Monte Carlo Sampling

G. Battistoni
INFN Milano

Overview:

More on sampling techniques from distribution

- by rejection
- Change of variable
- Gaussian sampling

MC statistical analysis

- Integral Estimators and Statistical errors
- Variance reduction techniques

Sampling from a distribution: the rejection technique

The rejection technique

- Some distributions cannot be easily sampled by integration and inversion.
- Let $f'(x)$ be one such distribution (normalized) that we want to sample
- Let $g'(x)$ be another normalized distribution function that can be sampled, such that $Cg'(x) \geq f'(x)$, for all $x \in [x_{min}, x_{max}]$
- Generate a uniform pseudo-random number $\xi_1 \in [0,1)$ to sample X from $g'(x)$
- Generate a second pseudo-random number ξ_2
- Accept X as a sample of $f'(x)$ if $\xi_2 < f'(X)/Cg'(x)$, otherwise re-sample ξ_1 and ξ_2

"Hit-Or-Miss" method

Sampling with the rejection technique

- The probability of X to be sampled from g′(x) is g′(X), the one that ξ_2 passes the test is f′(X)/Cg′(X) : therefore the probability to have X sampled and accepted is the product of probabilities g′(X) f′(X)/Cg′(X) = f′(X)/C
- The overall efficiency (probability accepted/rejected) is given by

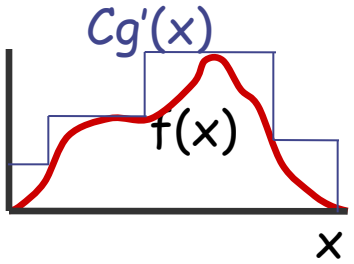
$$\varepsilon = \int \frac{f'(x)}{Cg'(x)} g'(x)dx = \int \frac{f'(x)}{C} dx = \frac{1}{C}\int f'(x)\,dx = \frac{1}{C}$$

f′(x) is normalized

- Proof that the sampling is unbiased (i.e. X is a correct sample from f′(x)): the probability P(X) dx of sampling X is given by:

$$P(X)\,dX = \frac{1}{\varepsilon} g'(X)\,\frac{f'(X)}{Cg'(X)}\,dX = f'(X)dX$$

- g′(X) is generally chosen as a uniform (rectangular) distribution or a normalized sum of uniform distributions (a piecewise constant function)



The "hit-or-miss" technique: example

Let be $f'(x) = (1+x^2)$, $x \in [-1,1]$

We choose $g'(x)$ to be constant, and:

$Cg'(x) = \max(f'(x)) = 2$

To normalize it:

$$\int_{-1}^{1} g'(x)\,dx = 1 \Rightarrow 2\,g'(x) = 1 \Rightarrow g'(x) = \frac{1}{2}$$

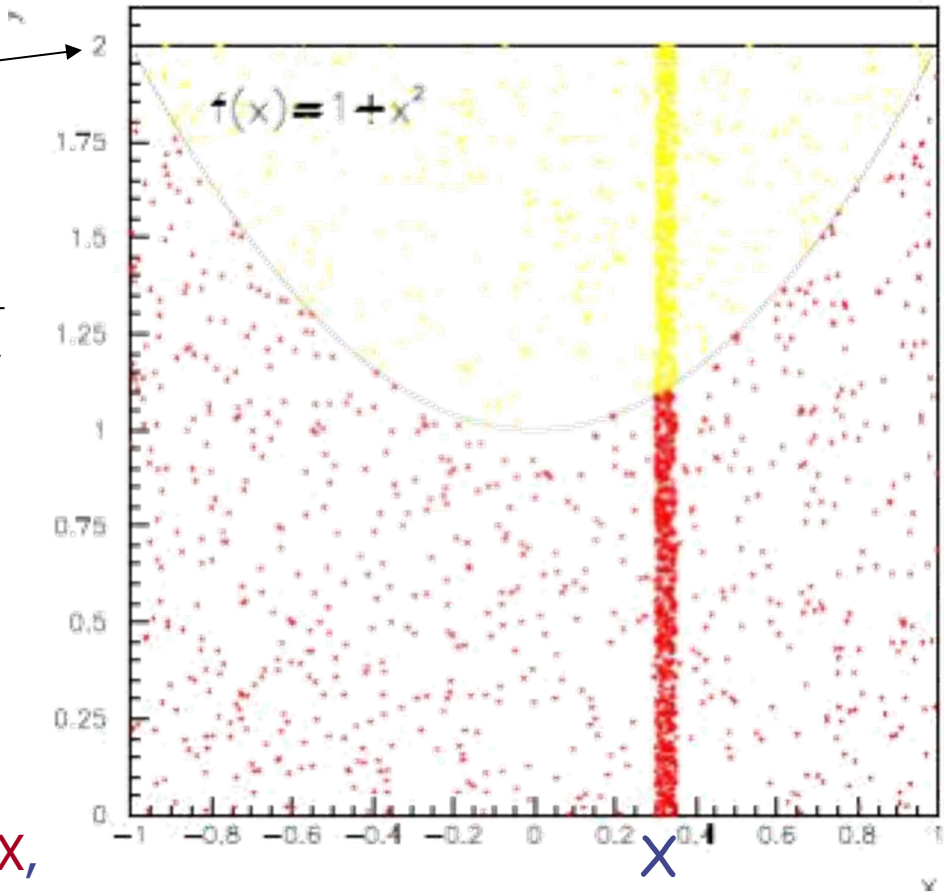
We obtain $C = 2/g'(x) = 4$

Generate two uniform pseudo-random numbers $\xi_1, \xi_2 \in [0,1)$

Sample X uniformly: $X = -1 + 2\xi_1$

Test:

if $(1+X^2)/Cg'(x) = (1+X^2)/2 > \xi_2$, accept X, otherwise re-sample ξ_1, ξ_2



The efficiency is the ratio of the red area to the total

A general recipe:

You can sample whichever pdf or distribution (think for instance to a set of histogrammed data), however complicated, by building the cumulative function F(x).

Once you have F(x) you can apply the same algorithm shown for the case of the binomial distribution

Densità di probabilità e trasformazioni di variabili

Cambiamento di variabile: $y = f(x)$ ← monotona!

Per la conservazione della probabilità deve essere:

$$p(y)dy = q(x)dx$$

Considerando la trasformazione inversa: $x = v(y) \equiv f^{-1}$

$$p(y)dy = q(v\,[y])dx]$$

$$p(y) = q(v\,[y])\left|\frac{dx}{dy}\right| \quad \rightarrow \quad p(y) = q(v\,[y])|v'[y]|$$

Se uno sa campionare un valore x da q(x) allora
→ **y si ottiene semplicemente imponendo y = f(x)**

Esempio: Partiamo da una varabile x distribuita secondo un'esponenziale:

$q(x)=e^{-x}\ (0<x<\infty)$ Funzione che sappiamo campionare esattamente:

$$x_i=-\log\xi_i$$

Vogliamo passare alla variabile: $y=\sqrt{x}\ (0<x<\infty)$

Per quanto detto prima: $y_i=\sqrt{x_i}$ ⇒ $y_i=\sqrt{-log\xi_i}$

Questo risultato ha una importanza più generale: quale è la *p(y)dy*, cioè la pdf di $y=\sqrt{x}$?

$$p(y) = q(v\,[y])|v'[y]| = \frac{q(x[y])}{\left|\frac{dy}{dx}\right|}$$

E' già automaticamente normalizzata!

$$x = y^2$$
$$y = \sqrt{x}$$

$$p(y) = \frac{q\left(x[y]\right)}{\left|\frac{dy}{dx}\right|} = \frac{e^{-y^2}}{\left|\frac{1}{2\sqrt{x}}\right|} = \frac{e^{-y^2}}{\left|\frac{1}{2y}\right|} = 2y\,e^{-y^2}$$

Quindi l'algoritmo derivato precedentemente ci permette di effettuare una campionatura random esatta della funzione: $f(x) = 2\,x\,e^{-x^2}$

$$x_i = \sqrt{-log\xi_i}$$

Evitando altri metodi inefficenti, quali per esempio l' hit-or-miss

Estensione ad un caso più generale:

$$p(y) = \frac{2}{\Gamma(n)} y^{2n-1} e^{-y^2} \quad n = 1,2,\ldots$$

$$\Gamma(t) = \int_0^\infty x^{t-1} e^{-x}\, dx \qquad \Gamma(n) = (n-1)! \qquad \text{(funzione Gamma)}$$

$$\Rightarrow \quad y_i = \left(-\log \prod_{i=1}^{n} \xi_i \right)^{\frac{1}{2}}$$

Esercizio: dimostrare tale generalizzazione…

Trasf. di variabili nel caso multidimensionale:

Da un insieme di variabili $\{x_i, x_j, \ldots, x_n\}$ distribute con la pdf $q(x_i, x_j, \ldots, x_n)dx_i\, dx_j \ldots dx_n$ vogliamo passare alle variabili $\{y_i, y_j, \ldots, y_n\} = f[x_i, x_j, \ldots, x_n]$

Queste saranno distribuite secondo la pdf $p(y_i, y_j, \ldots, y_n)dy_i\, dy_j \ldots dy_n$ ottenibile come:

$$p(f[x_i, x_j, \ldots, x_n])\left|\frac{dy_i}{dx_j}\right| dx_i dx_j \ldots dx_n = q(x_i, x_j, \ldots, x_n)\, dx_i dx_j \ldots dx_n$$

Determinante Jacobiano della trasf. di variabili

Come sfruttare i risultati precedenti per fare un generatore di numeri random gaussiani

Consideriamo due semi-gaussiane (cioè definite solo per x>0) indipendenti. Essendo indipendenti $p(x_1,x_2) = p(x_1)*p(x_2)$

$$p(\upsilon_1,\upsilon_2) = \frac{2}{\sqrt{\pi}} e^{-\upsilon_1^2} \frac{2}{\sqrt{\pi}} e^{-\upsilon_2^2}$$

Introduciamo una trasformazioni di variabili in coordinate polari:

$$\upsilon_1 = R\cos\theta;\ \upsilon_2 = R\sin\theta$$

$$J = \begin{vmatrix} \frac{\partial \upsilon_1}{\partial R} & \frac{\partial \upsilon_1}{\partial \theta} \\ \frac{\partial \upsilon_2}{\partial R} & \frac{\partial \upsilon_2}{\partial \theta} \end{vmatrix} = \begin{vmatrix} \cos\theta & -R\sin\theta \\ \sin\theta & R\cos\theta \end{vmatrix} = R$$

Applicando la trasformazione di variabili otteniamo la nuova epressione per la pdf in coordinate polari:

$$\frac{2}{\sqrt{\pi}} e^{-\upsilon_1^2} \frac{2}{\sqrt{\pi}} e^{-\upsilon_2^2} d\upsilon_1 d\upsilon_2 = \frac{4}{\pi} e^{-\upsilon_1^2 - \upsilon_2^2} d\upsilon_1 d\upsilon_2 =$$

$$= 2 R e^{-R^2} dR \left(\frac{2}{\pi} \right) d\theta$$

Ma questa è proprio la nuova funzione $y = 2 y \exp(-y^2)$ che abbiamo imparato a campionare a partire dal semplice caso dell'esponenziale!

Possiamo quindi arrivare a costruire un generatore ad efficienza 1 di due gaussiane indipendenti con la seguente procedura:

1. campioniamo indipendentemente R, con la prescrizione introdotta precedentemente, e θ uniformemente fra 0 e π/2
2. ricaviamo v_1 e v_2 dalla trasformazione inversa di variabili
3. Simmetriziamo l'intervallo di definizione di v_1 e v_2 da {0,∞} a {-∞,∞}: con una semplice estrazione random uniforme possiamo cambiare v_i in $-v_i$ con probabilità ½

Abbiamo cosi' ottenuto due numeri random v_1 e v_2 indipendenti distribuiti ciascuno come una gaussiana ($\mu=0$, $\sigma=1/\sqrt{2}$).

Sono necessarie 4 estrazioni random (1 per R, 1 per θ e 2 per la simmetrizzazione di v_1 e v_2; 3 estrazioni se ci interessa solo v_1)

Per ottenere una distribuzione gaussiana normale ($\mu=0$, $\sigma=1$) è sufficiente moltiplicare v_1 e/o v_2 per $\sqrt{2}$

$$g(x) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

A partire dalla gaussiana normale, per ottenere una distribuzione gaussiana con parametri μ, σ qualsiasi è sufficiente effettuare l'operazione:

$$v' = \mu + \sigma\, v$$

In realtà tutte le librerie matematiche dei vari linguaggi di programmazione hanno già implementato un generatore random gaussiano e non c'è bisogno che l'utente programmi esplicitamente tale generatore.

Esempi:

1) Root:

```
TRandom2 *rnd = new TRandom2();
....
// Estrazione uniforme in [0,1)
Dpuble_t x = rnd->Uniform(0,1);
// Estrazione gaussiana
Double_t mean = 0.;
Double_t sigma = 1.;
Double_t y = rnd->Gaus(mean,sigma);
```

2) Python

```
import random as rnd
....
#Estrazione uniforme in [0,1)
x = rnd.random()
#Estrazione gaussiana
mu = 0
sigma = 1
y = rnd.gauss(mu,sigma)
```

Python

Lista non esaustiva…

random.**random**() _____ Genera pseudorandom in (*0,1*)

random.**uniform**(*a, b*) _____ Genera pseudorandom in (*a,b*)

random.**triangular**(*low, high, mode*) ____ Distribuzione triangolare in (*low,hide*)

random.**expovariate**(*lambd=1.0*) _____ Distribuzione esponenziale

random.**gammavariate**(*alpha, beta*) _____ Distribuzione gamma

random.**normalvariate**(*mu=0.0, sigma=1.0*) _____ Distribuzione normal gaussiana

random.**gauss**(*mu=0.0, sigma=1.0*) _____ Distribuzione normal gaussiana
(più veloce di normalvariate)

random.**randint**(*a, b*) _____ Genera pseudorandom intero in (a,b)

random.**binomialvariate**(*n=1, p=0.5*) _____ Distribuzione binomiale

Ci sono altre possibilità usando l'estensione NumPy, comoda soprattutto per generare velocemente array o liste di numeri random

ROOT

Lista non esaustiva…

TRandom class (TRandom1, TRandom2, TRandom3)

Rndm () _____ Genera pseudorandom in (*0,1*)

Integer (UInt_t imax) _____ Genera pseudorandom intero in (*0,imax-1*)

Uniform (Double_t x1, Double_t x2) _____ Genera pseudorandom (*x1,x2*)

Exp (Double_t tau) _____ Distribuzione esponenziale

Gaus (Double_t mean=0, Double_t sigma=1) _ Distribuzione normal gaussiana

Rannor (Double_t &a, Double_t &b) _ coppia di pseudorandom normal-gaussiani

Landau (Double_t mean=0, Double_t sigma=1) _ Distribuzione di Landau

Sphere (Double_t &x, Double_t &y, Double_t &z, Double_t r)
_____ vettore 3d random di lunghezza r

Circle (Double_t &x, Double_t &y, Double_t r)
_____ **vettore 2d random di lunghezza r**

Poisson (Double_t mean) _____ Distribuzione di Poisson

Binomial (Int_t ntot, Double_t prob) _____ Distribuzione Binomiale

La riproducibilità:

Uso del "seed" per innescare una sequenza di numeri pseudorandom

ROOT: in TRandom

SetSeed (ULong_t seed=0)

GetSeed ()

Python:

random.**seed**(*a=None, version=2*)

random.**getstate()**

random.**setstate**(***state***)

L'algoritmo di Metropolis

Algoritmo per campionare da distribuzioni con integrale difficile, o impossibile da calcolare analiticamente

Il metodo si basa sulla generazione di una sequenza di numeri di 'test' che vengono accettati o rigettati in modo da ottenere la distribuzione voluta **P(x)**. Generalizzabile al caso di distribuzioni di probabilità $\mathbf{P(x_1, x_2, \ldots, x_N)}$ di un numero qualsiasi di variabili. Basato sulla teoria delle catene di Markov.

1) Preso l'ultimo valore $\mathbf{x_i}$ della variabile random nella sequenza si sceglie un valore di prova **x*** diverso da $\mathbf{x_i}$ tra tutti i valori possibili della variabile random.

2) Si calcola il rapporto $\mathbf{w = P(x^*)/P(x_i)}$

3) Se $\mathbf{w \geq 1}$ si accetta il nuovo valore $\mathbf{x^* = x_{i+1}}$
Se invece $\mathbf{w<1}$ il nuovo valore deve essere accettato con probabilità **w**:

- Si genera quindi un numero random **ξ** distribuito uniformemente nell'intervallo [0, 1);
- Se $\mathbf{\xi \leq w}$ si accetta il nuovo valore $\mathbf{x^* = x_{i+1}}$;
- Se invece $\mathbf{\xi > w}$ il nuovo valore viene rigettato

Per avere una buona stima della **P(x)** è necessario generare sequenze molto lunghe

Chiaramemente è un metodo conveniente solo per campionare distribuzioni particolarmente complesse

Stimatori Monte Carlo dell'Integrale

- Vogliamo calcolare l'integrale definito

$$I = \int_a^b f(x)\,dx$$

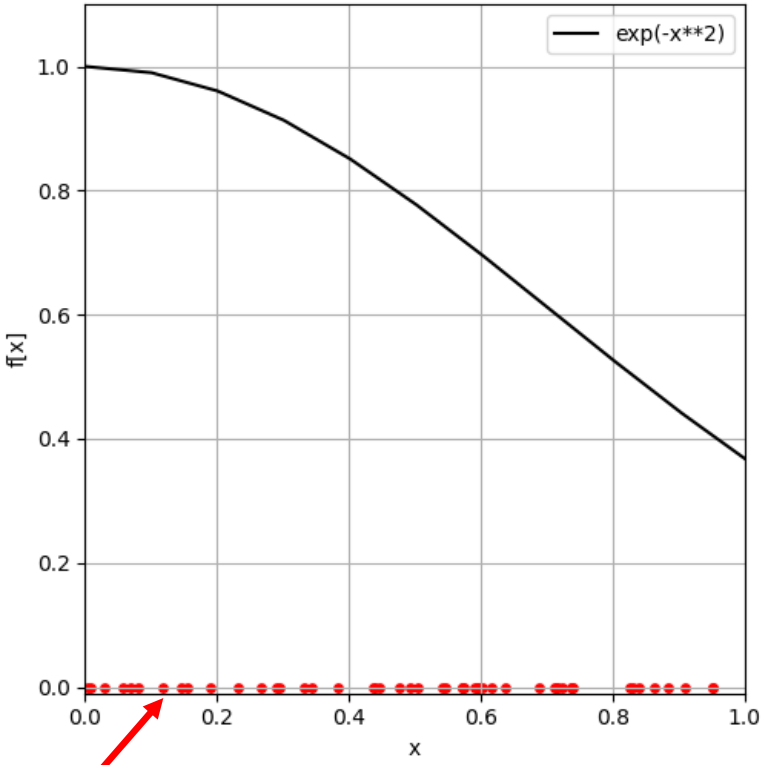
- Consideriamo una sequenza di N punti casuali uniformemente distribuiti nell'intervallo [a,b] (**Uniform Sampling**). Il metodo Monte Carlo (MC) stima un valore approssimato dell'integrale utilizzando lo stimatore media (aritmetica):

$$\hat{I} = (b-a)\frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

- Questo stimatore e' non distorto e consistente. Al crescere di N il valore stimato tende al valore vero dell'integrale.

Stimatori Monte Carlo dell'Integrale

Esempio: vogliamo calcolare il seguente integrale definito: $\int_0^1 e^{-x^2}\,dx$



$$\hat{I} = (b-a)\frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

Valore vero = 0.746824….

Con N = 10^5 campionamenti abbiamo un errore ~0.07%

N valori casuali di **x** <u>uniformemente</u> distribuiti nell'intervallo [0,1] dove vengono calcolati i valori $f(x_i)$

Stimatori Monte Carlo dell'Integrale

- La varianza dello stimatore della media è $\sigma_{\bar{f}}^2 = \frac{\langle f^2 \rangle - \langle f \rangle^2}{N}$
- con $\langle f \rangle = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \quad \langle f^2 \rangle = \frac{1}{N}\sum_{i=1}^{N} f^2(x_i)$

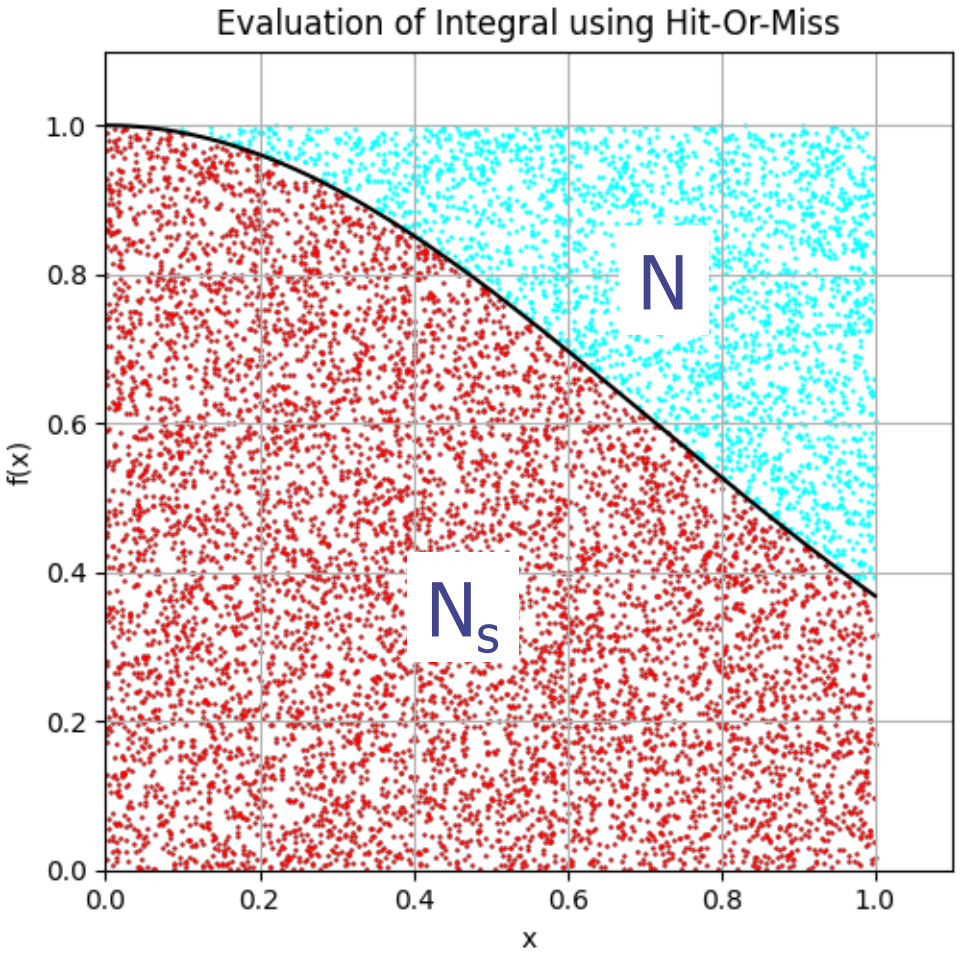
$\frac{\sigma_{\bar{f}}}{\sqrt{N}}$ e' detto **errore standard della media** . Quindi tenendo conto dell'errore il valore dell'integrale stimato dal metodo MC e':

$$\hat{I} \approx (b-a)\bar{f} \pm (b-a)\frac{\sigma_f}{\sqrt{N}}$$

- Se volessimo calcolare un integrale di volume, il valore approssimato sarebbe: $\hat{I} \approx V\bar{f} \pm V\frac{\sigma_f}{\sqrt{N}}$
- Notiamo che indipendentemente dalle dimensioni dello spazio di integrazione l'errore scala come $\frac{1}{\sqrt{N}}$

Integrazione MC con "hit-or-miss"

- Consideriamo l'integrazione di f(x) unidimensionale. Sia N il numero di punti considerati e N_s quelli tra questi N in cui y_i e' minore o uguale a $f(x_i)$. Il valore approssimato dell'integrale e' $I_n \propto N_s / N$



Di nuovo l'esempio $\int_0^1 e^{-x^2}\,dx$

Valore vero = 0.746824….

Con $N = 10^5$ campionamenti abbiamo un errore ~0.3%

Integrazione MC con "hit-or-miss"

- L'errore standard su questa valutazione è:

$$\sigma_f = \sqrt{\frac{(N_S/N) - (N_S/N)^2}{N}} = \sqrt{\frac{N_S - N_S^2/N}{N^2}} = \frac{\sqrt{N_S - N_S^2/N}}{N}$$

$$\hat{I} \propto \frac{N_S}{N} \pm \frac{\sqrt{N_S - N_S^2/N}}{N}$$

- Se $N_S << N \rightarrow$

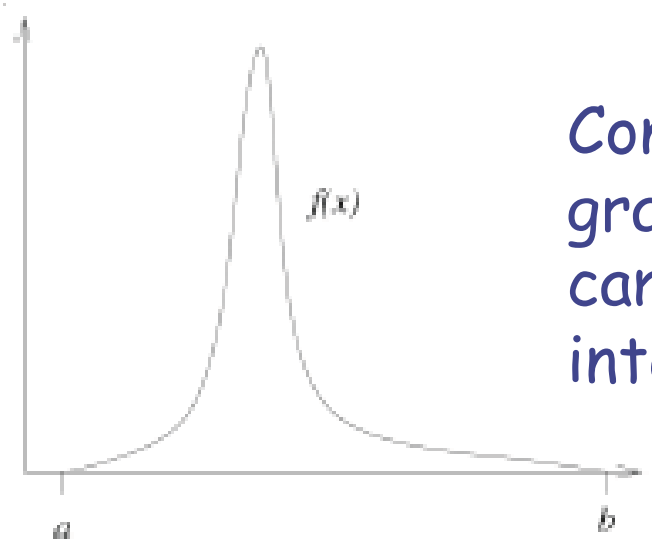
$$\hat{I} \propto \frac{N_S}{N} \pm \frac{\sqrt{N_S}}{N}$$

(Ns variabile poissoniana)

- Tutto cio' vale indipendentemente dal numero di dimensioni dello spazio di integrazopme

Riduzione della Varianza

- Il vantaggio enorme dell'integrazione MC e' che scala come $1/\sqrt{N}$ indipendentemente dalle dimensioni dell'integrale
- La velocita' di convergenza e' relativamente lenta pero'. Ci sono diversi metodi per ovviare a cio' e diminuire l'errore sulla stima.
- Questo puo' esse fatto diminuendo la varianza su f



Con campionamento uniforme, una gran parte dei punti di campionamento sara' fuori dall'area di integrazione.

Importance Sampling

- Le cose andrebbero molto meglio se la funzione da integrare fosse piu' piatta!!
- In questo caso la varianza diminuirebbe.
- Possiamo pensare di cambiare la variabile di integrazione in modo da integrare una funzione il piu' piatta possibile con la condizione che sappiamo poi tornare indietro per stimate l'integrale di partenza

Importance Sampling

- Si debba valcolare l'integrale definito della funzione f(x)

$$I = \int_a^b f(x)\,dx$$

- Consideriamo una "opportuna" funzione p(x) **a valori positivi** e normalizzata ad 1 tra a e b

$$\int_a^b p(x)\,dx = 1$$

- In questo modo la funzione p(x) puo' essere pensata come una pdf e il nostro integrale (con un cambiamento di variabile) puo' essere riscritto cosi:

$$I = \int_a^b \frac{f(x)}{p(x)}\left(p(x)\,dx\right)$$

- La stima MC di questo integrale e' semplicemente:

$$I = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$

- dove i punti di campionamento xi sono scelti secondo la pdf p(x)
- La funzione p(x) va scelta in modo opportuno. Se fosse piatta (distribuzione uniforme) allora p(x) = 1/(b-a) e si otterrebbe il risultato gia' visto del campionamento uniforme.
- L'incertezza statistica sul calcolo dell'integrale e'

$$\frac{\sigma_{f/p}}{\sqrt{N}}$$

- Deviazione standard di f(x)/p(x):

$$\sigma_{f/p} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{f(x_i)}{p(x_i)}\right)^2 - I^2}$$

- La funzione p(x) deve essere il piu' possibile vicina a |f(x)| nella forma in modo che il rapporto f(x)/p(x) sia il piu' piatto possibile. E' opportuna una scelta di p(x) per la quale sia semplice simulare punti secondo la p(x) e a basso costo computazionale.
- La cosa quasi ovvia sarebbe di prendere p(x) proporzionale a f(x) (se questa e' a valori positivi) **p(x) = C f(x)** ma questo non e' possibile. Essendo p(x) una pdf allora si dovrebbe avere p(x) = f(x)/I
- Notiamo che noi non conosciamo I (altrimenti non avremmo bisogno di alcuna integrazione!) . Di conseguenza non possiamo campionare punti secondo f(x)/I

Esempio

- Usando ancora l'esempio precedente calcoliamo con questo metodo l'integrale

$$\int_0^1 e^{-x^2}\,dx$$

- La funzione integranda per x da 0 ad 1 decresce da 1 ad 1/e proprio come la funzione $\mathbf{e^{-x}}$. Quindi scelgo questa come funzione p(x). Normalizziamo ad 1 questa funzione:

$$\int_0^1 e^{-x}\,dx = \frac{e-1}{e}$$

- p(x) normalizzata: $p(x) = \frac{e}{e-1} e^{-x}$

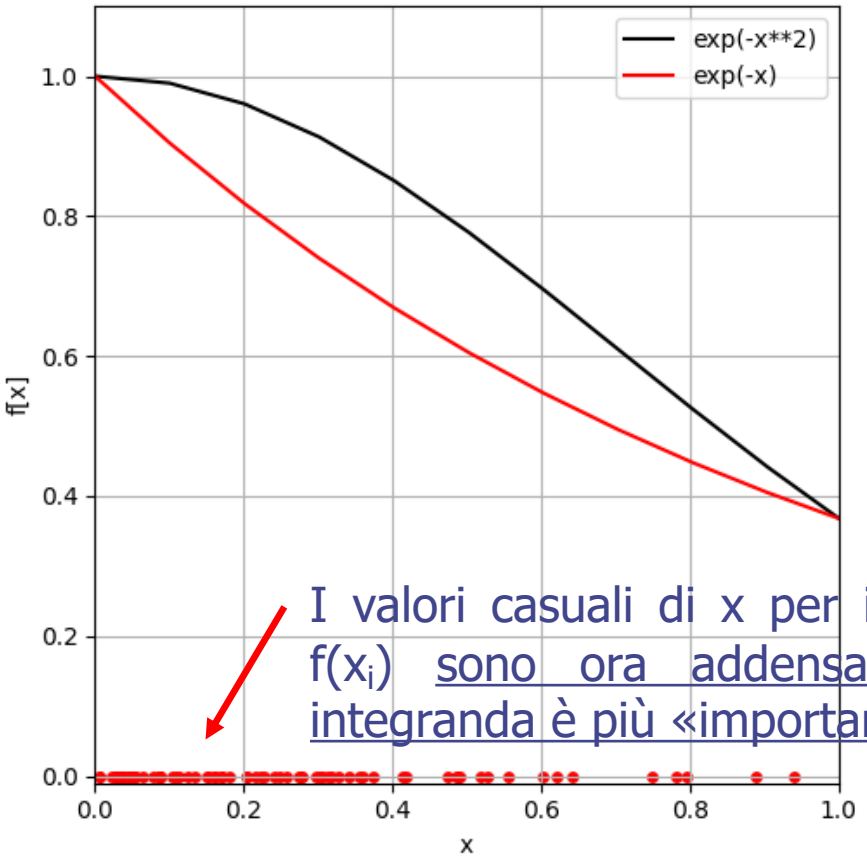
- Sappiamo campionare numeri casuali distribuiti secondo un esponenziale. In questo caso:

$$x(r) = -\log\left(1 - \frac{e-1}{e} r\right)$$

- con r distribuito uniformemente tra 0 e 1: x(r) risulterà distribuito, con andamento esponenziale decrescente, nell'intervallo di valori compreso fra 1 e 1/e

Esempio

- Quindi si generano **N** numeri x_i di questa sequenza e per ognuno di questi si calcolano $f(x_i)$ e $p(x_i)$ e quindi si usano questi valori nella stima MC dell'integrale.



$$I = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$

Valore vero = 0.746824….

Con N = 10^5 campionamenti abbiamo un errore ~0.009%

I valori casuali di x per il calcolo dei valori di $f(x_i)$ sono ora addensati deve la funzione integranda è più «importante»

Ovviamente, per questa funzione di esempio, non molto ripida, il metodo dell'importance sampling non presenta ancora vantaggi decisivi

Considerazioni sull'uso di questo metodo

- Se è il caso, per generare numeri secondo la pdf p(x) si puo' usare il metodo hit or miss.
- Comunque e' chiaro cha la p(x) va scelta anche sulla base che sia semplice e a basso costo computazionale la generazione di sequenze con questa pdf.
- Nella scelta della funzione p(x) bisogna tener conto dell'andamento delle code delle distribuzioni. Se per esempio la funzione p(x) va a zero molto piu' rapidamente della funzione f(x), valori della funzione f(x) lontano nelle code avranno un peso molto elevato nella determinazione del valore medio e questo crea una coda nella distribuzione dei valori dell'integrale.
- Per lo stesso motivo bisogna evitare funzioni p(x) che in qualche punto sono nulle o tendono rapidamente a zero.

Integration efficiency

- Traditional numerical integration methods (e.g., Simpson) converge to the true value as $N^{-1/n}$, where N = number of "points" (intervals) and n = number of dimensions
- Monte Carlo converges as $N^{-1/2}$, independent of the number of dimensions
- Therefore:
  - $n = 1$ ➔ MC is not convenient
  - $n = 2$ ➔ MC is about equivalent to traditional methods
  - $n > 2$ ➔ MC converges faster (and the more so the greater the dimensions)
- With the integro-differential Boltzmann equation the dimensions are the 7 of phase space, but we use the integral form: the dimensions are those of the largest number of "collisions" per history (the Neumann term of highest order)
- Note that the term "collision" comes from low-energy neutron/photon transport theory. Here it should be understood in the extended meaning of "interaction where the particle changes its direction and/or energy, or produces new particles"