Computational Statistic

unibo/terzo-anno/Computational Statistic/AR Algorithm.md

Combining Distributions

> Height: bi-normal distribution (men and women)

$$Y \sim Gamma(n, \beta)$$
$$X \mid Y \sim Poisson(y)$$

$$\int_{-\infty}^{\infty} f_{X|Y} f_Y \, dy = f_X$$

taking a look at the joint distribution of X and Y and the marginal distribution of Y.

$$\begin{aligned} & X, Y \text{ two continuous r.vs} \\ f_{X|Y=y}(x) &= \frac{f_{X,Y}(X,Y)}{f_Y(y)} \\ &= \frac{f_{Y|X=x}(y) \cdot f_X(x)}{f_Y(y)} \\ &= \frac{f_{Y|X=x}(y) \cdot f_X(x)}{\int_{\mathbb{D}x} f_{X,Y}(u,y)\, dx} \end{aligned}$$

Expected value

$$E_{f(X)}(X) = \int_{\mathbb{D}x} x f_X(x)\, dx$$

Accept-Reject Algorithm

> The algorithm is used to sample from a distribution that is **hard to sample** from. The idea is to sample from a distribution that is easy to sample from and then to accept or reject the sample based on a certain criterion.

using two functions:

- $f(x)$: the target distribution
- $g(y)$: the candidate distribution

How to select the candidate

use a function that has tails bigger than the target distribution, otherwise it will explode

Constraints

- $f(x) \leq M \cdot g(x)$
- $g(x) > 0$ for all x when $f(x) > 0$

Steps

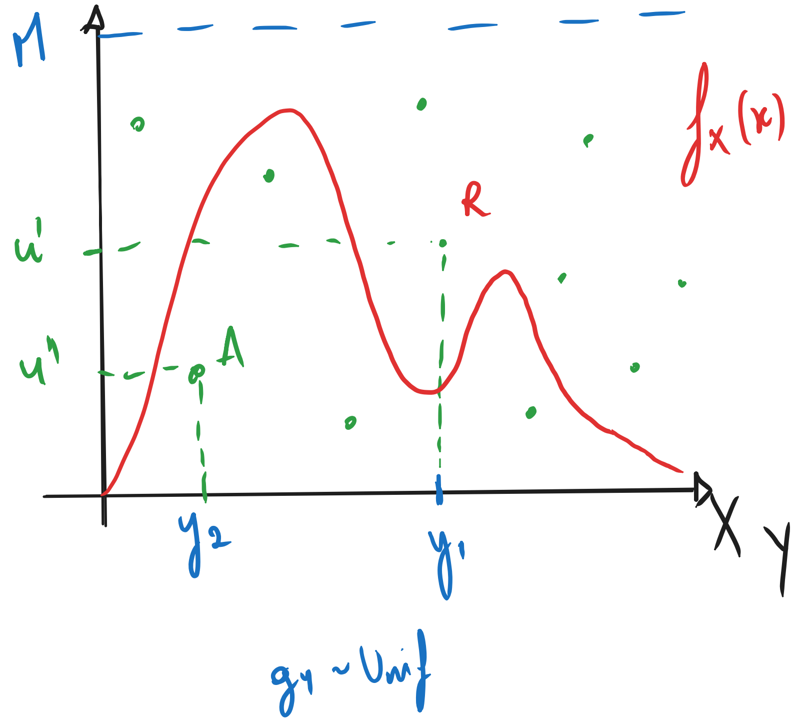
1. Sample $Y \sim g(y)$
2. Sample u from $U \sim Uniform(0, 1)$
    1. If $u \leq \frac{1}{M} \cdot \frac{f(Y)}{g(Y)}$ then accept Y as a sample from $f(x)$
    2. Else, reject Y and go back to step 1

Code

```
u=runig(1)*M
y=randg(1)
while(u>f(y)/g(y)) {
        u=runif(1)*M
        y=randg(1)
}
```

Problems

- if M large, the acceptance rate is low
- if M small, the acceptance rate is high but the algorithm is slow
- if the ratio is not bounded at some point it will explode ⇒ if you are accepting a lot of samples, you are not sampling from the right distribution (especially if the candidate is the target)
- the candidate CANNOT be the target
    - it's the best approximation
    - you already know how to sample from the target



Unif

what's the best value for M?
the tanget in the point where the ratio is maximal.

> **exam**
>
> function in R that calculate the maximum ⇒ write the **value**.
> We are not doing the derivative and etc...

we are giving horizontally the same probability to all values. Is this the best approach? No, we need to follow the density of the target!

> **Un-normalized density**
>
> $$\forall x \in \mathbb{D}_x \; f(x) \geq 0$$
> $$\int_{-\infty}^{\infty} f(x)\, dx = k \neq 1$$

Plot the normalized and the non-normalized density to show the difference.

r from a *beta*:

```
Nsim=10000
X=rbeta(Nsim, 2.7, 6.3)
hist(X, freq=F)

# let's calculate the density with A/R and then compare with the
normalized density
f=function(x) dbeta(x,2.7,6.3)
g=function(x) dunif(x,0,1)
# get the maximum
M=optimize(f,lower=0,upper=1)$objective
# generate the sample
Y=numeric(Nsim)
for(i in 1:Nsim){
  u=runif(1)*M
  y=runif(1)
  while(u>f(y)/g(y)){
    u=runif(1)*M
    y=runif(1)
  }
  Y[i]=y
}

# Istogramma dei campioni diretti
hist(X, freq = FALSE, col = rgb(0, 0, 1, 0.5), main = "Confronto
tra rbeta e AR", xlab = "x")
# Istogramma dei campioni AR
hist(Y, freq = FALSE, col = rgb(1, 0, 0, 0.5), add = TRUE)
# Densità teorica
curve(f, 0, 1, add = TRUE, col = "black", lwd = 2)
legend("topright", legend = c("rbeta", "AR", "Densità teorica"),
       fill = c(rgb(0, 0, 1, 0.5), rgb(1, 0, 0, 0.5), NA),
       border = c("blue", "red", NA), lty = c(NA, NA, 1), lwd =
c(NA, NA, 2))
```

Laplace

basically a double exponential distribution. It has a peak in the middle and then it goes down.

The best candidate is the normal distribution.

es 2.18

$$f(x) \propto \exp\left(-\frac{x^2}{2}\right) \cdot \{\sin(6x)^2 + 4\cos(x) * 2\sin(4x)^2 + 1\}$$

note that the function is **not** normalized.

- we want to sample using AR algorithm.

I. plot $f(x)$ and show it can b bounded by $\tilde{M} \cdot g(x)$, with
$g(x) = \frac{1}{2\pi}\exp\left(-\frac{x^2}{2}\right)$
II. generate 2500 balues from $f(x)$ using AR
III. Decide from the acceptance rate an approximation of the normalized constant for $\tilde{f}(x)$; compare the histogram of the generated sequecnce with $f(x)$

> **Note**
>
> Check the histogram if match with the normalized function!!!
> and with the acf plot

es 2.19

$$f_X(x) = Norm(0,1) \quad f_X(x) = \frac{1}{2\pi} \cdot \exp\left(-\frac{x^2}{2}\right)$$

$$g(x|\alpha) = \frac{\alpha}{2} \cdot \exp(-\alpha \cdot \mid x \mid) \text{ laplace/double exponential}$$

I. Compute M and show that $\alpha = 1$ is the value optimizing the acceptance rating

$$
\begin{gathered}
r = \frac{1}{M} \quad M = \sqrt{\frac{2}{\pi}} \cdot \exp\left\{\frac{\alpha^2}{2}\right\} \cdot \frac{1}{\alpha} \\
r = \frac{1}{\sqrt{\frac{2}{\pi}}} \cdot \alpha \cdot \exp\left\{-\frac{\alpha^2}{2}\right\} \\
r' = \exp\left\{-\frac{\alpha^2}{2}\right\} + \alpha(-\alpha) \cdot \exp\left\{-\frac{\alpha^2}{2}\right\} \\
= \exp\left\{-\frac{\alpha^2}{2}\right\} \cdot (1-\alpha^2) \\
\text{so the maximim is when } (1-\alpha^2) = 0 \\
\alpha = \pm 1 \wedge \alpha > 0 \implies \alpha = 1
\end{gathered}
$$

for this exercise we calculated alpha analytically, but we can also found it by trials.

unibo/terzo-anno/Computational Statistic/Discrete Distributions.md

Extracted from: PIT

Discrete Distributions

$$
\begin{aligned}
&X \sim \text{random variable} \\
&D_X = \{1,2,3\} \qquad \text{prob. MASS function} \\
&P(X=x) = P_X(x) \\
&P_X(1) = 0.5 \qquad P_X(2) = 0.3 \qquad P_X(3) = 0.2 \\
&F_X(x) = \sum_{y=1}^{X} P_X(y)
\end{aligned}
$$

> What if we flip the plot?

What we get on the X axes is a $Unif$ that has segments corresponding to the domain of my initial D_X.

1. **Understand the PIT Principle**:

For a random variable (X) with cumulative distribution function (CDF) (F_X(x)), the PIT states that:

- If (X \sim F_X), then (U = F_X(X)) is uniformly distributed on ([0, 1]) for continuous distributions.

For **discrete distributions**, however, the CDF (F_X(x)) is a step function, and the probability mass function (PMF) (P(X = x)) assigns probabilities to discrete points.

---

2. **Adapting the PIT for Discrete Distributions**:

In the discrete case, the transformation (U = F_X(X)) does not result in a truly uniform random variable because the CDF takes discrete jumps. Instead, you can define (U) as a random variable uniformly distributed over the interval corresponding to (F_X(X - 1)) and (F_X(X)):

$$U \sim \text{Uniform}(F_X(X-1), F_X(X)),$$

where (F_X(X-1)) is the CDF value just before the jump at (X).

---

3. **Steps to Apply the PIT to a Discrete Distribution**:

1. **Calculate the CDF**: For a discrete random variable (X) with PMF (P(X = x)), compute the CDF:

$$F_X(x) = P(X \leq x) = \sum_{k \leq x} P(X = k).$$

2. **Generate the Random Variable**: For a given realization (x) of (X), identify the interval ([F_X(x-1), F_X(x)]).
3. **Transform to Uniform**: Sample (U) from (\text{Uniform}(F_X(x-1), F_X(x))).

---

4. **Illustrative Example**:

Consider a discrete random variable (X) with PMF:
[

P(X = 1) = 0.2, , P(X = 2) = 0.5, , P(X = 3) = 0.3.
]

- Compute the CDF:

$$F_X(x) = \begin{cases} 0, & x < 1, \\ 0.2, & 1 \leq x < 2, \\ 0.7, & 2 \leq x < 3, \\ 1, & x \geq 3. \end{cases}$$

- For (X = 2), the interval is ([F_X(1), F_X(2)] = [0.2, 0.7]).
- Sample (U \sim \text{Uniform}(0.2, 0.7)).

---

5. **Key Observations**:

- The transformed variable (U) is not exactly uniform over ([0, 1]) but matches the discrete CDF.
- This adaptation ensures the transformation respects the discrete nature of the distribution.

```
# Load the MASS package
library(MASS)

# Parameters
r <- 10
p_values <- c(0.01, 0.1, 0.5)
n <- 1000

# Custom negative binomial generator
generate_negbin <- function(r, p, n) {
  replicate(n, {
    y <- 0
    prod <- 1
    while (prod > runif(1)) {
      prod <- prod * (1 - p)
      y <- y + 1
    }
    y - 1
  })
```

```
}

# Plotting
par(mfrow = c(3, 2))  # 3 rows, 2 columns for plots
for (p in p_values) {
  # Generate random variables
  my_samples <- generate_negbin(r, p, n)
  rnegbin_samples <- rnegbin(n, mu = r * (1 - p) / p, theta = r)

  # Histograms
  hist(my_samples, breaks = 20, prob = TRUE, main =
paste("Custom Generator, p =", p),
       xlab = "Value", col = "lightblue")
  hist(rnegbin_samples, breaks = 20, prob = TRUE, main =
paste("rnegbin, p =", p),
       xlab = "Value", col = "lightgreen")

  # Overlay theoretical probabilities
  x_vals <- 0:max(my_samples)
  theoretical_probs <- dbinom(x_vals, size = r, prob = 1 - p)
  points(x_vals, theoretical_probs, col = "red", pch = 16)
}
```

unibo/terzo-anno/Computational Statistic/GTM.md

Extracted from: PIT

General Transformation Method

> How can we calculate the density function after a transformation?

$$
\begin{aligned}
X \sim f_X(x) \qquad & x \in D_X \text{ continuos} \\
Z = g(x) \qquad & \text{such that } g^{-1}(z) = x \\
f_Z(z) = f_X(g^{-1}(z)) \cdot \left| \frac{\partial g^{-1}(z)}{\partial z} \right| &
\end{aligned}
$$

Note

Look at the example on the slides with the uniform distribution.

Box Muller algorithm

another direct method of calculating two "normals" from two uniforms.

using sin and cos because at a certain point we cannot no longer resolve an integral analytically, so we switch to polar coordinates.

`rnorm` → not using box Muller alg. instead uses a PIT, with an accurate representation of the normal cdf.

We also need to distinguish the difference between sampling error and numerical calculation error.

Multi variant Normals

- Cholesky decomposition $\Sigma = AA'$
- $Y \sim N_p(0, I) \implies AY \sim N_p(0, \Sigma)$

unibo/terzo-anno/Computational Statistic/Importance Sampling.md

Introduction

$$\mathbb{E}[h(X)] = \int_X h(x) \frac{f(x)}{g(x)} g(x)\, dx = \mathbb{E}_g \left[\frac{h(X) f(X)}{g(x)} \right]$$

you want a w_i not exploding that matches:

$$f(x_i) \cdot h(x_i)$$

Exercise

$$P(Z > 4.5) \quad \text{where} \quad Z \sim N(0,1)$$
$$pnrom(-4.5) = 3.39e - 06$$

$$\int_{4.5}^{\infty} \mathbb{1} \, f_Z(z) \, dz$$

so in this case we use importance sampling:

$$\begin{aligned} supp(h \times f) &\subseteq supp(g) \\ [4.5, \infty[&\subseteq [4.5, \infty[\end{aligned}$$

$g(z)$ trucated Exp = exponential shifted by 'a'

$$\begin{aligned} Z &\sim Exp(1) \\ Z &\in [-, +\infty[\\ (Z + a) &= Exp + N(1, a) \end{aligned}$$

3.4

$$h(x) \exp\left\{-\frac{(x-3)^2}{2}\right\} + \exp\left\{-\frac{(x-6)^2}{2}\right\}$$

$f(x)$ ⇒ pdf of a $Gaussian(\mu, \sigma^2)$

a. show that $E_f(h(X))$ has a closed from solution

$$X \cdot Y??$$

b. use classical mc integration with $f_X = N(0, 1)$ with $n = 10^3$
c. use the important sampling $g \sim Unif[-8, -1]$

unibo/terzo-anno/Computational Statistic/MASS.md

extracted-from:: Monte Carlo Integration

MASS library R

a cosa serve?

Quali funzioni vengono usate?

- `area`
- `dlaplace`

unibo/terzo-anno/Computational Statistic/Monte Carlo Integration.md

🤑 Introduction

$$\int_X h(x) f(x) \, dx.$$

- f density

Probabilistic resoults:

- law of large numbers
- central limit theorem

$$\bar{h} = \frac{1}{n} \sum_{i=1}^{n} h(x_i) \rightarrow_{n \rightarrow \infty} E_f(h(x))$$

sample random from the density

$$var(\bar{h}_n) = \frac{\sigma^2}{n} \text{ extimator}$$

$$Var(X) = \int_{-\infty}^{\infty} (X - E_f(x)) f_X(x) \, dx = E((X - u)^2)$$

Definition

Definizione

Monte Carlo integration is a numerical method that uses *random sampling* to **estimate the value of an integral**. It is particularly useful for solving integrals in high-dimensional spaces where traditional numerical methods (like trapezoidal or Simpson's rule) become computationally expensive or impractical.

Confident Interval

Confidence interval (CI) is a range of values used to **estimate an unknown** population parameter (e.g., mean, proportion) with a certain level of confidence. It provides an *interval* that, if the *process were repeated multiple times*, would contain the true parameter in a specified **percentage of cases**.

Formula

$$\int_X h(x)f(x)\,dx = E_f(h(x))$$

Notes

- in statistic, integrals and samplings are often switched

integrals not in 0-1

if we are integrating something like:

$$\int_0^2 h(x)\,dx$$

$$unif(0,2) = \frac{1}{2} \text{ is the density}$$

we can still use the integral, we need to adjust the density:

$$\int_0^2 h(x)\,dx = 2\int_0^2 \frac{1}{2}h(x)\,dx = 2 \cdot E_{unif}(h(x))$$

Exercise

3.1

solution to the normal-Cauchy estimator (Bayesian):

$$\delta(x) = \frac{\int_{-\infty}^{\infty} \theta \cdot \frac{1}{1+\theta^2} \cdot \exp(-\frac{(x-\theta)^2}{2})\,d\theta}{\int_{-\infty}^{\infty} \frac{1}{1+\theta^2} \cdot \exp(-\frac{(x-\theta)^2}{2})\,d\theta}$$

solve the question for $x = \{0, 2, 4\}$

1. plot the *integrands* and use MC integration based on Cauchy sampling.

Note

- if you have the kernel of a gaussian, we can switch the sign on the kernel and the integral will still be the same.
- we also don't have pi in the Cauchy distribution → it's a constant so we can add it!

```
# Point a.
num <- function(x,the){(the/(1+the^2))*exp(-(x-the)^2/2)}
# \int h(x)f(x) dx
den <- function(x,the){(1/(1+the^2))*exp(-(x-the)^2/2)}

num0 <- function(the) num(0,the)
num2 <- function(the) num(2,the)
num4 <- function(the) num(4,the)
den0 <- function(the) den(0,the)
den2 <- function(the) den(2,the)
den4 <- function(the) den(4,the)
```

```
par(mfrow=c(2,1))
curve(num0,-20,20,ylim=c(-0.5,0.5))
curve(num2,-20,20,lty=2,add=T)
curve(num4,-20,20,lty=4,add=T)

curve(den0,-20,20)
curve(den2,-20,20,lty=2,add=T)
curve(den4,-20,20,lty=4,add=T)

integrate(num4,-Inf,Inf)

# just h(x)
hnum <- function(the) the*exp(-(4-the)^2/2)
x1 <- hnum(rcauchy(10^4))
int_num <- mean(x1)*pi
err_num <- sqrt(sum((x1-int_num)^2))/(10^4)
c(int_num-2*err_num,int_num+2*err_num)

hden <- function(the) exp(-(4-the)^2/2)
x2 <- hden(rcauchy(10^4))
int_den <- mean(x2)*pi
err_den <- sqrt(sum((x2-int_den)^2))/(10^4)
c(int_den-2*err_den,int_den+2*err_den)
integrate(den4,-Inf,Inf)
```

empirical CDF: the length of the sequence is your sample.

$$\hat{\Phi}(t) = \frac{1}{n}\sum^{n} 1_{x_i \leq t} \to$$

cost of having a long sequence, time and cost complexity.

unibo/terzo-anno/Computational Statistic/Note esame.md

Note per l'esame

- copiare anche le costanti
- compreso di 3 domande, cap 2, 3, 5

- done all in R

unibo/terzo-anno/Computational Statistic/PIT.md

Monte Carlo Methods

Esempio R

Generators

```
### Built-in Random generation
rgamma(3,2.5,4.5)
curve(dgamma(x,2.5,8), from=0, to=4)
```

`rgamma( ... )`

- families
  - gamma
  - norm
  - Poisson
  - binomial
  - ...
- type:
  - q - quantile
  - r
  - d - take a probability mass function
  - p - cumulative distribution function

Cumulative (Distribution Function): integral of the probability density function.

Cumulative distribution function

By definition

$$\forall x, P(X = x) = 0$$

$$cor(x_1, x_2) = 0 \not\Rightarrow x_1 \perp x_2$$

works only with the Gaussian.

Probability Inverse Transform

```
Nsim=10^4
U=runif(Nsim)
# X=-log(U)
X=-log(1-U)
Y=rexp(Nsim)
par(mfrow=c(1,2))
hist(X,freq=F,main="Exp from Uniform")
curve(dexp(x),col="red",add=TRUE)
hist(Y,freq=F,main="Exp from R")
curve(dexp(x),col="red",add=TRUE)
```

works for any random variable, continuous or discrete.

if X density has f density and cdf F.

we set $F(X) = U$ e risolviamo per X.

$$\begin{aligned} X &\sim Exp(1) & F &= 1 - e^{-x} \\ u &= 1 - e^{-x} & x &= -\log(1-u) \end{aligned}$$

es 2.1:

> **Note**
>
> $$P(U \leq u) = F_U(u) = u$$
>
> thanks to the $Unif$ distribution.

$$F^{-1}(u) = inf\{x; F_X(x) \geq u\}$$
$$F_X(x) = P(X \leq x)$$

F →
input: x, a value from the domain of X
output: a value in $[0, 1]$

F^{-1} →
the opposite

show that $U \sim Unif(0, 1)$
$Z = F_X^{-1}(U)$ distributed like X.

Note

Applying a function to a random variable, you still get the randomness.

CDN is monotonic. When applying to a dis-equality, the sign is preserved.

In practice, what we need to know: we can transform any distribution into a random variable iff $\exists F_X(x)$, invertibile.

HOW TO:

- simulate from the uniform U.
- get the distribution of the sequence.

Exam!!!

$$X = F_X^{-1}(U)$$

Distributions

Exponential

Example 1:
bus stop probability → $\sim Exp$
you can go further in time (non negative).

$$X \sim Exp(1)$$

$$F_X(X) = 1 - e^{-x}$$

$$\begin{aligned} U = F_X(X) = 1 - e^{-x} \\ U - 1 = -e^{-x} \\ -\ln(1-U) = x \end{aligned}$$

- now we generate $u_1, u_2, \ldots u_n$
- compute $x_1, \ldots x_n$

at the end of the exercise we know that:

$$Exp = -\log(Unif(Nsim))$$

Logistic distribution

$$F(x; \mu, s) = \frac{1}{1 + e^{-\frac{x-\mu}{s}}}$$

where:

- x is the variable,
- μ is the location parameter,
- s is the scale parameter.

> **Note**
>
> Pay attention to the **minus** otherwise the integral explode as it's an infinity large value.

$$\begin{gathered} \ldots \\ X = \mu - \beta \log\left(\frac{1-U}{U}\right) \end{gathered}$$

same sequence at the numerator and denominator.

- empirical CDF - CDF
- density approximation - history

Empirical C.D.F.j

$$\hat{F}_{X,n}(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{x_i \leq x\}$$

we count how many numbers are below where i'm currently counting.

Cauchy Distribution

$$F(x; x_0, \gamma) = \frac{1}{\pi}\arctan\left(\frac{x - x_0}{\gamma}\right) + \frac{1}{2}$$

where:

- x is the variable,
- x_0 is the location parameter (the median of the distribution),
- γ is the scale parameter (which determines the half-width at half-maximum).

$$X = \mu + \sigma \cdot \tan\left(\pi\left(U - \frac{1}{2}\right)\right)$$

Chi square distribution

cannot use for odd degree(?) of transformation.

Altre

- Gamma
- Beta

Scale parameter

Most of the times at the denominator.

> Gaussian / Chi-Square (df) → t student
> df - degree of freedom

> X_1^2/X_2^2 → Fischer

unibo/terzo-anno/Computational Statistic/Poisson Ditribution.md

extracted-from::README

Poisson Distribution

$$X \sim Poisson(\lambda)$$

> $E(X) = \lambda = Var(X)$

- over dispersion: $Var(X) > E(X)$
- under dispersion: $Var(X) > E(X)$

if λ is large ⇒ *Gaussian* $X \approx N(\lambda; \lambda)$

Upper limit given. Trimming off the tails of the Normal distribution: the pros are worth it ⇒ $P(X < \lambda - 3\sqrt{\lambda}) + P(X > \lambda + 3\sqrt{\lambda}) \approx 0$

> Introduced to measure the number of events in a fixed interval of time or space → number of people being kicked by donkeys.

R implementation

```
# numero simulazioni, parametro distribuzione
Nsim=10^4; lamda = 100;

# calculate the spread
spread= 3 * sqrt(lamda)
```

```r
# sequenza di valori all'interno del range
t=round(seq(max(0, lamda-spread), lamda+spread, 1))

# calcolo della cdf per ogni valore della sequenza t
prob = ppois(t, lamda)

X = rep(0, Nsim)
for(i in 1:Nsim){
        u = runif(1)
        X[i] = t[1] + sum(prob < u) - 1
}
```

unibo/terzo-anno/Computational Statistic/README.md

📚 Computational Statistics

File (15)	tags	creation	Mentions
Monte Carlo Integration	• computational-statistic • integration	02_12_2024-10:25	• MASS • 2025-01-15 • 2025-01-14 • _TOC_
Importance Sampling	• computational-statistic	04_12_2024-12:19	• 2025-01-15 • _TOC_
prima lezione	• computational-statistic	11_11_2024-09:38	• _TOC_
PIT	• computational-statistic • random • variables	12_11_2024-09:35	• README • GTM • Discrete Distributions • 2025-01-02 • _TOC_
GTM	• computational-statistic	14_11_2024-12:10	• README • _TOC_

File (15)	tags	creation	Mentions
Discrete Distributions	• computational-statistic	18_11_2024-09:26	• README • _TOC_
AR Algorithm	• computational-statistic • accept-reject	19_11_2024-10:13	• newton-raphson • README • 2025-01-16 • 2025-01-14 • 2025-01-13 • 2025-01-10 • _TOC_
stochastic-search	• computational-statistic	2024_12_10-10:17	• _TOC_
Poisson Ditribution	• computational-statistic	2024-12-06 16:41	• README • _TOC_
mc-optimization	• computational-statistic	2024-12-09 09:41	• stochastic-search • simulating-annealing • newton-raphson • 2025-01-16 • _TOC_
newton-raphson	• computational-statistic	2024-12-10 10:19	• stochastic-search • mc-optimization • _TOC_
simulating-annealing	• computational-statistic • metallurgic • iterative	2024-12-10 10:48	• _TOC_
r-notes	• computational-statistic	2025_01_02-10:33	• _TOC_

File (15)	tags	creation	Mentions
acf	• computational-statistic	2025_01_02-11:54	• README • _TOC_
MASS	• computational-statistic	2025_01_21-22:28	• _TOC_

flow of study

- [ ] acf
- [x] ~~PIT & GTM (not requested for exam)~~
- [x] ~~Discrete Distributions Poisson Ditribution~~
- [ ] AR Algorithm (learn it)
- [ ]

unibo/terzo-anno/Computational Statistic/acf.md

extracted-from:: README

🏛️ (Auto)Corellogramma

Correlazione

indice-pearson

se una variabile cresce, allora cresce anche un altra.

$-1 \leq r \leq 1 \quad r \neq 0$

- $|r| = 1$ grande correlazione
- 0.8 ok
- 0.3 bassa
- 0 nessuna, random

Grafico

o serie storica.

insieme di valori che *dato un fenomeno* assume in successivi istanti di tempo.

- lag 1: correlazione misurata tra un dato e il suo subito precedente.
- lag 2: y_t and y_{t-2}
- cosi' via...

devo controllare che il grafico rimanga sotto la stima massima e minima che mi impongo.

altrimenti ottengo un **trend**.

Y t Y_t	Y t − 1 Y_{t-1}	Y t − 2 Y_{t-2}	Y t − 3 Y_{t-3}	… $\cdots$	Y t − K Y_{t-K}
Y 1 Y_1					
Y 2 Y_2	Y 1 Y_1				
Y 3 Y_3	Y 2 Y_2	Y 1 Y_1			
Y 4 Y_4	Y 3 Y_3	Y 2 Y_2	Y 1 Y_1		
⋮ $\vdots$	⋮ $\vdots$	⋮ $\vdots$	⋮ $\vdots$	⋮ $\vdots$	⋮ $\vdots$
Y T − 2 Y_{T-2}	Y T − 3 Y_{T-3}	Y T − 4 Y_{T-4}	Y T − 5 Y_{T-5}	⋮ $\vdots$	Y T − K − 2 Y_{T-K-2}
Y T − 1 Y_{T-1}	Y T − 2 Y_{T-2}	Y T − 3 Y_{T-3}	Y T − 4 Y_{T-4}	⋮ $\vdots$	Y T − K − 1 Y_{T-K-1}
Y T Y_T	Y T − 1 Y_{T-1}	Y T − 2 Y_{T-2}	Y T − 3 Y_{T-3}	⋮ $\vdots$	Y T − K Y_{T-K}

$$r_k = \frac{\sum_{t=K+1}^{T}(Y_t - \bar{Y})(Y_{t-k} - \bar{Y})}{\sum_{t=K+1}^{T}(Y_t - \bar{Y})^2}$$

unibo/terzo-anno/Computational Statistic/mc-optimization.md

extracted-from::README

Monte Carlo Optimization

where \ how	numerical	stochastic
global	indipendent sampling	stochastic global search
local		

max vs **arg max**.

what's the value such that ... ⇒ maximum likelihood.

monotonic transformation are useful! They do not change the maximum of the give function.

$$\max(h(\theta)) \qquad \theta \in \mathbb{R}^p$$

Deterministic

depends on function properties:

- convexity
- boundedness
- smoothness

computational problem

while dealing with really small number, we can have underflow problems, messing around the plot.

Stochastic

reproducible only by the seed.

if h is **complex** or the Θ (domain) is **irregular**

no underflow problem → we are not using products but instead sums.

we also use some 'noise' $-\sin(y * 100)^2$ because our data are not perfect → so it works as a stabilizer.

R functions

- `optim`
- `nlm`

Optimizing walk

unibo/terzo-anno/Computational Statistic/newton-raphson.md

extracted-from::README mc-optimization

Newton-Raphson method

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \left[\frac{\partial^2 h}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T}(\boldsymbol{\theta}_i)\right]^{-1} \frac{\partial h}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}_i).$$

https://www.youtube.com/watch?v=YSl37OYMLFw

iterative approach.

following the derivative

when to use it:

- h is quadratic
  problems:
- deteriorate when h is highly nonlinear
- **depends** on the starting point

using the fist derivative to find the direction in witch you want to *move* and the matrix of the second derivative to find the *step*.

if the domain of the function is not easy to compute, we can approximate it with a stochastic approach.

example 5.2

$$\mathcal{L}(\mu_1, \mu_2, x) = \frac{1}{4}\mathcal{N}(\mu_1, 1) + \frac{3}{4}\mathcal{N}(\mu_2, 1)$$

```
#data: mixture of two normals
da <- sample(rbind(rnorm(10^2),2.5+rnorm(3*10^2)))
```

```
#minus the log-likelihood function
# mu is a VECTOR
like <- function(mu){
  sum(log((.25*dnorm(da-mu[1])+.75*dnorm(da-mu[2]))))
}
```

Note

lable switching problem

```
#log-likelihood surface
mu1 <- mu2 <- seq(-2,5,le=250)
lli <- matrix(0,nco=250,nro=250)

for (i in 1:250){
  for (j in 1:250){
    lli[i,j]=like(c(mu1[i],mu2[j]))
  }
}

par(mar=c(4,4,1,1))
image(mu1,mu2,lli,
      xlab=expression(mu[1]),ylab=expression(mu[2]),
      col=hcl.colors(30))
contour(mu1,mu2,lli,nle=100,add=T)
```

cumulative maximum

vector containing the maximum till the index i.

Multivariate optimization

if we are working on larger spaces:

$$h(\theta) \qquad \theta \in \Theta = R^p$$

$$\text{if } \int_\Theta h(x)\, dx \text{ is finite} \qquad h(\theta) \geq 0, \forall \theta in \Theta$$

the function may be a **density**, so we start looking for the mode of the *density*!

now we can you a random sampling using AR Algorithm to find the mode of the density.

didn't we tried to find the mode of the density for not using a uniform distribution for the sampling?

It's a mater of shifting responsibility.

Acceptable solution

let's transform the function.

> **Note**
>
> - $\exp$ is a monotonic function not altering the maximum(s) of the function.
> - $T > 0$ is the temperature of the system aka make the plot smooth or rough.

$$H(\Theta) \propto \exp\left(\frac{h(\theta)}{T}\right)$$

unibo/terzo-anno/Computational Statistic/prima lezione.md

What to know

linear regression. Definition of probability. CDF. Mass function. Random Variable.

unibo/terzo-anno/Computational Statistic/r-notes.md

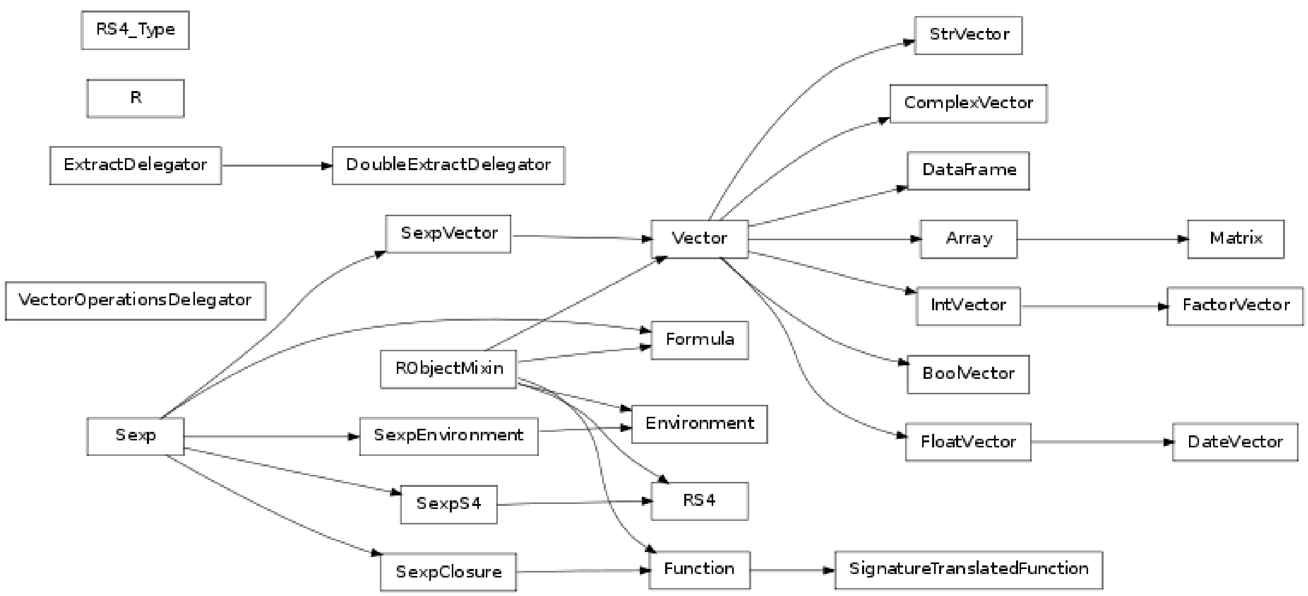
extracted-from:: README

💻 R language

`demo(image/graphics)`

Mode vs class

https://stackoverflow.com/questions/35445112/what-is-the-difference-between-mode-and-class-in-r



Graphs

```
hist(x)
plot(x1,x2)
acf(x)
```

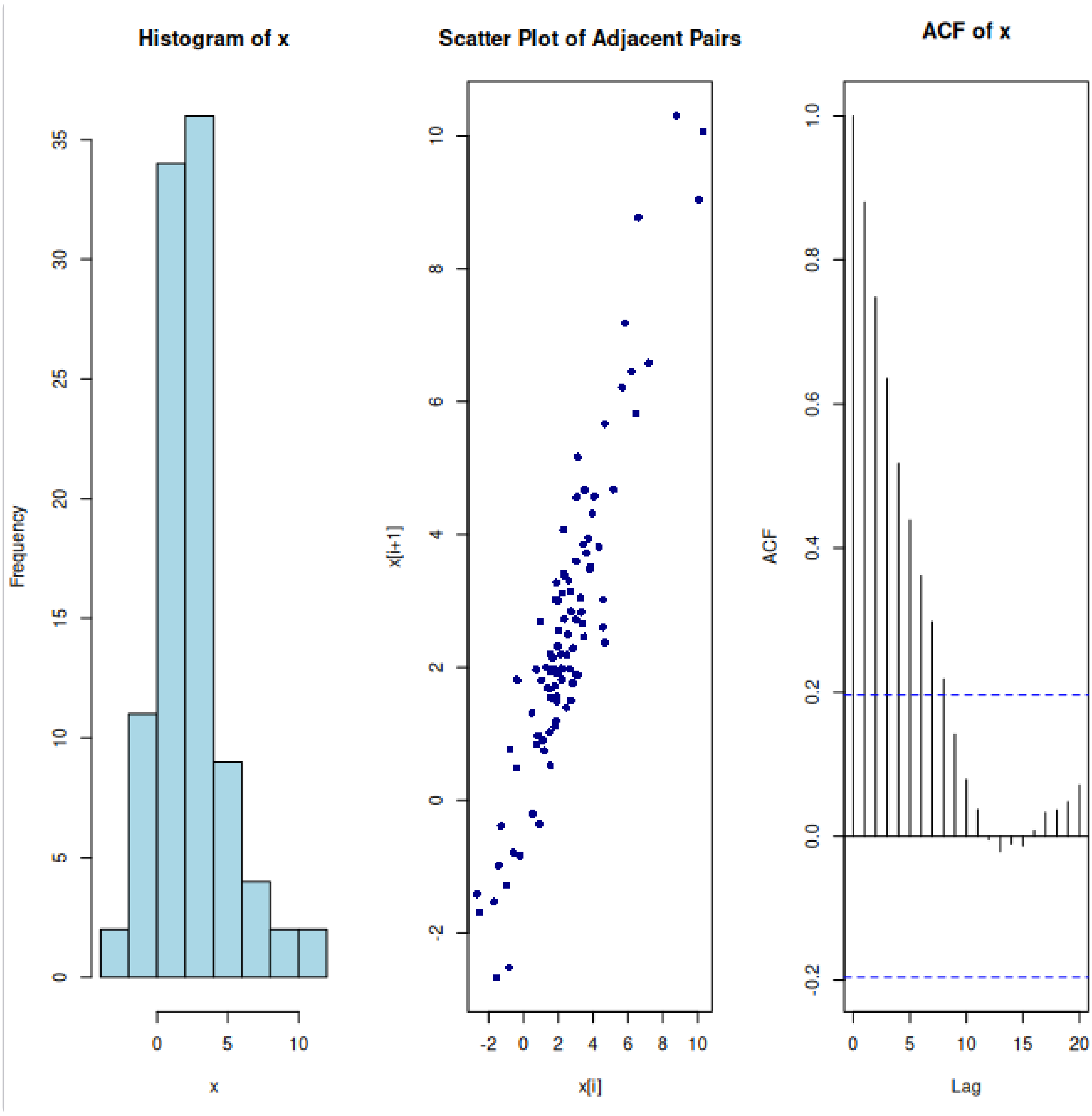
Graph	Focus	Insight
`hist(x)`	Distribution of values	Shape of the data (e.g., uniform, normal, skewed).
`plot(x1, x2)`	Relationship between neighbors	Patterns or clustering in adjacent values.
`acf(x)`	Correlation at different lags	Strength of relationships across time/sequential lags.

Correlated vs uncorrelated data

```r
# Generate correlated data
set.seed(123) # For reproducibility
Nsim <- 100
x <- cumsum(rnorm(Nsim, mean = 0, sd = 1)) # Random walk (cumulative sum of random noise)

# Create adjacent pairs
x1 <- x[-Nsim]
x2 <- x[-1]

# Plot the graphs
par(mfrow = c(1, 3)) # Arrange plots in a row
hist(x, main = "Histogram of x", col = "lightblue")
plot(x1, x2, main = "Scatter Plot of Adjacent Pairs", xlab = "x[i]", ylab = "x[i+1]", pch = 16, col = "darkblue")
acf(x, main = "ACF of x")
```




```
Nsim=10^4
x=runif(Nsim)
x1=x[-Nsim] #vectors to plot
x2=x[-1] #adjacent pairs

par(mfrow=c(1,3))
hist(x)
plot(x1,x2)
acf(x)
```

unibo/terzo-anno/Computational Statistic/simulating-annealing.md

extracted-from::README mc-optimization

🌀 Simulating Annealing

we change at each iteration the temperature T of the distribution.

example of hot metal ⇒ hot, spread out, as it cools off, getting smaller

the final distribution should have the same maximum as the target function.

$$\theta_{t+1} = \begin{cases} \theta_t + \varsigma & \text{with probability } \rho \\ \theta_t & \text{with probability } 1-\rho \end{cases} \quad \text{where} \quad \rho = \exp(\nabla h / T) \wedge 1$$

i can decide **not** to **move** with some probability.

random injection:

- sequence of distribution

unibo/terzo-anno/Computational Statistic/stochastic-search.md

extracted-from::README mc-optimization

🔦 Stochastic search

still searching inside teta.

taking the same idea as newton-raphson of a walk, but by applying a random step.

$$\theta_{j+1} = \theta_j + \epsilon_j$$

more formally

$$\theta_{j+1} = \theta_j + \alpha_j \nabla(h_j), \alpha_j > 0$$

calcolo is back discesa del gradiente

stochastic variation, remember the definition of *derivative*.

$$||\varsigma|| = 1$$

the perturbation can bring you into the **non optimal** direction.