\vec{\theta})}{\partial\theta_j}\right\rangle\right] . \tag{9.23}$$

Using Jeffreys' approach leads to prior PDF that are usually not uniform. Table 9.3 shows a number of typical cases. For instance, for a Poissonian counting experiment Jeffreys' prior is proportional to $1/\sqrt{s}$,

Table 9.3: Jeffreys priors for a number of typical PDFs.

PDF parameter	Jeffreys prios
Poissonian mean	$p(s) \propto 1/\sqrt{s}$
Poissonian mean with background	$p(s) \propto 1/\sqrt{s+b}$
Gaussian mean	$p(\mu) \propto 1$
Gaussian r.m.s.	$p(\sigma) \propto 1/\sigma$
Binomial parameter	$p(\varepsilon) \propto 1/\sqrt{\varepsilon(1-\varepsilon)}$

not uniform as assumed to determine Eq. 9.15.

9.8 Frequentist limits: a simple case

In case we observe $n = 0$ events we can state that the number of observed signal events is $n_s = 0$, and the number of observed background events is $n_b = 0$. If we also assume for simplicity that the expected background is nebligible, we can set $b \simeq 0$, hence we will have for any observed number of events n that $n_b = 0$ and $n_s = n$. The probability to observe n events when we expect s events is given by Poissonian distribution:

$$p = P(n;s) = \frac{e^{-s}s^n}{n!} . \tag{9.24}$$

For $n = 0$ we have:

$$p = P(0;s) = e^{-s} . \tag{9.25}$$

We can set an upepr limit on the expected signal yield s *excluding* values of s for which $p < \alpha = 1 - \mathrm{CL}$. Hence, we allow signal values s that satisfy:

$$p = s^{-s} \geq \alpha = 1 - \mathrm{CL}\,. \tag{9.26}$$

The above relation can be inverted, and gives:

$$s \leq -\ln \alpha = s^{\mathrm{up}}\,, \tag{9.27}$$

which, for $\alpha = 5\%$ or $\alpha = 10\%$ gives:

$$s \leq 3.00 \text{ at } 95\%\,\mathrm{CL}, \tag{9.28}$$

$$s \leq 2.30 \text{ at } 90\%\,\mathrm{CL}. \tag{9.29}$$

$$\tag{9.30}$$

Those results coincide accidentally with the results obtained under the Bayesian approach and shown Table 9.2. The coincidence of limits under the Bayesian and frequentist approaches, like in this case, may lead to confusion. There is no intrinsic reason for which limits evaluated under the two approaches should coincide, and in general, with very few exceptions, like in this case, Bayesian and frequentist limits don't coincide.

9.9 Steps towards a frequentist approach

An effort to conciliate Bayesian frequentist limits obtained by Helene in Ref. [5] and the frequentist approach was attempted by G. Zech [8]. In order to determine the probability distribution of the number of events from the sum of two Poissonian processes with s and b expected number of events from signal and background, respectively, one can write the probability distribution for the total observed number of events n as:

$$P(n; s, b) = \sum_{n_b=0}^{n} \sum_{n_s=0}^{n-n_b} P(n_b; b) P(n_s; s)\,, \tag{9.31}$$

where $P(n_b; b)$ and $P(n_s; s)$ are Poissonian probability distribution. It's easy to demonstrate that $P(n; s, b)$ is again a Poissonian distribution with average $s + b$. Zech proposed to modify the first term, $P(n_b; b)$, to take into account the observation of n events that would limit the possible values of n_b from 0 to n. In this way, one would replace $P(n_b; b)$ with

$$P'(n_b; b) = P(n_b; b) / \sum_{n'_b=0}^{n} P(n'_b; b)\,. \tag{9.32}$$

This modification leads to the same result obtained by Helene in Eq. (9.20). Though the approach was later criticized [9] because it led to uncorrect coverage, and Zech himself admitted the non rigorous application of the frequentist approach, his intuition anticipates the formulation of the *modified frequentist approach* that will be discussed later on in Section 9.14.

9.10 Frequentist approach: Neyman's confidence intervals

A rigorous and general frequentist treatment of confidence intervals is due to J. Neyman [11]. Let's consider a variable x distributed according to a PDF which depends on an unknown parameter θ. Neyman's procedure to determine confidence intervals proceeds in two steps. First, a *confidence belt* is determined scanning the parameter space by varying θ within its allowed range. For each fixed value $\theta = \theta_0$ we know the corresponding PDF which describes the distribution of x, $f(x|\theta_0)$. According to the PDF $f(x|\theta_0)$ a confidence interval $[x_1(\theta_0), x_2(\theta_0)]$ is determined whose corresponding probability is equal to the specified

CL$= 1-\alpha$, usually equal to 68.27%, 90% or 95%:

$$1-\alpha = \int_{x_1(\theta_0)}^{x_2(\theta_0)} f(x|\theta_0)\mathrm{d}x\,. \tag{9.33}$$

Neyman's construction is graphically illustrated in Fig. 9.5, left. The choice of $x_1(\theta_0)$ and $x_2(\theta_0)$ has still

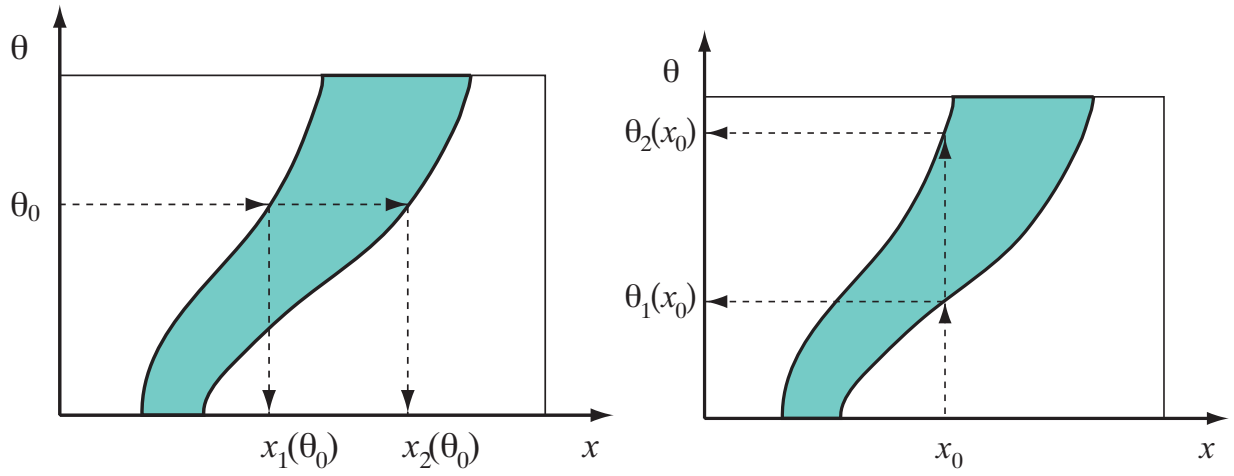


Figure 9.5: Graphical illustration of Neyman's belt construction (left) and inversion (right).

some arbitrariness, since there are different possible intervals having the same probability, according to Eq. (9.33). The choice of this interval is referred to in litterature as *ordering rule*. For instance, one can chose an interval centered around the average value of x given θ_0, i.e.: an interval:

$$[x_1(\theta_0), x_2(\theta_0)] = [\langle x|\theta_0\rangle - \delta, \langle x|\theta_0\rangle + \delta]\,, \tag{9.34}$$

where δ is such to ensure that Eq. (9.33) holds. Or one can chose the interval such that

$$\int_{-\infty}^{x_1(\theta_0)} f(x|\theta_0)\mathrm{d}x = \frac{\alpha}{2} \quad \text{and} \quad \int_{x_2(\theta_0)}^{+\infty} f(x|\theta_0)\mathrm{d}x = \frac{\alpha}{2}\,. \tag{9.35}$$

One can also chose one of the two possible fully asymmetric intervals: $[x_1(\theta_0), +\infty]$ or $[-\infty, x_2(\theta_0)]$. Fig. 9.6 shows the three possible cases described above. Other possibilities are also considered in literature. A special ordering rule introduced by Feldman and Cousins based on a likelihood ratio criterion will be discussed in Section 9.12. Given a choice of the ordering rule, the intervals $[x_1(\theta), x_2(\theta)]$, for all possible values of θ, define the Neyman belt in the space (x, θ) as shown in Fig. 9.5.

As second step of the Neyman procedure, given a measurement $x = x_0$, the confidence interval for θ is determined inverting the Neyman belt (Fig. 9.5, right): two extreme values $\theta_1(x_0)$ and $\theta_2(x_0)$ are determined as the intersections of the vertical line $x = x_0$ with the two boundary curves of the belt, i.e. we find the values $\theta = \theta_1(x_0)$ and $\theta = \theta_2(x_0)$. The interval $[\theta_1(x_0), \theta_2(x_0)]$ has, by construction, a *coverage* equal to the confidence level $1-\alpha$. This means that, if θ is equal to a true value θ_0, extracting $x = x_0$ randomly according to the PDF $f(x|\theta_0)$, θ_0 will be included in the determined confidence interval, $[\theta_1(x_0), \theta_2(x_0)]$ in a fraction $1-\alpha$ of the cases, in the limit of a large number of extractions.

Upper or lower limits on θ can be determined using fully asymmetric intervals for x. In particular, assuming that the Neyman belt is monotonically increasing, the choice of intervals $[x_1(\theta_0), +\infty[$ leads to a confidence interval $[0, \theta(x_0)]$ for θ which corresponds to a un upper limit $\theta^{\mathrm{up}} = \theta(x_0)$. This case is illustrated in Fig. 9.7.

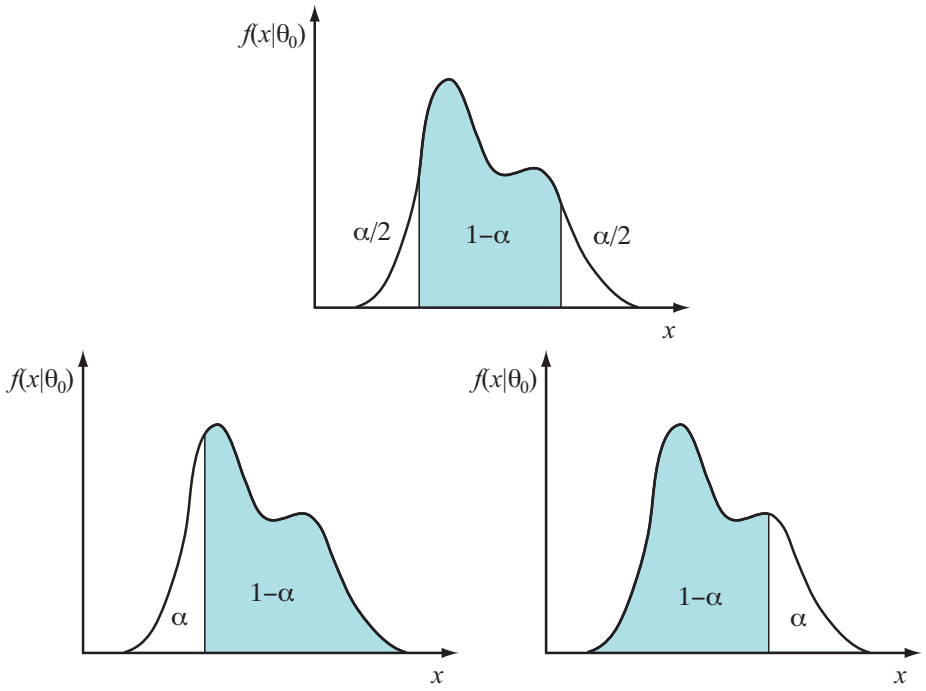


Figure 9.6: Three possible choices of ordering rule: central interval (top) and fully asymmetric intervals (bottom left, right).

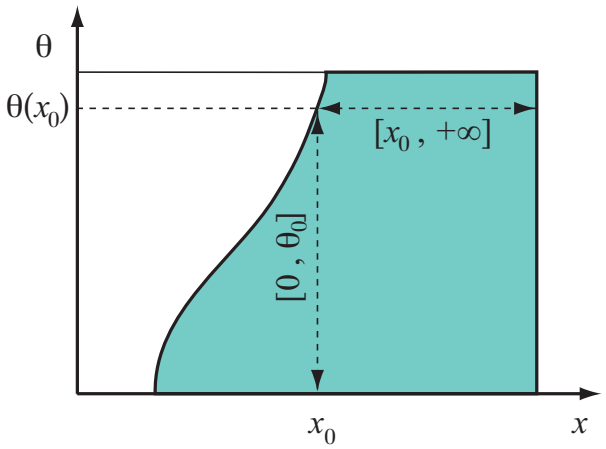


Figure 9.7: Graphical illustration of Neyman's belt construction for upper limits determination.

9.11 The "flip-flopping" problem

In order to determine confidence intervals, a consistent choice of ordering rule has to be adopted. Feldman and Cousins demonstrated [12] that the ordering rule choice must not depend on the outcome of the measurements, otherwise the quoted confidence intervals or upper limits could correspond to incorrect confidence level (i.e.: coverage). In some cases, experiment searching for a rare signal make the chose, while quoting their result, to switch from a central interval to an upper limit depending on the outcome of the measurement. A typical choice is to quote an upper limit if the significance of the observed signal is smaller than 3σ, and a central value otherwise. This problem is sometimes referred to in literature as *flip-flopping*, and can be illustrated in a simple example. Imagine a model where a random variable x obeys a Gaussian distribution with a fixed and known r.m.s., for simplicity we can take $\sigma = 1$, and an unknown average μ which is bound to be greater or equal to zero (this is the case of a signal yield). The quoted central value must always be greater than or equal to zero, given the assumed constraint. Assume we decide to quote zero if the significance is less than 3σ:

$$\mu = \begin{cases} x & \text{if} \quad x/\sigma \geq 3 \\ 0 & \text{if} \quad x/\sigma < 3 \end{cases} , \tag{9.36}$$

From a single measurement of x we could decide to quote a central value if $x/\sigma \geq 3$ with a symmetric

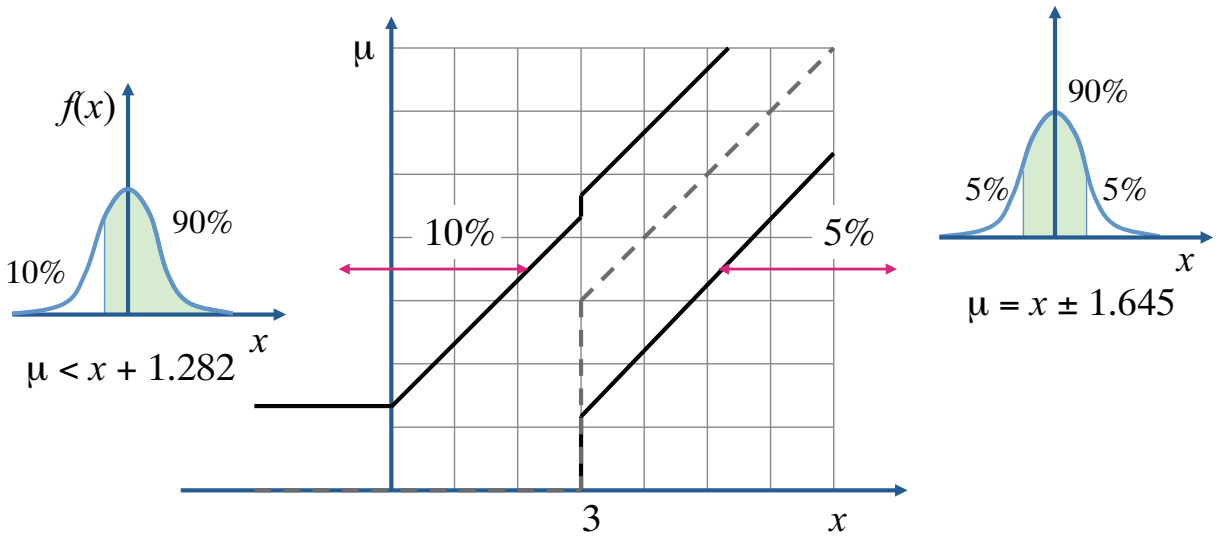


Figure 9.8: Illustration of the *flip-flopping* problem. The plot shows the quoted central value of μ as a function of the measured x (dashed line), and the 90% confidence interval corresponding to a choice to quote a central interval for $x/\sigma \geq 3$ and an upper limit for $x/\sigma < 3$.

error: $\pm\sigma$ at the 68.27% CL, or $\pm 1.645\sigma$ at 90% CL. Instead, we may decide to quote an upper limit if $x/\sigma < 3$. The upper limit to μ can be derived using a fully asymmetric interval, and corresponds to $\mu < x + 1.282$ at 90% CL. The quoted confidence interval at 90% CL becomes:

$$[\mu_1, \mu_2] = \begin{cases} [x - 1.645, x + 1.645] & \text{if} \quad x/\sigma \geq 3 \\ [0, x + 1.282] & \text{if} \quad x/\sigma < 3 \end{cases} , \tag{9.37}$$

The situation is shown in Fig. 9.8.

The choice to switch from a central interval to a fully asymmetric interval (upper limit) based on the observation of x clearly spoils the statistical coverage. Looking at Fig. 9.8, depending on the value of μ, the interval $[x_1, x_2]$ obtained by crossing the confidence belt in by an horizontal line, one may have cases where the coverage decreases from 90% to 85%, which is lower than the desired CL. Next Section 9.12 presents the method due to Feldman and Cousins to approach consistently the coverage problem without incurring the flip-flopping problem.

9.12 The unified Feldman-Cousins approach

The ordering rule proposed by Feldman and Cousins [12] provides a Neyman confidence belt, as defined in Section 9.10, that smoothly changes from a central or quasi-central interval to an upper limit in the case of low observed signal yield. The ordering rule is based on the likelihood ratio introduced in Section 9.2.1: given a value θ_0 of the unknown parameter under a Neyman construction, the chosen interval on the variable x is defined from the ratio of two PDFs of x, one under the hypothesis that θ is equal to the considered fixed value θ_0, the other under the hypothesis that θ is equal to the maximum-likelihood estimate value $\theta_{\rm best}(x)$ corresponding to the given measurement x. The likelihood ratio must be greater than a constant k_α whose value depends on the chosen confidence level $1-\alpha$:

$$\lambda(x|\theta_0) = \frac{f(x|\theta_0)}{f(x|\theta_{\rm best})} > k_\alpha \,. \tag{9.38}$$

The confidence interval R_α for a given value θ_0 is given by:

$$R_\alpha(\theta_0) = \{x : \lambda(x|\theta_0) > k_\alpha\} \,, \tag{9.39}$$

and the constant k_α is chosen in such a way that:

$$\int_{R_\alpha} f(x|\theta_0)\mathrm{d}x = 1-\alpha \,. \tag{9.40}$$

This case is illustrated in Fig. 9.9.

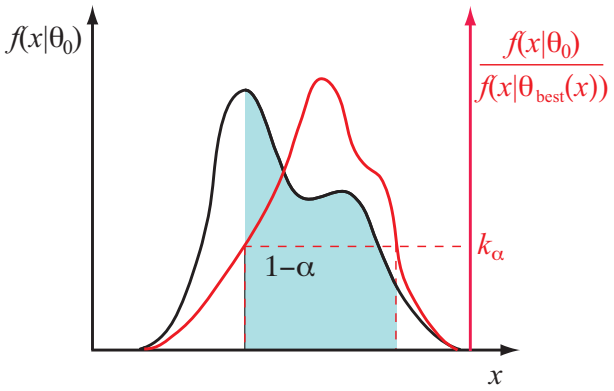


Figure 9.9: Ordering rule in the Feldman–Cousins approach, based on the likelihood ratio.

Feldman and Cousins computed the confidence interval for the simple Gaussian case discussed in Section 9.11. The maximum-likelihood value for μ, given x and under the constraint $\mu \geq 0$, is:

$$\mu_{\rm best} = \max(x, 0) \,. \tag{9.41}$$

The PDF for x using the maximum-likelihood estimate for μ becomes:

$$f(x|\mu_{\rm best}) = \begin{cases} \frac{1}{\sqrt{2\pi}} & \text{if} \quad x \geq 0 \\ \frac{1}{\sqrt{2\pi}} e^{-x^2/2} & \text{if} \quad x < 0 \end{cases} \,. \tag{9.42}$$

The likelihood ratio in Eq. 9.38 can be written for this case as:

$$\lambda(x|\mu) = \frac{f(x|\mu)}{f(x|\mu_{\rm best})} = \begin{cases} \exp(-(x-\mu)^2/2 & \text{if} \quad x \geq 0 \\ \exp(x\mu - \mu^2/2) & \text{if} \quad x < 0 \end{cases} \,. \tag{9.43}$$

The interval $[x_1(\mu_0), x_2(\mu_0)]$, for a given $\mu = \mu_0$, can be found numerically using the equation $\lambda(x|\mu) > k_\alpha$ and imposing the normalization from Eq. 9.40, given the desired value of α. The results are shown in

Fig. 9.10, and can be compared to Fig. 9.8. Using the Feldman–Cousins (FC) approach, for large values

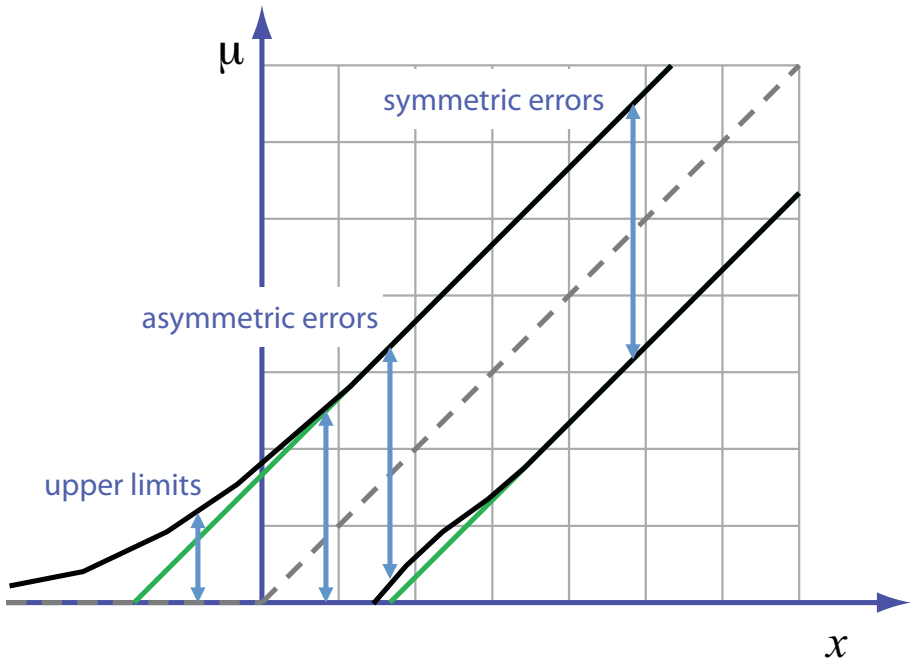


Figure 9.10: Neyman confidence belt constructed using the Feldman–Cousins ordering.

of x one gets the usual symmetric confidence interval. As x moves to lower values, the interval becomes more and more asymmetric, and at some point it is fully asymmetric, determining an upper limit. For negative values of x the result is always an upper limit avoiding unphysical values with negative values of μ. This approach smoothly changes from a central interval to an upper limit, yet ensuring the correct 90% coverage.

9.13 Frequentist upper limits on discrete variables

In the case of a discrete variable n, like for a Poissonian counting experiment, it's not always possible to find an interval $\{n_1, \ldots, n_k\}$ that has the exact coverage. In such cases, one as to take the smallest interval having a probability greater or equal to the desired CL. In this way the determined limit is *conservative*, i.e. the procedure ensures that the probability that the true value s lies within the determined confidence interval $[s_1, s_2]$ is *greater or equal* to CL$= 1 - \alpha$ (*overcoverage*). Fig. 9.11 shows an example of Poissonian distribution corresponding to the case with $s = 4$ and $b = 0$. Using a fully asymmetric interval as ordering rule, the interval $\{2, 3, \cdots\}$ of the discrete variable n corresponds to a probability $P(n \geq 2) = 1 - P(0) - P(1) = 0.9084$, and is the smallest interval which has a probability gretaer or equal to a desired CL of 0.90. Given an observation of n events, we could set the upper limit s^{up} such that:

$$s^{\text{up}} = \min_{\sum_{m=0}^{n} P(m;s)<\alpha} (s)\,. \tag{9.44}$$

In the simplest case where $n = 0$, we have:

$$s^{\text{up}} = \min_{P(0;s)<\alpha} (s) = \min_{e^{-s}<\alpha} (s) = -\ln\alpha\,. \tag{9.45}$$

Since α, is equal to one minus the CL, we have:

$$s^{\text{up}} = -\ln(1 - \text{CL})\,. \tag{9.46}$$

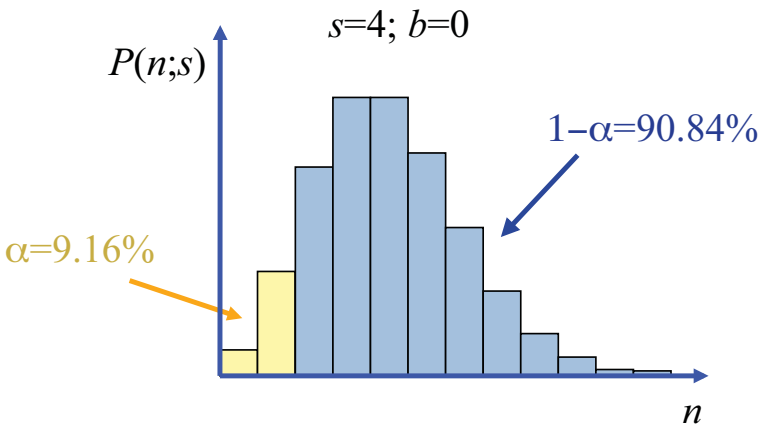


Figure 9.11: Poissonian distribution in the case of a signal $s = 4$ and $b = 0$. The dark bins show the smallest possible fully asymmetric confidence interval that gives at least the coverage of $1 - \alpha = 90\%$.

For values of $\alpha = 0.05$ (95% CL) or $\alpha = 0.1$ (90% CL), we have again the results derived in Sec. 9.8:

$$s^{\rm up} = 2.00 \text{ at } 95\%\text{CL}, \tag{9.47}$$

$$s^{\rm up} = 2.30 \text{ at } 90\%\text{CL}. \tag{9.48}$$

From the purely frequentist point of view, anyway, this result suffers from the flip-flopping problem discussed in Section 9.11: the choice to switch from fully asymmetric to central interval according to the observed result leads to an incorrect coverage, which can be fixed adopting the FC approach. In the Poissonian case, the 90% confidence belt obtained with the FC approach is shown in Fig. 9.12. The

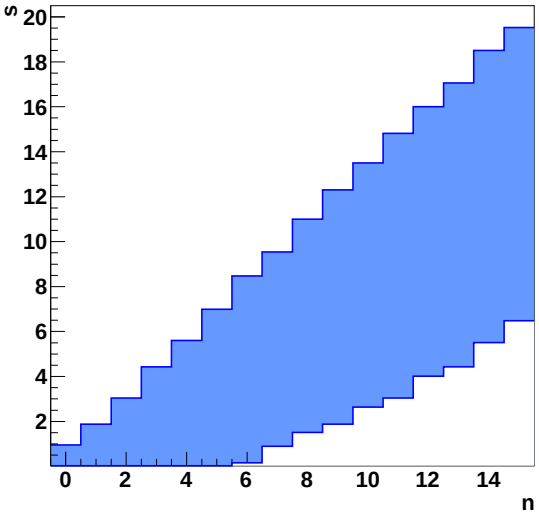


Figure 9.12: 90% confidence belt for a Poissonian process using Feldman–Cousins ordering, in the case of $b = 3$.

results in the case of no background ($b = 0$) are reported in Table 9.4. For different numbers of observed events n and different expected background b, the upper limits derived using the FC method are shown in Fig. 9.13. Comparing Table 9.4 with Table 9.2, which contains the Bayesian results, FC upper limits are in general larger, i.e.: less stringent, than Bayesian limits. In particular, for the case of $n = 0$, the upper limit increases from 2.30 to 2.44 for a 90% CL and from 3.00 to 3.09 for a 95% CL. But, as remarked before, the interpretation of frequentist and Bayesian limits is very different.

	$1-\alpha = 90\%$		$1-\alpha = 95\%$	
n	s^{lo}	s^{up}	s^{up}	s^{lo}
0	0.00	2.44	0.00	3.09
1	0.11	4.36	0.05	5.14
2	0.53	5.91	0.36	6.72
3	1.10	7.42	0.82	8.25
4	1.47	8.60	1.37	9.76
5	1.84	9.99	1.84	11.26
6	2.21	11.47	2.21	12.75
7	3.56	12.53	2.58	13.81
8	3.96	13.99	2.94	15.29
9	4.36	15.30	4.36	16.77
10	5.50	16.50	4.75	17.82

Table 9.4: Upper and lower limits in presence of negligible background ($b = 0$) obtained using the Feldman–Cousins approach.

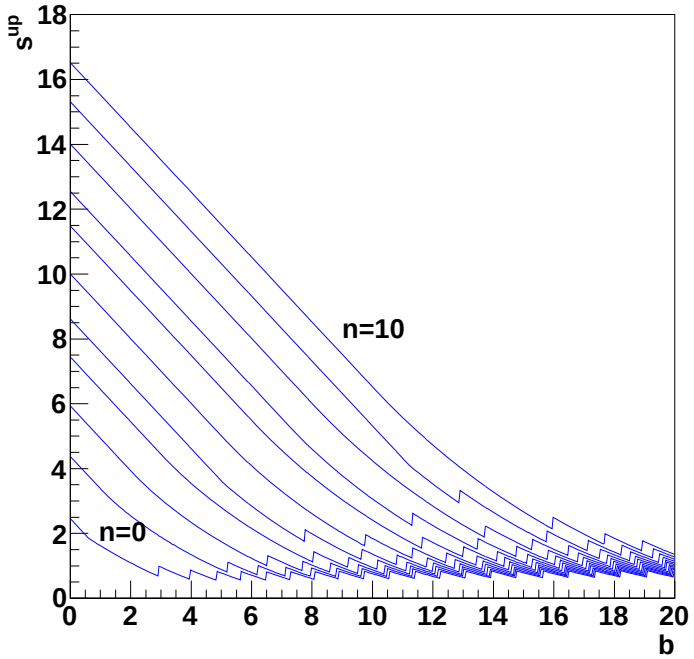


Figure 9.13: Upper limits at 90% confidence belt for Poissonian process using Feldman–Cousins ordering as a function of the expected background b and for number of observed events n from 0 to 10.

A peculiar feature of FC upper limits is that, for $n = 0$, a larger expected background b corresponds to a more stringent, i.e.: lower, upper limit, differently from what happens to Bayesian limits that do not depend on the expected background for $n = 0$. This dependence on the expected amount of background is somewhat counterintuitive: imagine two experiments performing a search for a rare signal designed to achieve a low background level. If both measure zero counts, the experiment that achieves the most stringent limit is the one which has the highest expected background level!

The Particle Data Group published in their review [13] the following sentence about the interpretation of frequentist upper limits, in particular concerning the difficulty to interpret a more stringent limit if the expected background increases for the $n = 0$ case:

"The intervals constructed according to the unified [Feldman Cousins] procedure for a Poisson variable n consisting of signal and background have the property that for $n = 0$ observed events, the upper limit decreases for increasing expected background. This is counter-intuitive, since it is known that if $n = 0$ for the experiment in question, then no background was observed, and therefore one may argue that the expected background should not be relevant. The extent to which one should regard this feature as a drawback is a subject of some controversy".

This feature of frequentist limits is often considered unpleasant by physicists. The need to come to an agreed procedure to determine upper limits, mainly triggered by the need to combine the results of the four LEP experiments on Higgs boson search, lead to the proposal of a new method that modifies the purely frequentist approach, as will be discussed in the following section.

9.14 Modified frequentist approach: the CL_s method

The concerns about frequentist limits discussed at the end of the previous section have been addressed in the definition of a new procedure that was adopted for the first time in the combination of the results of the search for the Higgs boson [14] of the four LEP experiments, Aleph, Delphi, Opal and L3. The modification of the purely frequentist confidence level by a conservative correction factor can cure, as will be presented in the following, the counterintuitive peculiarities of the frequentist limit procedure.

The original proposal of the *modified frequentist approach* adopted a test statistics based on the ratio of the likelihood functions evaluated under two different hypotheses: the presence of signal plus background, and the presence of background only:

$$\lambda = \frac{L_{s+b}}{L_b}\,. \tag{9.49}$$

Different test statistics have been applied after the original definition of the LEP procedure, but the remaining part of the method described in the following has been adopted mainly unchanged on the different kinds of test statistics. In the case of a simple event counting, assuming that the expected signal and background yields depend on the unknown parameters $\vec{\theta} = (\theta_1, \cdots, \theta_m)$, the likelihood function only depends on the number of observed event n, and the likelihood ratio is:

$$\lambda(\vec{\theta}) = \frac{L_{s+b}(n|\vec{\theta})}{L_b(n|\vec{\theta})}\,, \tag{9.50}$$

where L_{s+b} and L_b are Poissonian probabilities whose expected average are $s + b$ and b respectively, and the signal and background yields s and b depend on $\vec{\theta}$. More explicitly, we can write:

$$\lambda(\vec{\theta}) = \frac{e^{-(\mu s(\vec{\theta})+b(\vec{\theta}))}\left(\mu s(\vec{\theta}) + b(\vec{\theta})\right)^n}{n!}\frac{n!}{e^{-b(\vec{\theta})}b(\vec{\theta})^n} = e^{-\mu s(\vec{\theta})}\left(\frac{\mu s(\vec{\theta})}{b(\vec{\theta})} + 1\right)^n\,. \tag{9.51}$$

Moving to the negative logarithm the above expression becomes:

$$-\ln\lambda(\vec{\theta}) = \mu s(\vec{\theta}) - n\ln\left(\frac{\mu s(\vec{\theta})}{b(\vec{\theta})} + 1\right)\,. \tag{9.52}$$

If we consider, in addition to the pure counting information n, a set of k measured variables $\vec{x} =$

$(x_1, \cdots, x_k)$ that characterize each event, can write the ratio of extended likelihood functions as:

$$\lambda(\vec{\theta}) = \frac{P\left(n \left| \mu s(\vec{\theta}) + b(\vec{\theta}) \right.\right) \prod_{i=1}^{n} f_{s+b}(\vec{x}_i|\vec{\theta})}{P\left(n \left| b(\vec{\theta}) \right.\right) \prod_{i=1}^{n} f_b(\vec{x}_i|\vec{\theta})}, \tag{9.53}$$

where $P(n|s+b)$ and $P(n|b)$ are Poissonian probabilities as in Eq 9.51, and f_{s+b} and f_b are the PDF for signal plus background and background only respectively of the variables $\vec{x}$. Explicitating the Poissonian terms and writing f_{s+b} as as the superposition of signal and background compoments, similarly to Eq. 9.8, we have:

$$\lambda(\vec{\theta}) = \frac{e^{-(\mu s(\vec{\theta})+b(\vec{\theta}))} \left(\mu s(\vec{\theta}) + b(\vec{\theta})\right)^n}{e^{-b(\vec{\theta})} b(\vec{\theta})^n} \prod_{i=1}^{n} \frac{\left(\mu s(\vec{\theta}) f_s(\vec{x}_i;\vec{\theta}) + b(\vec{\theta}) f_b(\vec{x}_i;\vec{\theta})\right)}{\left(\mu s(\vec{\theta}) + b(\vec{\theta})\right)} \frac{1}{f_b(\vec{x}_i;\vec{\theta})}, \tag{9.54}$$

where f_s is the PDF of signal event. With a bit of math, we can rewrite Eq. 9.54 as:

$$\lambda(\vec{\theta}) = e^{-\mu s(\vec{\theta})} \prod_{i=1}^{n} \left(\frac{\mu s(\vec{\theta}) f_s(\vec{x}_i;\vec{\theta})}{b(\vec{\theta}) f_b(\vec{x}_i;\vec{\theta})} + 1 \right). \tag{9.55}$$

Moving to the negative logarithm, we have:

$$-\ln \lambda(\vec{\theta}) = \mu s(\vec{\theta}) - \sum_{i=1}^{n} \ln \left(\frac{\mu s(\vec{\theta}) f_s(\vec{x}_i;\vec{\theta})}{b(\vec{\theta}) f_b(\vec{x}_i;\vec{\theta})} + 1 \right). \tag{9.56}$$

In the case of a single parameter θ (i.e.: $m = 1$), one can plot $-\ln \lambda(\theta)$ as a function of θ, and the presence of a significant minimum at $\theta = \hat{\theta}$ is an indication of the possible presence of a signal having a value of the parameter θ near $\hat{\theta}$ within some uncertainty. If the background PDF does not depend on θ (for instance, if θ is the mass of an unknown particle) $L_b(\vec{x}|\theta)$ does not depend on θ and the likelihood ratio $\lambda(\theta)$ is equal, up to a multiplicative factor, to the likelihood $L_{s+b}(\vec{x}|\theta)$. Hence, the maximum likelihood estimate of θ is $\theta = \hat{\theta}$, and the error on θ can be determined as usual in maximum likelihood estimates from the shape of $-2\ln \lambda(\theta)$ around its minimum, finding its intersection with an horizontal line at $-2\ln \lambda(\hat{\theta}) + 1$.

In order to determine the significance of the measured value of θ, if the conditions to apply Wikls' theorem [2] are valid (see Section 9.2.2), the value $2\ln \lambda(\theta)$ can be approximated by a chi-squared. Hence, its value at the minimum:

$$Z = \sqrt{2 \ln \lambda(\hat{\theta})} \tag{9.57}$$

gives an approximate estimate of the significance Z. In Section 9.17 the interpretation of significance in the case of parameter estimates from data will be further discussed, and it will be clear that the estimate of significance at a fixed value of a measured parameter may suffer from a systematic overestimate (so called: *look-elsewhere effect*).

In order to quote an upper limit using the frequentist approach, the distribution of the test statistics λ (or equivalently $-2\ln \lambda$) in the hypothesis of signal plus background $(s+b)$ has to be known, and the p-value corresponding to the observed value $\lambda = \hat{\lambda}$ has to be determined. The proposed modification to the purely frequentist approach consist of finding two p-values corresponding to the $s+b$ and b hypotheses:

$$\mathrm{CL}_{s+b}(\theta) = P_{s+b}(\lambda(\theta) \leq \hat{\lambda}), \tag{9.58}$$

$$\mathrm{CL}_{b}(\theta) = P_{b}(\lambda(\theta) \leq \hat{\lambda}). \tag{9.59}$$

From those two probabilities, the following quantity can be derived:

$$\boxed{\mathrm{CL}_s(\theta) = \frac{\mathrm{CL}_{s+b}(\theta)}{\mathrm{CL}_b(\theta)}\,.} \tag{9.60}$$

Upper limits are determined excluding the range of the *parameters of interest* (e.g.: a particle's mass) for which $\mathrm{CL}_s(\theta)$ is lower than the conventional exclusion confidence level (typically 95% or 90%). For this reason, the modified frequentist approach is often referred to as the *CL_s method*.

In most of the cases, the probabilities P_{s+b} and P_b are not trivial to obtain analytically and are determined using numerical Monte Carlo extractions, often referred to as *pseudoexperiments*, or *toy Monte Carlo*. In this way, CL_{s+b} and CL_b can be estimated as the fraction of tossed pseudoexperiments for which $\lambda(\theta) \leq \hat{\lambda}$ in the cases of $s+b$ and b respectively. An example of the outcome of this numerical approach is shown in Fig. 9.14.

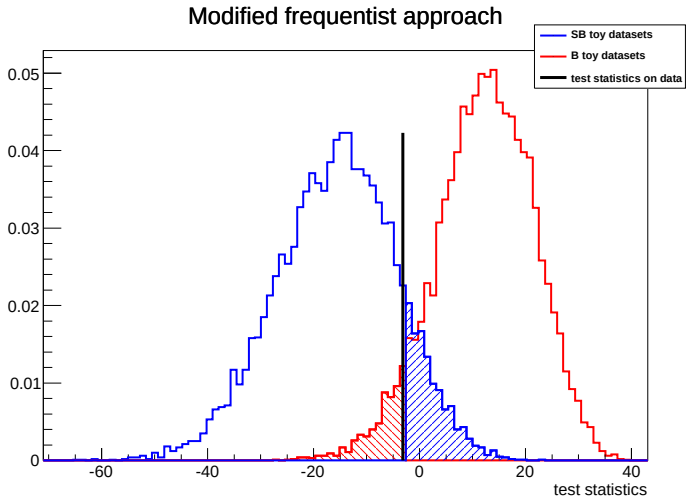


Figure 9.14: Example of determination of CL_s from pseudoexperiments. The distribution of the test statistics $-2\ln\lambda$ is shown in blue in the signal-plus-background hypothesis and in red in the background-only hypothesis. The black line shows the value of the test statistics measured in data, and the hatched areas represent CL_{s+b} (blue) and $1-\mathrm{CL}_b$ (red).

This method does not produce the desired (90% or 95%, usually) coverage from the frequentist point of view, but does not suffer from the problematic features of frequentist upper limits that were observed at the end of Section 9.12. The CL_s method has convenient statistical properties:

- It is conservative from the frequentist point of view. In fact, since $\mathrm{CL}_b \leq 1$, we have that $\mathrm{CL}_s(\theta) \geq \mathrm{CL}_{s+b}(\theta)$. So, it *overcovers*. This means that a CL_s upper limit is less stringent than a purely frequentist limit.

- Combining several measurements can be performed by multiplying likelihood functions of individual channels to produce a combined likelihood function. This is an advantage from the technical point of view. Moreover, combining a measurement with a second measurement with low sensitivity implies multiplying λ by the likelihood ratio of the added channel which is close to one (the $s+b$ and b hypothesis have similar values of the likelihood functions if the sensitivity to signal is low), hence the combined test statistics is not much different from the most sensitive measurement and the corresponding limit won't be much different from the one obtained using the most sensitive channel only.

- If no signal event is observed ($n = 0$), the observed limit does not depend on the expected amount of background.

For a simple Poissonian counting experiment with expected signal s and a background b, using the likelihood ratio of Eq. 9.52, one can demonstrate that the CL_s approach leads to a result identical to the Bayesian one (Eq. 9.20). And in general, it turns out that often numerically CL_s are very similar to Bayesian upper limits computed with a uniform prior. But of course the meaning of Bayesian limits is very different.

Anyway, the interpretation of limits obtained using the CL_s method is not obvious, and it does not match neither the frequentist nor the Bayesian approaches. It has been defined as [15]:

"approximation to the confidence in the signal hypothesis, one might have obtained if the experiment had been performed in the complete absence of background."

9.15 Incorporate systematic uncertainties (nuisance parameters)

Some of the parameters in the set $\vec{\theta} = (\theta_1, \cdots, \theta_m)$ are not of direct interest for our measurement, but are needed to model unknown characteristics of our data sample. Those parameters are defined *nuisance parameters*. Nuisance parameters may appear when the yield of the observed background is estimated with some uncertainty from simulation or control samples in data, or in the modeling of distributions of the observed variables in signal and background events, including the effect of detector resolution. The resolution needed to model the experimental width of a new particle's mass peak is an example of nuisance parameter. If we are only interested in the measurement of a signal strength μ, all parameters θ_i are nuisance parameters. In case we are also interested in the measurement of the mass of a new particle, like the Higgs boson, the parameter corresponding to the particle mass, say θ_1, is, like μ, a *parameter of interest* (sometimes referred to as POI) and $\theta_2, \cdots, \theta_m$ are nuisance parameters. More in general, let's divide the parameter set in two sets: the POIs $\vec{\theta} = (\theta_1, \cdots, \theta_h)$ and the nuisance parameters, $\vec{\nu} = (\nu_1, \cdots, \nu_l)$, where $m = h + l$.

The treatment of nuisance parameters is a well defined task under the Bayesian approach. The posterior joint probability distribution for all the unknown parameters can be defined as follows:

$$P(\vec{\theta}, \vec{\nu}|\vec{x}) = \frac{L(\vec{x}; \vec{\theta}, \vec{\nu})\pi(\vec{\theta}, \vec{\nu})}{\int L(\vec{x}; \vec{\theta'}, \vec{\nu'})\pi(\vec{\theta'}, \vec{\nu'})\mathrm{d}^h\theta'\mathrm{d}^l\nu'}\,, \tag{9.61}$$

where $\pi(\vec{\theta}, \vec{\nu})$ is the prior distribution of the unknown parameters and $L(\vec{x}; \vec{\theta}, \vec{\nu})$ is the likelihood function. The probability distribution of $\vec{\theta}$ can be obtained as marginal PDF, integrating the joint PDF over all nuisance parameters:

$$P(\vec{\theta}|\vec{x}) = \int P(\vec{\theta}, \vec{\nu}|\vec{x})\mathrm{d}^l\nu = \int \frac{L(\vec{x}; \vec{\theta}, \vec{\nu})\pi(\vec{\theta}, \vec{\nu})}{\int L(\vec{x}; \vec{\theta'}, \vec{\nu'})\pi(\vec{\theta'}, \vec{\nu'})\mathrm{d}^h\theta'\mathrm{d}^l\nu'}\mathrm{d}^l\nu\,. \tag{9.62}$$

The problem is well defined, and the only difficulty is the numerical integration in multiple dimensions. Several algorithms can be adopted for this problem; a particularly performant algorithm in those cases is the Markov-chain Monte Carlo [16].

The treatment of nuisance parameters under the frequentist approach is more difficult to perform rigorously. Cousins and Highlands [17] proposed to adopt the same approach used for the Bayesian treatment to determine approximate likelihood functions for the signal-plus-background and background-only hypotheses. This *hybrid* Bayesian-frequentist approach does not provide an exact frequentist solution, but in most of the cases can be proven to be a very close approximation to the exact treatment. The hybrid likelihood functions can be written, integrating Eq. 9.9, as:

$$L_{s+b}(\vec{x}_1, \cdots, \vec{x}_k|\mu, \vec{\theta}) = \frac{1}{n!}\int e^{-\left(\mu s(\vec{\theta}, \vec{\nu}) + b(\vec{\theta}, \vec{\nu})\right)} \prod_{i=1}^{n} \Big(\mu s(\vec{\theta}, \vec{\nu}) f_s(\vec{x}_i; \vec{\theta}, \vec{\nu}) + b(\vec{\theta}, \vec{\nu}) f_b(\vec{x}_i; \vec{\theta}, \vec{\nu})\Big)\,\mathrm{d}^l\nu\,, \tag{9.63}$$

$$L_b(\vec{x}_1, \cdots, \vec{x}_k|\vec{\theta}) = \frac{1}{n!}\int e^{-b(\vec{\theta}, \vec{\nu})} b(\vec{\theta}, \vec{\nu})^n \prod_{i=1}^{n} f_b(\vec{x}_i; \vec{\theta}, \vec{\nu})\mathrm{d}^l\nu\,.$$

In order to include detector resolution effects, for instance to model the width of a signal peak, the hybrid

approach requires the convolution of the likelihood function with the experimental resolution function.

The above likelihood functions can be used to compute CL_s limits, as it was done in the combined Higgs limit at LEP [14].

In the case of an event counting problem, if the number of background events is known with some uncertainty, the PDF of the background estimate b' can be modeled as a function of the true unknown expected background b: $P(b';b)$. The likelihoods, as a function of the parameter of interest s and the unknown nuisance parameter b, can be written as:

$$L_{s+b}(n,b';s,b) = \frac{(s+b)^n}{n!}e^{-(s+b)}P(b';b)\,, \tag{9.64}$$

$$L_b(n,b';b) = \frac{b^n}{n!}e^{-b}P(b';b)\,. \tag{9.65}$$

In order to eliminate the dependence on the nuisance parameter b, the hybrid likelihoods can be written as:

$$L_{s+b}(n,b';s) = \int_0^\infty \frac{(s+b)^n}{n!}e^{-(s+b)}P(b';b)\mathrm{d}b\,, \tag{9.66}$$

$$L_b(n,b') = \int_0^\infty \frac{b^n}{n!}e^{-b}P(b';b)\mathrm{d}b\,. \tag{9.67}$$

In the most lucky case, for instance when $P(b';b)$ is a Gaussian function, the integration can be performed analytically [18]. In this case, when the r.m.s of the distribution is not much smaller than b', $P(b';b)$ extends to negative values of b, and the integration includes unphysical regions. In order to avoid such cases, distributions whose range is limited to positive values is preferred. For instance, a log-normal distribution (the distribution of a random variable whose logarithm is distributed according to a Gaussian) is usually preferred to a plain Gaussian.

9.16 Profile likelihood

An alternative procedure to the hybrid treatment of nuisance parameters is to introduce the *profile likelihood* defined as follows:

$$\lambda(\mu) = \frac{L(\mathrm{data}|\mu,\hat{\hat{\vec{\theta}}}(\mu))}{L(\mathrm{data}|\hat{\mu},\hat{\vec{\theta}})}\,, \tag{9.68}$$

where $\hat{\mu}$ and $\hat{\vec{\theta}}$ are the best fit values for μ and $\vec{\theta}$ corresponding to the observed data sample, and $\hat{\hat{\vec{\theta}}}(\mu)$ is the best fit value for $\vec{\theta}$ obtained for a fixed value of μ. Above we have assumed that all parameters are treated as nuisance parameter and μ is the only parameter of interest.

Usually the distribution of the profile likelihood function is broadened with respect to the original likelihood function, due to the loss of information introduced by the presence of nuisance parameters.

The profile likelihood cannot be treated as an ordinary likelihood function which depends only o μ, but has several interesting property that make it more convenient to use than the hybrid likelihoods, since it requires no numerical integration. In particular, being defined as the ratio of two likelihood functions, the Wilks theorem can be applied, in case of sufficiently large samples. In this case, the distribution of the test statistics $q_\mu = -2\ln\lambda(\mu)$ is asymptotically distributed according to a χ^2 with one degree of freedom (corresponding to the single parameter of interest not being profiled), and the significance corresponding to value of μ that minimizes q_μ can be approximated as $Z_\mu \simeq \sqrt{q_\mu}$ [19].

Different variation in the definition of the profile likelihood have been proposed and adopted by various experiment. A review of the main adopted procedures at LEP, Tevatron and LHC can be found in [20].

In particular, for Higgs search at LHC the adopted test statistics is:

$$\tilde{q}_\mu = \begin{cases} -2\ln\frac{L(\text{data}|\mu,\hat{\hat{\vec{\theta}}}(\mu))}{L(\text{data}|0,\hat{\hat{\vec{0}}})} & \hat{\mu} < 0\,, \\ -2\ln\frac{L(\text{data}|\mu,\hat{\hat{\vec{\theta}}}(\mu))}{L(\text{data}|\hat{\mu},\hat{\vec{\theta}})} & 0 \le \hat{\mu} \le \mu\,, \\ 0 & \hat{\mu} > \mu\,. \end{cases} \tag{9.69}$$

Above, the constraint $\hat{\mu} < 0$ protects against unphysical values of the signal strength, while the cases which have an upward fluctuations of the data, such to give $\hat{\mu} > \mu$, are not considered as evidence against the signal hypothesis with signal strength equal to μ, setting the test statistics to zero in those cases. For the definition of the $\tilde{q}_\mu$ test statistics, as well for the most adopted variations of the profile likelihood, asymptotic approximations which extend the results of Wilk's theorem have been computed and are treated extensively in [4]. As an example, the asymptotic approximation for the distribution of $\tilde{q}_\mu$ is:

$$f(\tilde{q}_\mu|\mu) = \frac{1}{2}\delta(\tilde{q}_\mu) + \begin{cases} \frac{1}{2\sqrt{2\pi}}\frac{1}{\sqrt{\tilde{q}_\mu}}e^{-\tilde{q}_\mu/2} & 0 < \tilde{q}_\mu \le \mu^2/\sigma^2\,, \\ \frac{1}{\sqrt{2\pi}(2\mu/\sigma)}\exp\left[-\frac{1}{2}\frac{(\tilde{q}_\mu+\mu^2/\sigma^2)^2}{(2\mu/\sigma)^2}\right] & \tilde{q}_\mu > \mu^2/\sigma^2\,, \end{cases} \tag{9.70}$$

where $\delta(\tilde{q}_\mu)$ is a Dirac delta function, to model the cases in which the test statistics is set to zero, and where $\sigma^2 = \mu^2/q_{\mu,A}$, in which $q_{\mu,A}$ is the value of the test statistics $-2\ln\lambda$ evaluated on the so-called *Asimov set* [21], i.e.: a *representative* data set in which the yields of all data samples are set to their expected values and nuisance parameter at their nominal value. Asimov sets can also be used to compute approximate estimates of expected experimental sensitivity, which would require the extraction of a large number of pseudoexperiments. The square roots of the test statistics evaluated at Asimov data sets corresponding to the assumed signal strength μ can be used to approximate the median significance, assuming a data sample distributing according to the background-only hypothesis:

$$\text{med}[Z_\mu|0] = \sqrt{\tilde{q}_{\mu,A}}\,. \tag{9.71}$$

A comprehensive treatment of asymptotic approximations can be found in [4].

9.17 The look-elsewhere effect

In several cases experiments look for resonances at unknown mass values. This is the case, for instance, of the Higgs boson search. If an excess of data compared to the background expectation is found at *any* mass value it can be interpreted as possible signal of the new resonance at the observed mass, but the peak could be produced either by the presence of a real signal or by a background fluctuation. The computation of the signal significance can be done from the p-value of the measured test statistics q assuming a fixed value m_0 of the resonance mass. This is called *local significance*, and can be written as:

$$p(m_0) = \int_{q_{\text{obs}(m_0)}}^{\infty} f(q|\mu = 0)\text{d}q\,, \tag{9.72}$$

where $f(q|\mu)$ is the PDF of the adopted test statistics q for a given value of the signal strength μ. The local significance gives the probability corresponding to a background fluctuation at a fixed value of the mass m_0. The probability to have a background overfluctuation at *any* mass value, called *global* p-value, is larger than the local p-value, which underestimates the probability of a background fluctuation at *any* mass value in the range of interest, which would measure the *global* significance.

The magnitude of the effect is larger as the mass resolution gets worse. In fact, assuming a small intrinsic width of the new particle, a very good mass resolution implies that a peak can appear from a background fluctuation if background events masses are by chance all close within the experimental resolution, which is less likely as the resolution gets smaller.

More in general, when an experiment is looking for a signal where one or more parameters $\vec{\theta}$ are unknown (could be the mass, the width and other properties of a new signal) in the presence of an

excess in data with respect to the background expectation, the unknown parameter (or parameters) can be determined from the data sample itself. In those cases, the local significance, expressed in terms of a p-value computed at fixed values of the unknown parameter set $\vec{\theta}_0$ is an underestimate of the global significance, which expresses the probability associated to a background fluctuation for any possible values of the parameter in the range of interest. The global p-value can be computed using as test statistics the largest value of the estimator over the entire parameter range:

$$q(\hat{\vec{\theta}}) = \max_{\substack{\theta_i^{\min}<\theta_i<\theta_i^{\max},\\ i=1,\cdots,m}} q(\vec{\theta})\,. \tag{9.73}$$

The distribution of $q(\hat{\vec{\theta}})$ from Eq. (9.73) is not easy to determine, and usually intensive random extraction of pseudo experiments are needed. In order to determine a significance close to the discovery level of 5σ, p-values of the order of 3×10^{-7} need to be evaluated, hence tens of millions of pseudo experiment representing background only needed to be extracted, and in many case this brute-force approach is intractable.

An approximate way to determine a global significance taking into account the look-elsewhere effect is reported in [22], relying on asymptotic behavior of likelihood ratio estimators. It is possible to demonstrate [23] that the probability that the test statistics $q(\hat{m})$ is larger than a given value u is bound by:

$$p^{\text{glob}} = P(q(\hat{m}) > u) \leq \langle N_u \rangle + P(\chi^2 > u)\,, \tag{9.74}$$

where $P(\chi^2 > u)$ comes from the Wilk's asymptotic approximation of the distribution of the local test statistics $q(m)$ as a χ^2 distribution with one degree of freedom, and $\langle N_u \rangle$ is the average number of *upcrossings*, i.e. the average number of times the curve $q = q(m)$ crosses an horizontal line at a given level $q = u$ with a positive derivative. This is visualized in an example in Fig. 9.15.

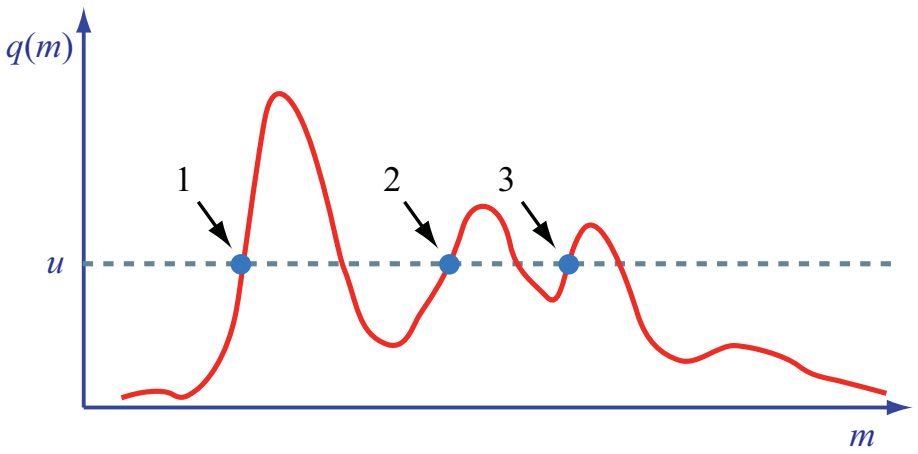


Figure 9.15: Visual illustration of upcrossing, computed to determine $\langle N_{u_0} \rangle$. In this example, we have $N_u = 3$.

The value of $\langle N_u \rangle$ could be very small, depending on the level u. Fortunately, a scaling law exists, so, starting from a different level u_0 one can extrapolate $\langle N_{u_0} \rangle$ as:

$$\langle N_u \rangle = \langle N_{u_0} \rangle\, e^{-(u-u_0)/2}\,. \tag{9.75}$$

This allows to evaluate $\langle N_{u_0} \rangle$ generating a number of pseudo experiment much smaller than what would be needed to determine $\langle N_u \rangle$ with comparable precision.

References

[1] J. Neyman, E. Pearson, "On the Problem of the Most Efficient Tests of Statistical Hypotheses", Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character, **231** (1933) 289.

[2] S. Wilks, "The large-sample distribution of the likelihood ratio for testing composite hypotheses", Ann. Math. Stat., **9** (1938) 60.

[3] S. Baker, R.D. Cousins, "Clarification of the use of chi-square and likelihood functions in fit to histograms", Nucl. Instr. Meth., **A221** (1984), 437.

[4] G. Cowan, K. Cranmer, E. Gross and O. Vitells, "Asymptotic formulae for likelihood-based tests of new physics", Eur.Phys.J. **C71** (2011) 1554.

[5] O. Helene, "Upper limit of peak area", Nucl. Instr. and Meth. **A212** (1983) 319.

[6] G. D'Agostini, "Bayesian Reasoning in Data Analysis: A Critical Introduction", World Scientific (2003) ISBN 981-238-356-5,

[7] J. Jeffreys, "An Invariant Form for the Prior Probability in Estimation Problems", Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences **186** no. 1007 ((1946) 453.

[8] G. Zech, "Upper limits in experiments with background or measurement errors", Nucl. Instr. and Meth. **A277** (1989) 608.

[9] V.L. Highland, R.D. Cousins, "Comment on "Upper limits in experiments with background or measurement errors" [Nucl. Instr. and Meth. A277 (1989) 608-610]", Nucl. Instr. and Meth. **A398** (1989), 429.

[10] G. Zech, "Reply to "Comment on "Upper limits in experiments with background or measurement errors" [Nucl. Instr. and Meth. A 277 (1989) 608-610]" ", Nucl. Instr. and Meth. **A398** (1989) 431.

[11] J. Neyman, J, "Outline of a theory of statistical estimation based on the clasiscal theory of probability", Philosophical Transactions of the Royal Society of London, **A236**, no. 767 (1937), 333.

[12] G.J. Feldman, R.D. Cousins, "Unified approach to the classical statistical analysis of small signals", Phys. Rev. **D57** (1998) 3873.

[13] C. Amsler, C. it et al. (Particle Data Group), "The Review of Particle Physics", Phys. Lett. **B667** (2008) 1.

[14] G. Abbiendi *et al.* (The LEP Working Group for Higgs Boson Searches), "Search for the Standard Model Higgs Boson at LEP", Phys. Lett. **B565** (2003) 61.

[15] A.L. Read, "Modified frequentist analysis of search results (the CL_s method)", 1st Workshop on Confidence Limits", CERN (2000).

[16] B.A. Berg, "Markov Chain Monte Carlo Simulations and Their Statistical Analysis", World Scientific", Singapore (2004).

[17] R.D. Cousins, V.L. Highland, "Incorporating Syst ematic Uncertainties into an Upper Limit", Nucl. Instr. Meth. **A320** (1992) 331.

[18] L. Lista, "Including gaussian uncertainty on the background estimate for upper limit calculations using Poissonian sampling", Nucl. Instr. Meth. **A517** (2004) 360.

[19] G. Cowan *et al.*, "Asymptotic formulae for likelihood-based tests of new physics" EPJC **71** (2011) 1554.

[20] The ATLAS Collaboration, the CMS collaboration, the LHC Higgs combination group, "Procedure for the LHC Higgs boson search combination in Summer 2011", ATL-PHYS-PUB-2011-011, CMS NOTE-2011-005 (2011)

[21] I. Asimov, "Franchise", in I. Asimov, "The Complete Stories", vol. 1, Broadway Books, New York, 1990.

[22] E. Gross, O. Vitells, "Trial factors for the look elsewhere effect in high energy physics", Eur. Phys. J. **C70** (2010) 525.

[23] R.B. Davies, "Hypothesistestingwhenanuisance parameter is present only under the alternative", Biometrika **74** (1987), 33.

more