Università degli Studi di Padova

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria dell'Informazione

Il metodo Monte Carlo e alcune sue applicazioni nell'ambito dell'Ingegneria dell'Informazione

Relatore: Prof. Alessandro Chiuso

Laureanda: *Francesca Zorzi*

Anno Accademico 2023/2024
Data di Laurea: 25-09-2024

Abstract

Il seguente elaborato ha come obiettivo quello di analizzare il metodo Monte Carlo e di studiarne le sue proprietà mostrando come possa essere utilizzato nell'ambito specifico dell'Ingegneria dell'informazione.

In una prima parte vi è l'introduzione al problema che porta alla formulazione del metodo in cui viene fatto l'esempio della stima bayesiana per mettere in luce le motivazioni che portano a dover utilizzare metodi Monte Carlo in alcune applicazioni.

Successivamente viene fornito il supporto teorico che sta alla base del metodo e ne garantisce l'applicabilità. Dalla teoria però emerge l'importanza della generazione di campioni indipendenti da distribuzioni di probabilità arbitrarie, in particolare nel secondo capitolo si mostrano alcuni generatori semplici da varibili aleatorie uniformi, che formano gli strumenti di base per algoritmi di generazione da distribuzioni generiche. Uno dei quali, basato sull'utilizzo delle Markov Chain, è l'algoritmo Metropolis-Hastings che viene presentato nel terzo capitolo. Ne viene poi mostrato un esempio implementativo in MATLAB.

Indice

Capitolo 1

Introduzione al metodo Monte Carlo

Il metodo Monte Carlo ha avuto origine all'inizio degli anni '40 nella città di Los Alamos, sede del programma di ricerca statunitense dedicato principalmente allo sviluppo della bomba atomica che prese il nome di progetto Manhattan.
Il nome "Monte Carlo" venne suggerito dal matematico e fisico polacco Stanislaw Ulam ispirandosi all'omonimo casinò, data la sua natura probabilistica e casuale che ricorda i giochi d'azzardo.

Il metodo Monte Carlo consiste nell'utilizzo di simulazioni numeriche basate su campionamenti casuali per risolvere problemi matematici complessi, spesso inaccessibili tramite metodi analitici tradizionali. Durante la Seconda Guerra Mondiale questo approccio fu impiegato per risolvere problemi relativi al comportamento dei neutroni in materiali fissionabili. L'idea chiave era quella di rappresentare processi fisici mediante sequenze di numeri casuali e poi utilizzare tali simulazioni per stimare soluzioni numeriche di problemi deterministici.

Il metodo si basa su tre passaggi fondamentali: La definizione del problema in termini probabilistici, la generazione di campioni casuali secondo la distribuzione definita e l'analisi statistica dei risultati per ottenere una stima delle quantità di interesse.
L'importanza del metodo Monte Carlo risiede nelle sue caratteristiche: prime fra tutte la semplicità e l'efficienza che rendono il metodo flessibile, scalabile e veloce, e insieme all'aleatorietà rendono il metodo Monte Carlo applicabile a una vasta gamma di problemi: dai sistemi fisici alle computazioni numeriche deterministi-

che, fornendo soluzioni approssimative a situazioni in cui altre tecniche falliscono. In particolare fu molto utile nel contesto dei calcoli riguardanti le reazioni nucleari, dove le interazioni complesse tra le particelle potevano essere modellate e studiate in modo efficiente attraverso simulazioni casuali [3] [8].

1.1 Introduzione al problema e limiti del calcolo esatto

In numerosi settori dell'ingegneria, della fisica, dell'economia e di altre discipline, si presenta spesso la necessità di risolvere integrali per affrontare una vasta gamma di problemi. La forma più generica con cui si può descrivere l'integrale che sarà poi di interesse in seguito è la seguente

$$\int_a^b f(x)g(x)dx \tag{1.1}$$

Esempio 1.1.1 (Stima Bayesiana)
Per introdurre il problema è utile utilizzare un esempio significativo nel campo dell'ingegneria dell'informazione che riguarda la stima bayesiana. Quando si affronta un problema di stima l'obiettivo è quello di determinare il parametro migliore all'interno di una famiglia che descriva al meglio i dati osservati.
L'aspetto caratterizzante della stima bayesiana è che il parametro preso in esame viene considerato come una variabile stocastica, cioè associato a una distribuzione di probabilità a priori. Quando si risolve un problema di stima di questo tipo, viene utilizzato uno stimatore che minimizza il valore atteso della funzione di costo in relazione alla distribuzione di probabilità a posteriori [12].
Sia $\mathbf{x}$ un vettore aleatorio con una sua distribuzione $p(\mathbf{x})$, l'approccio di Bayes è quello di ottenere uno stimatore che cerchi il parametro che risulti essere più probabile tenendo conto sia della conoscenza pregressa (la probablilità a priori), sia delle nuove informazioni ottenute dai dati (la probabilità a posteriori).

Dalla densità di probabilità a posteriori $p_{x|y}(\mathbf{x}|\mathbf{y})$, si possono ottenere vari stimatori, come lo stimatore a minima varianza

$$\hat{\mathbf{x}} = \mathbb{E}[\mathbf{x}|\mathbf{y}] = \int \mathbf{x}\, p_{x|y}(\mathbf{x}|\mathbf{y})dx \tag{1.2}$$

oppure lo stimatore MAP (*Maximum a posteriori probability*)

$$\hat{\mathbf{x}} = \arg\max p_{x|y}(\mathbf{x}|\mathbf{y}) \tag{1.3}$$

In entrambi gli stimatori 1.2 1.3 si può ricavare una stima di x dalla densità di probabilità a posteriori $p_{x|y}(\mathbf{x}|\mathbf{y})$.
Rimane ora il problema del calcolo appunto di $p_{x|y}(\mathbf{x}|\mathbf{y})$ che con questo approccio si basa sulla ben nota formula di Bayes

$$p_{x|y}(\mathbf{x}|\mathbf{y}) = \frac{p_{y|x}(\mathbf{y}|\mathbf{x})p_x(\mathbf{x})}{p_y(\mathbf{y})} \tag{1.4}$$

Come emerge in 1.2 il calcolo dello stimatore a minima varianza implica il calcolo di un integrale. Non solo ma anche nella formula di Bayes 1.4 il calcolo di $p_y(\mathbf{y})$ implica lo stesso tipo di calcolo.

$$p_y(\mathbf{y}) = \int p_{y|x}(\mathbf{y}|\mathbf{x})p_x(\mathbf{x})dx \tag{1.5}$$

L'integrazione ha quindi un ruolo fondamentale nella stima bayesiana. Questo aspetto può non rappresentare un problema quando il vettore aleatorio assume valori in uno spazio di dimensioni ridotte, oppure quando il vettore aleatorio ha una distribuzione a priori che presenta particolari simmetrie, come nel caso della Normale, che quindi semplificano molto il calcolo. In tutti gli altri casi invece ci si ritrova a dover affrontare una problematica importante che prende il nome di **maledizione della dimensionalità** (*curse of dimensionality*). Quando lo spazio vettoriale è di grandi dimensioni, la conseguenza è che integrali come 1.2 - 1.5 risultino molto complessi da calcolare. Infatti per calcolare ogni integrale tramite Riemann va valutata la funzione in un numero di punti che è esponenziale nella dimensione dello spazio. Ciò comporta che in alte dimensioni questa valutazione non sia possibile e l'introduzione di metodi la cui complessità non dipenda dalla dimensione, come il Metodo Monte Carlo, è fondamentale.
L'Esempio del confronto tra il calcolo integrale risolto usando Riemann e con il metodo Monte Carlo si trova nella Sezione 1.5.2.

Prima di procedere con l'enunciato del metodo Monte Carlo bisogna riformulare l'integrale generico 1.1 ed esprimerlo come il valore atteso di una variabile aleatoria X, avente densità di probabilità $\pi(x)$ e sia inoltre $f : \mathbb{R} \to \mathbb{R}$ una funzione misurabile, allora $f(X)$ è una variabile aleatoria e il suo valore atteso è

[9]

$$\mathbb{E}[f(X)] := \int_{-\infty}^{+\infty} f(x)\pi(x)dx$$

π rappresenta una qualsiasi densità di probabilità della variabile aleatoria, che come visto può essere $p_{x|y}(\mathbf{x}|\mathbf{y})$ o $p_y(\mathbf{y})$, rispetto alla quale si sta considerando il valore atteso.

1.1.1 Enunciato del metodo Monte Carlo

Il metodo Monte Carlo consente di calcolare il valore atteso di una variabile aleatoria con una distribuzione qualsiasi, mediante l'approssimazione del risultato tramite simulazioni.

Sia X una variabile aleatoria e f una funzione misurabile, allora

$$\begin{aligned}\mathbb{E}[f(X)] &= \int_{-\infty}^{+\infty} f(x)\pi(x)dx \\ &\approx \frac{1}{n}\sum_{i=1}^{n} f(X_i) \qquad n \to \infty\end{aligned}$$

Dove X_i sono realizzazioni ***indipendenti*** *della v.a. X.*
Si generano quindi molti campioni casuali dalla distribuzione della v.a. e si utilizza la media di questi campioni come stima del valore atteso.

1.2 Fondamenta teoriche a supporto del metodo

Il metodo Monte Carlo funziona perché basato su una solida teoria: esso si basa sulla legge dei grandi numeri 1.2.2 che è un importante teorema limite del calcolo della probabilità.
Si tratta di una legge relativa alla convergenza in probabilità 1.2.1 delle medie aritmetiche di sequenze di v.a. indipendenti e identicamente distribuite (i.i.d.).
L'importanza risiede nel fatto di possedere uno strumento che fornisca una legge esatta che estrae ordine dal caos. [9].
Di seguito sono quindi riportate: la definizione di convergenza in probabilità e l'enunciato della legge dei grandi numeri.

1.2.1 Convergenza in probabilità

Definizione 1.2.1. La sequenza di variabili aleatorie $\{X_n\}$ converge in probabilità alla v.a. X se

$$\lim_{n\to\infty} P(|X_n - X| \geq \epsilon) = 0, \quad \text{per ogni } \epsilon > 0$$

Che simbolicamente si scrive:

$$X_n \xrightarrow{P} X$$

1.2.2 Legge dei grandi numeri

La legge dei grandi numeri (*Large Law Number LLN*) studia la convergenza in probabilità delle medie aritmetiche di sequenze di variabili aleatorie.
Sia $\{X_n\}_{n\geq 1}$ una data sequenza di variabili aleatorie, e per ogni $n \in \mathbb{N}$,

$$\overline{X_n} := \frac{1}{n}\sum_{i=1}^{n} X_i$$

Ognuna delle v.a. $\overline{X_n}$ è una semplice funzione: la media campionaria delle prime n v.a. della sequenza data. La LLN studia le proprietà di convergenza della sequenza delle medie campionarie $\{X_n\}_{n\geq 1}$.
La versione del teorema che oggi viene comunemente considerata classica è quella di Khinchine (1928), che vale per le v.a. X_n i.i.d. con $\mathbb{E}(X_1) < \infty$.

Teorema 1.2.2. *Legge dei grandi numeri - Khinchine [9]*
Sia $\{X_n\}$ una sequenza di v.a., indipendenti ed identicamente distribuite. Se le v.a. della sequenza hanno valore atteso, $\mu := \mathbb{E}(X_1) < \infty$, allora:

$$\overline{X_n} \xrightarrow{P} \mu = \mathbb{E}(X_1)$$

1.3 Implementazione del metodo Monte Carlo

Il fondamento teorico del metodo Monte Carlo, come discusso, è solido. Tuttavia, nella pratica ci sono aspetti importanti da considerare durante le implementazioni.

Uno di questi è il fatto che, nelle applicazioni reali, il numero di campioni a disposizione è sempre limitato. Questo è cruciale, poiché la legge dei grandi

numeri 1.2.2 garantisce la convergenza al valore atteso solo nel caso in cui $n \to \infty$

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} f(x_i) = \mathbb{E}(f(X))$$

Ciò implica che, nelle applicazioni pratiche, con un numero finito di campioni n, la convergenza non è garantita, evidenziando un problema insormontabile. Di conseguenza, ci si deve accontentare di un'approssimazione che, però, deve essere valutata qualitativamente per comprendere quanto i risultati ottenuti dai dati si discostino da quelli teorici. Idealmente, si desidera calcolare la probabilità che l'errore sia contenuto entro limiti accettabili.

$$P(\left|\overline{X_n} - \mu\right| \leq \epsilon) \tag{1.6}$$

dove $\overline{X_n} = \frac{1}{n}\sum_{i=1}^{n} f(x_i)$, $\mu = E(X_1)$, e ϵ è un valore positivo arbitrariamente piccolo.
La valutazione esatta di 1.6 è in generale difficile perché le $\overline{X_n}$, in assenza di particolari condizioni, non sono indipendenti, come sono invece X_n e non sono neanche identicamente distribuite, infatti $\text{var}(\overline{X_n}) = \frac{\sigma^2}{n}$ dipende da n.

Ancora una volta ci si avvale di un teorema limite molto importante che è il teorema del limite centrale, di seguito enunciato, che in questo capitolo viene preceduto dalla definizione di convergenza in distribuzione.

1.3.1 Convergenza in distribuzione

Definizione 1.3.1. Una sequenza di v.a. $\{X_n\}_{n\geq 1}$ di funzioni di distribuzione $\{F_n(x)\}_{n\geq 1}$, converge in distribuzione se esiste una funzione di distribuzione $F(x)$ tale che

$$\lim_{n\to\infty} F_n(x) = F(x), \quad \text{per ogni } x \text{ dove } F(x) \text{ è continua.}$$

Che simbolicamente si scrive:

$$X_n \xrightarrow{D} F(x)$$

1.3.2 Teorema del Limite Centrale

Come precedentemente affermato la Legge dei Grandi Numeri 1.2.2 garantisce la convergenza della media campionaria al valore atteso quando n tende all'infinito. Tuttavia nella pratica abbiamo a disposizione solo un numero finito di campio-

ni. Per determinare se la nostra approssimazione è sufficientemente accurata ci affidiamo al Teorema del Limite Centrale (CLT).

Teorema 1.3.2. *CLT di Lindeberg-Lévy*
Sia $\{X_n\}$ *una sequenza di v.a. indipendenti e identicamente distribuite. Se* $\mathbb{E}(X_1^2) < \infty$ *e* $\sigma^2 := var(X_1) > 0$*, allora detto* $\mu := \mathbb{E}(X_1)$ *e* $\mathcal{N}(0,1)$ *una gaussiana standard,*

$$\overline{X_n} \xrightarrow{D} \mathcal{N}(\mu, \frac{\sigma^2}{n}) \tag{1.7}$$

Vuol dire che

$$\begin{aligned} P(\left|\overline{X_n} - \mu\right| \leq \epsilon) &= P(\frac{\sqrt{n}}{\sigma}\left|\overline{X_n} - \mu\right| \leq \frac{\sqrt{n}}{\sigma}\epsilon) \\ &\approx P(Z \leq \frac{\sqrt{n}}{\sigma}\epsilon) \\ &\approx \Phi(\frac{\sqrt{n}}{\sigma}\epsilon) \end{aligned}$$

Il teorema del limite centrale asserisce che la somma di un grande numero di variabili aleatorie indipendenti e dotate della stessa distribuzione è approssimativamente normale, a prescindere dalla loro distribuzione.

La Legge dei Grandi Numeri afferma che la media campionaria di una sequenza di variabili aleatorie indipendenti e identicamente distribuite (i.i.d.) converge in probabilità al valore atteso della variabile aleatoria quando il numero di osservazioni tende all'infinito, quindi significa che il metodo Monte Carlo realizza delle stime corrette o *unbiased* del valore atteso. Questo però non è sufficiente a garantire la qualità della stima Monte Carlo. Un altro aspetto importante è la varianza delle stime. Si vuole cioè che la variabilità delle stime sia molto piccola e che quindi venga sempre prodotto un risultato che si discosta molto poco dal valore vero. Nel caso delle stime Monte Carlo, si è osservato che la varianza è espressa come $\frac{\sigma^2}{n}$, dove n è il numero di campioni. Questo risultato evidenzia come, all'aumentare del numero di campioni, le stime tendano a diventare progressivamente più precise. In particolare, il fatto che la varianza non dipenda dalla dimensione dello spazio è ciò che rende il metodo Monte Carlo particolarmente efficace nell'affrontare il problema deterministico della curse of dimensionality.

1.3.3 Altre considerazioni

Un altro aspetto da considerare in alcune implementazioni del metodo Monte Carlo riguarda il fatto che i campioni ottenuti potrebbero non essere indipendenti e identicamente distribuiti (i.i.d.). In molte applicazioni pratiche infatti, i campioni sono correlati tra loro, il che può compromettere la validità delle stime. È quindi necessario uno strumento che, data una distribuzione target, sia in grado di generare campioni che rispettino tale distribuzione.

Le Markov chain forniscono una soluzione efficace in questo contesto: algoritmi come Metropolis-Hastings, basati sulle catene di Markov, permettono di generare una sequenza di campioni che, pur non essendo indipendenti, convergono alla distribuzione desiderata. Le Markov chain e l'algoritmo Metropolis-Hastings sono discussi nella Sezione 3.1.

1.4 Importance Sampling

L'Importance Sampling è una tecnica che consente di stimare il valore atteso di una variabile aleatoria rispetto a una distribuzione complessa, sfruttando il valore atteso di una trasformazione della stessa variabile rispetto a una distribuzione alternativa, scelta in modo strategico. Questa distribuzione alternativa tipicamente è più semplice da campionare, facilitando la generazione. L'Importance Sampling diventa particolarmente utile quando il metodo Monte Carlo standard risulta inefficiente a causa della complessità della distribuzione originale.

La scelta accurata della nuova distribuzione non solo semplifica il campionamento, ma può anche ridurre la varianza della stima. In effetti, questa tecnica è in grado di diminuire la dispersione dei risultati delle simulazioni, portando a stime più precise del valore atteso. Ridurre la varianza è cruciale nelle simulazioni Monte Carlo, poiché permette di ottenere risultati accurati con un numero ridotto di campioni, migliorando l'efficienza computazionale e riducendo il costo delle simulazioni.

L'obiettivo è calcolare il valore atteso di una funzione $f(\mathbf{x})$ in cui $\mathbf{x}_i$ sono i campioni estratti dalla funzione di distribuzione $\pi(\mathbf{x})$

$$\int_{-\infty}^{+\infty} \pi(\mathbf{x}) f(\mathbf{x}) dx = \mathbb{E}_\pi \left[f(x) \right]$$

Spesso accade che quasi tutto il valore della funzione $f(\mathbf{x})$ provenga da una sotto-regione in cui π campiona raramente. Significa che molti campioni contribuiscono poco e di conseguenza rappresentano uno spreco computazionale.
Il significato dell'importance Sampling è proprio quello di cercare di ottenere risultati migliori utilizzando lo stesso numero di simulazioni cercando quindi di campionare in modo preferenziale nelle regioni in cui l'integranda $\pi(\mathbf{x})f(\mathbf{x})$ è grande in valore assoluto.

Per realizzare quindi un campionamento migliore si trasforma l'integrale introducendo una distribuzione di probabilità arbitraria $g(\mathbf{x})$.

$$\int_{-\infty}^{+\infty} \pi(\mathbf{x}) f(\mathbf{x}) \frac{g(\mathbf{x})}{g(\mathbf{x})} dx = \int_{-\infty}^{+\infty} g(\mathbf{x}) \left[\frac{\pi(\mathbf{x}) f(\mathbf{x})}{g(\mathbf{x})} \right] dx = \mathbb{E}_g \left[\frac{f(\mathbf{x})\pi(\mathbf{x})}{g(\mathbf{x})} \right]$$

dove $W(\mathbf{x}) = \frac{\pi(\mathbf{x})}{g(\mathbf{x})}$ è detta *weight function*. Questo vuol dire che

$$\mathbb{E}_\pi \left[f(x) \right] = \mathbb{E}_g \left[\frac{f(\mathbf{x})\pi(\mathbf{x})}{g(\mathbf{x})} \right] \approx \frac{1}{n} \sum_{i=1}^{n} \frac{\pi(x_i)}{g(x_i)} f(x_i)$$

i campioni $x_1, \ldots, x_n$ sono estratti da $g(\mathbf{x})$. Vuol dire che anche introducendo la distribuzione arbitraria $g(\mathbf{x})$ il valore atteso di $f(\mathbf{x})$ non cambia.

Considerando invece la varianza

$$\begin{aligned}
\mathrm{Var}(f(\mathbf{x})) &= \mathbb{E}_\pi \left[[f(\mathbf{x})]^2 \right] - \mathbb{E}_\pi \left[f(\mathbf{x}) \right]^2 \\
\mathrm{Var}(f(\mathbf{x})W(\mathbf{x})) &= \mathbb{E}_g \left[\left[\frac{f(\mathbf{x})\pi(\mathbf{x})}{g(\mathbf{x})} \right]^2 \right] - \mathbb{E}_g \left[\frac{f(\mathbf{x})\pi(\mathbf{x})}{g(\mathbf{x})} \right]^2 \\
&= \mathbb{E}_g \left[\left[\frac{f(\mathbf{x})\pi(\mathbf{x})}{g(\mathbf{x})} \right]^2 \right] - \mathbb{E}_g \left[f(\mathbf{x}) \right]^2
\end{aligned}$$

In generale quindi

$$\begin{aligned}
\mathbb{E}_g \left[\left[\frac{f(\mathbf{x})\pi(\mathbf{x})}{g(\mathbf{x})} \right]^2 \right] &= \int_{-\infty}^{+\infty} f^2(\mathbf{x}) W^2(\mathbf{x}) g(\mathbf{x}) dx \\
&= \int_{-\infty}^{+\infty} f^2(\mathbf{x}) W(\mathbf{x}) \pi(\mathbf{x}) dx
\end{aligned}$$

$$\mathbb{E}_g\left[\left[\frac{f(\mathbf{x})\pi(\mathbf{x})}{g(\mathbf{x})}\right]^2\right] = \int_{-\infty}^{+\infty} f^2(\mathbf{x})W(\mathbf{x})\pi(\mathbf{x})dx \neq \int_{-\infty}^{+\infty} f^2(\mathbf{x})\pi(\mathbf{x})dx = \mathbb{E}_\pi\left[[f(\mathbf{x})]^2\right]$$

Da questo si nota che le due varianze sono quindi diverse.
Se $\mathrm{Var}(f(\mathbf{x})W(\mathbf{x})) < \mathrm{Var}(f(\mathbf{x}))$ allora l'importance sampling permette di ottenere una stima migliore della media campionaria usando però lo stesso numero di simulazioni e inoltre permette di campionare una distribuzione diversa da quella di partenza [7] [11].

1.5 Esempi

Di seguito sono riportati due esempi che riguardano l'applicazione del metodo Monte Carlo e ne dimostrano le potenzialità e la correttezza.

1.5.1 Stima di π

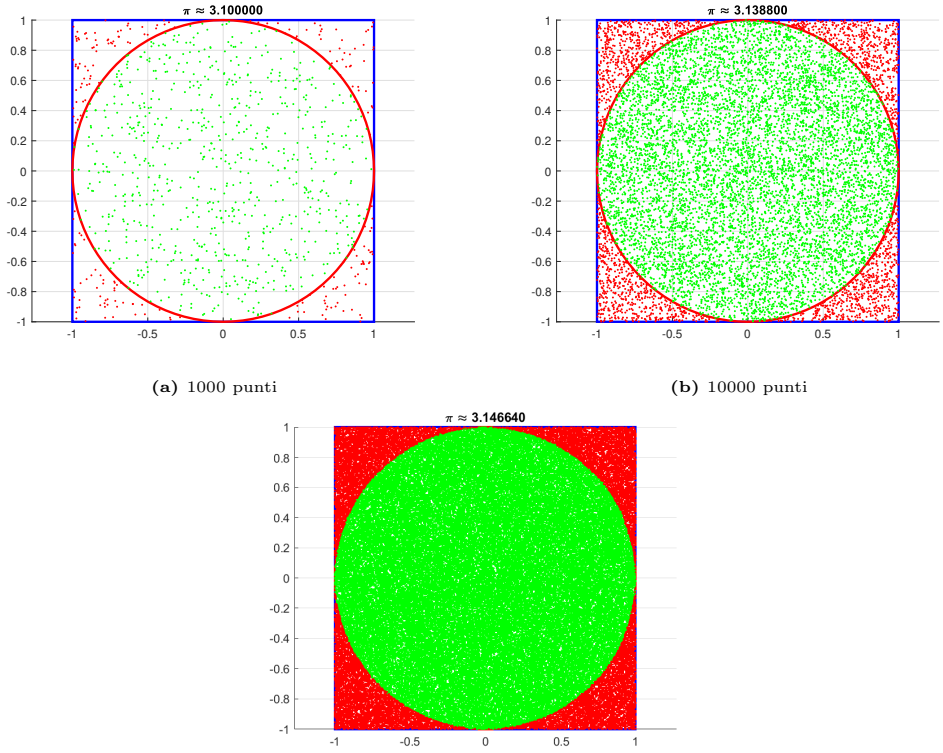
Un primo esempio riguarda la stima del valore di π attraverso un calcolo Monte Carlo. Si consideri di avere una circonferenza inscritta in un quadrato. Il rapporto tra l'area della circonferenza e quella del quadrato è pari a $\frac{\pi}{4}$. Se si generano un numero finito di punti all'interno del quadrato, è verosimile supporre che una frazione di $\frac{\pi}{4}$ di essi si troverà all'interno della circonferenza. Utilizzando MATLAB ho generato coppie casuali di coordinate cartesiane che rappresentano i punti casuali situati all'interno del quadrato. Contando quindi quanti di questi si trovano all'interno della circonferenza, rispetto al totale, si può ricavare un valore che approssima $\frac{\pi}{4}$ e di conseguenza ottenere una stima del valore di π.

Dalla Figura 1.1 si vede come all'aumentare del numero di punti, la stima si avvicina al valore vero di π, che ricordo essere 3,1415926.

In questa applicazione si doveva calcolare l'integrale definito

$$\int_0^1 \int_0^{\sqrt{1-x^2}} 1\, dxdy$$

Con il metodo Monte Carlo si è giunti ad una buona approssimazione anche senza svolgere i conti matematici.



(a) 1000 punti

(b) 10000 punti

(c) 100000 punti

Figura 1.1: Stima del valore di *pi*

1.5.2 Confronto approssimazione calcolo integrale: Riemann e Monte Carlo

Si supponga di dover calcolare numericamente

$$I := \int_0^1 f(x)\,dx$$

per una funzione $f(x)$, continua nell'intervallo $[0,1]$, avente però antiderivata non nota. Il metodo classico è quello di utilizzare l'approssimazione di Riemann che consiste nel suddividere l'intervallo $[0,1]$ in n sottointervalli di lunghezza uguale, pari a $\Delta := \frac{1}{n}$, allora

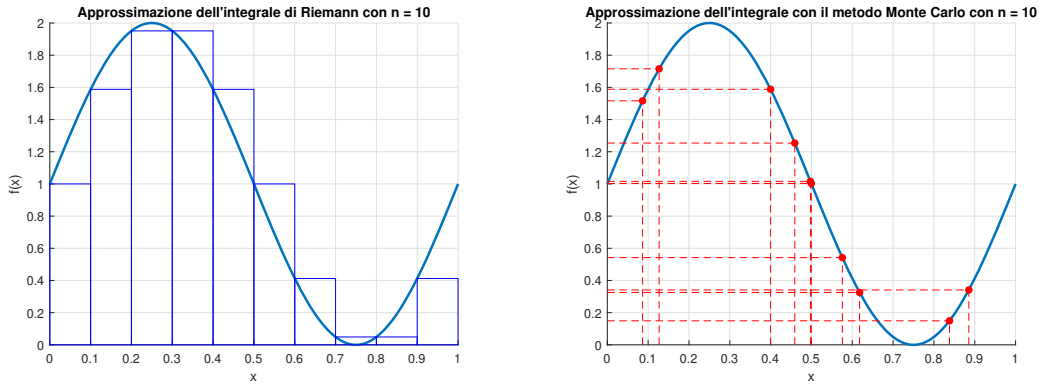
$$\int_0^1 f(x)\,dx \approx \sum_{k=1}^{n} f(k\Delta)\Delta$$

L'approssimazione di Riemann è tanto migliore tanto più è grande n.
Un modo alternativo [9] è invece quello di usare il metodo Monte Carlo, che invece di suddividere l'intervallo in sotto intervalli regolari, genera n punti in $[0,1]$ in modo casuale, siano essi $x_1, x_2, \ldots x_n$. Si calcola poi la media dei valori della

funzione in questi punti e si ottiene l'approssimazione dell'integrale

$$\int_0^1 f(x)\,dx \approx \frac{1}{n}\sum_{k=1}^{n} f(x_k)$$

Per funzioni semplici o per basse dimensioni, le somme di Riemann sono generalmente più efficienti. Tuttavia, in spazi ad alta dimensione, il metodo Monte Carlo diventa spesso più pratico, poiché la sua accuratezza non diminuisce drasticamente con l'aumento delle dimensioni.



(a) Riemann **(b)** Monte Carlo

Figura 1.2: Metodi a confronto

Capitolo 2

Generazione di numeri casuali e campionamento di variabili aleatorie

La generazione di numeri casuali è essenziale per il metodo Monte Carlo come anticipato nella Sezione 1.3.3. In generale si vorrebbe generare una sequenza infinita di campioni presi da variabili aleatorie indipendendi e identicamente distribuite i.i.d. da una distribuzione di probabilità arbitraria. Quando tale distribuzione assume valori equiprobabili nell'intervallo $(0, 1)$, il generatore di tale distribuzione è detto generatore di numeri casuali uniformi.
La maggior parte dei computer possiede un generatore di numeri casuali che richiede di inserire dall'utente un valore iniziale detto "seed" e successivamente vengono prodotti dei numeri casuali nell'intervallo $(0, 1)$. In MATLAB per esempio si utilizza il comando **rand** proprio per questo scopo. [7]

2.1 Numeri casuali e pseudocasuali

Per lo studio del metodo Monte Carlo è necessario fare una premessa importante sulla differenza effettiva tra i numeri casuali e pseudocasuali. Quando si parla di numeri casuali si intende un valore preso da una variabile aleatoria. Tuttavia nelle applicazioni del metodo Monte Carlo quando si determina una sequenza di numeri non è necessario che questa sia casuale nella definizione puramente statistica, ma è sufficiente che abbia delle proprietà che siano solo simili a una vera sequenza casuale.
I numeri propriamente casuali hanno le proprietà di essere imprevedibili e irriproducibili. Tali caratteristiche però appartengono solo a processi casuali fisici, alcuni esempi significativi sono: il decadimento radioattivo, il rumore termico

nei dispositivi elettronici, i tempi di arrivo dei raggi cosmici,... Una sequenza di numeri casuali generata con uno dei precedenti esempi presi dal mondo fisico sarebbe assolutamente valido per affrontare un qualunque problema Monte Carlo, se non fosse che nella pratica è molto difficile costruire dei generatori casuali che si avvalgano di processi fisici. Questo perché i processi attraverso cui vengono generati tali numeri risultano non essere abbastanza veloci.
Un'eccezione però è il lavoro di Frigerio e Clark (1975) e Frigerio e altri (1978). Loro considerarono una sorgente di particelle α e un contatore ad alta risoluzione che misurava il numero di decadimenti avvenuti ogni 20 ms. Per ogni periodo si contavano in media $24,315$ decadimenti. Quando tale numero risultava dispari, veniva impostato in un nastro magnetico il bit a 1 e quando era pari a 0. Si otteneva così una sequenza di 0 e 1. Per eliminare il bias introdotto dal fatto che la probabilità $P(1)$ fosse diversa da $P(0)$, accoppiarono a due a due i bit trovati e successivamente eliminarono tutte le coppie di numeri uguali. Delle coppie di numeri diversi invece usarono solo il secondo. Questi passaggi garantivano proprio $P'(0) = P'(1)$, infatti:

$$P'(0) = P(1)\,P(0)$$
$$P'(1) = P(0)\,P(1)$$

E' necessario però considerare che a seguito di tale esperimento i valori di $P'(0)$ e $P'(1)$ non normalizzano a 1: le due probabilità sono uguali, ma la loro somma è minore di 1. Infatti la probabilità di rigettare le coppie di bit uguali è

$$P^2(0) + P^2(1) \geq \frac{1}{2}$$

In aggiunta a ciò vi è anche il fatto che metà dei bit vengono rigettati, portando quindi a un'efficienza solo del 25%.

I numeri pseudo-casuali sono invece quelli che sono riproducibili, ma che sono indistinguibili da una sequenza matematicamente casuale. Tuttavia nella maggior parte delle applicazioni del metodo Monte Carlo vengono utilizzati numeri pseudo-casuali.[2]

Per brevità verrà utilizzato successivamente il termine casuale al posto di pseudo-casuale.

2.2 Generazione di numeri casuali uniformi

La maggior parte degli algoritmi di generazione di numeri casuali più utilizzati attualmente si avvale di algoritmi semplici, che possono essere implementati facilmente in un computer. [7] Questi algoritmi sono descritti da una tupla (S, f, μ, U, g) dove:

- S è un insieme finito di stati
- $f : S \to S$
- μ è una distribuzione di probabilità su S
- U è lo spazio di output, che assumeremo (0,1)
- $g : S \to U$

Un algoritmo di generazione di numeri casuali ha origine da un *seed* S_0 che è un valore iniziale estratto da una distribuzione di probabilità μ definita su un insieme di stati S. La scelta del *seed* è fondamentale perché determina l'intera sequenza di numeri: *seed* diversi originano sequenze diverse. Successivamente viene calcolato lo stato attuale applicando una funzione f allo stato precedente seguendo una dinamica deterministica definita proprio dalla funzione f. Una volta determinato il nuovo stato si applica una funzione g per produrre il numero casuale U_t.

L'algoritmo ha la seguente struttura:

Algoritmo 1 Generazione di numeri casuali

1: **Inizializzazione** Campiona il *seed* S_0 dalla distribuzione μ su S e inizializza $t = 0$.
2: **Transizione** Imposta $S_t = f(S_{t-1})$.
3: **Output** Imposta $U_t = g(S_t)$
4: **Ripetizione** $t = t + 1$ e ripeti da 2.

L'algoritmo produce una sequenza $U_1, U_2, U_3, \ldots$ di numeri casuali. A partire da un certo *seed*, la sequenza di stati (e quindi di numeri casuali) deve ripetersi, questo perché lo spazio degli stati è finito e quindi è finito anche il numero di stati che si possono generare.

Definizione 2.2.1. Si definisce *lunghezza del periodo del generatore di numeri casuali* il minor numero di passi necessari prima di entrare in uno stato già visitato.

2.2.1 Proprietà di un buon generatore

La scelta di un buon generatore di numeri casuali dipende da vari fattori, come il tipo di applicazione che si deve implementare e per cui possono essere necessarie delle propietà particolari. [7] [4] Generalmente le caratteristiche di un buon generatore possono essere riassunte nelle seguenti voci:

- Come anticipato nella Sezione 2.1 è importante che il generatore produca una sequenza di numeri che superino i test statistici e quindi risultino indistinguibili da una sequenza i.i.d..
- Un buon generatore deve inoltre essere veloce, efficiente economico e facile da usare e richiedere poca memoria del computer
- Per quanto riguarda invece la riproducibilità, una sequenza deve poter essere riproducibile senza dover memorizzare l'intera sequenza e deve avere anche un periodo il più lungo possibile per evitare problemi di duplicazione e dipendenza.

2.3 Generatori basati su ricorrenze lineari

I generatori basati su ricorrenze lineari sono i più conosciuti e sono gli algoritmi più largamente diffusi per produrre numeri casuali. Questo perché hanno dei vantaggi che li rendono molto competitivi nella pratica. Sono infatti caratterizzati dalla loro velocità, facilità nell'implementazione e portabilità.
In questa Sezione è riportato il generatore più classico di questa tipologia, che è il Generatore Lineare Congruenziale.

2.3.1 Generatore Lineare Congruenziale

Un Generatore Congruenziale Lineare (*Linear Congruential Generators* LCG) è un generatore di numeri casuali della forma di Algoritmo 1, in cui lo stato $S_t = X_t \in \{0, \dots, m-1\}$, per un certo numero intero positivo m chiamato modulo, è definito da

$$X_t = (aX_{t-1} + c) \mod m, \quad t = 1, 2, \dots$$

Dove a viene detto moltiplicatore e c incremento e m modulo, $a, c \in \{0, \dots, m-1\}$. L'operatore $\mod m$, applicato all'algoritmo 1, divide $(aX_{t-1} + c)$ per il va-

lore di m, e assegna il valore del resto di tale divisione a X_t. Ottenendo quindi come massimo valore $m - 1$.
Operando le dovute modifiche all'Algoritmo 1 si ottiene:

Algoritmo 2 Generatore congruenziale lineare

1: **Inizializzazione** Campiona il *seed* S_0 dalla distribuzione μ su S e inizializza $t = 0$.
2: **Transizione** Imposta $S_t = X_t = (aX_{t-1} + c) \mod m$.
3: **Output** Imposta $U_t = g(S_t)$
4: **Ripetizione** $t = t + 1$ e ripeti da 2.

Quando $c = 0$, il generatore viene chiamato *generatore congruenziale moltiplicativo*. La maggior parte delle implementazioni esistenti di LCG è di questa forma, questo perché in generale l'incremento c non ha un grande impatto sulla qualità di un LCG. [7] [4]
La funzione di output per un LCG è semplicemente

$$U_t = \frac{X_t}{m}.$$

La semplicità di questo algoritmo rappresenta il suo punto di forza maggiore, tuttavia non è garantito che per ogni scelta di a, c, m, la sequenza di numeri casuli ottenuta sia sufficientemente buona.
Di seguito viene riportato un esempio che mette in luce i limiti di una cattiva scelta dei parametri.

Esempio 2.3.1 (Conseguenza di una cattiva scelta dei parametri)
Scegliendo $a = 5$, $b = 1$, $m = 100$ e scegliendo come *seed* S_0 il valore 1, si avrà:

$$\begin{aligned}
S_0 &= 1 \\
S_1 &= (5 * 1 + 1) \mod 100 = 6 \\
S_2 &= (5 * 6 + 1) \mod 100 = 31 \\
S_3 &= (5 * 31 + 1) \mod 100 = 56 \\
S_4 &= (5 * 56 + 1) \mod 100 = 81 \\
S_5 &= (5 * 81 + 1) \mod 100 = 6 \\
S_6 &= S_2 = 31 \\
S_7 &= S_3 = 56 \\
\vdots &= \vdots
\end{aligned}$$

Si nota facilmente che i valori possibili generati da questo esempio sono solo 4, come il periodo del generatore e quindi la scelta progettuale precedente di a, c, m non è assolutamente buona.

Per ottenere un generatore migliore dell'Esempio 2.3.1 è necessario fare una scelta diversa sui parametri. Il valore di m deve essere il più grande possbile, esso quindi dipende dal processore che si sta utilizzando. Facendo un esempio con un processore a 32 bit, si considera un valore pari a $m = 2^{t-1} - 1$, dove t è il numero di bit. Quindi nell'esempio $m = 2^{31} - 1$. E' ragionevole poi scegliere $a = 7^5$ e $b = 0$ [6].

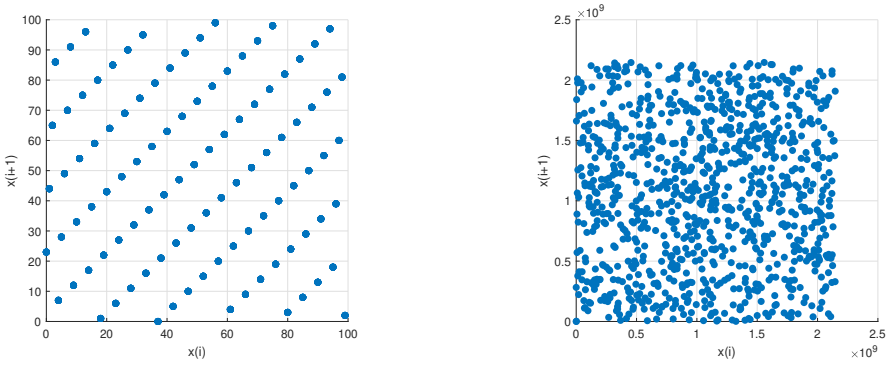
2.3.2 Test di Casualità

Come è appena stato dimostrato i generatori basati sul metodo congruenziale lineare presentano delle criticità. Gli studi di Marsaglia [1] in merito dimostrano che i numeri generati da questi metodi tendono a non coprire in modo uniforme l'intero spazio multidimensionale, cadendo su una serie di piani paralleli, evidenziando una certa regolarità. Marsaglia notò l'impatto critico dovuto alla qualità della scelta dei parametri, come mostrato nell'Esempio 2.3.1, sulla correlazione tra i numeri della sequenza generata, riducendo l'efficacia del metodo per simulazioni o altre applicazioni numeriche. La **figura di Marsaglia** o scatter plot

di Marsaglia è un metodo per capire graficamente se vi è correlazione tra i vari numeri generati.
Bisogna costituire coppie di numeri consecutivi

$$(x_i, x_{i+1}) \quad \forall i \in (0, n)$$

e rappresentare ciascuna coppia come punto nel piano cartesiano. Se i punti sono distribuiti in modo da evidenziare pattern regolari allora vuol dire che vi è correlazione tra di essi, altrimenti il generatore è buono.



(a) $a = 21, c = 23, m = 100$ **(b)** $a = 16807, c = 0, m = 2^{31} - 1$

Figura 2.1: Esempi della Figura di Marsaglia con $n = 1000$ punti

L'esempio in Figura 2.1a mostra che si creano delle linee parallele proprio a indicare la correlazione dei campioni che invece dovrebbero essere sparsi, come in Figura 2.1b. In particolare la Figura 2.1b rappresenta il generatore di Lehmer [4] che porta a buoni risultati.

2.4 Generatori di variabili aleatorie

Nella Sezione 2.3 si è discusso della generazione di campioni da variabili uniformi, ora invece l'obiettivo è quello di generare campioni di variabili aleatorie indipendenti e identicamente distribuite aventi una distribuzione di probabilità generica.

1. Prima bisogna generare una sequenza di numeri causali uniformi $U_1, \ldots U_k$ per $k = 1, 2, \ldots$ come visto nella sezione precedente

2. Poi si restituisce $\mathbf{X} = g(U_1, \ldots, U_k)$, $g : (0, 1)^k \to \mathbb{R}^d$

Vi sono molti metodi per generare variabili casuali in modo esatto, in cui cioè ogni variabile generata ha esattamente la distribuzione richiesta. Di seguito viene riportato uno dei più importanti.

2.4.1 Metodo della Trasformazione Inversa

Tutte le variabili aleatorie con funzione di distribuzione invertibile possono essere scritte come trasformazione di una variabile aleatoria uniforme.
Sia X una variabile aleatoria con Funzione di Distribuzione (FdD) F. Essendo F una funzione non decrescente, la sua inversa si può definire come

$$F^{-1}(y) = \inf\{x : F(x) \geq y\}, \quad 0 \leq t \leq 1.$$

Sia inoltre $U \sim \mathcal{U}(0,1)$. La funzione di distribuzione (FdD) di $F^{-1}(U)$ è data da

$$P(F^{-1}(U) \leq x) = P(U \leq F(x)) = F(x)$$

Quindi per generare una variabile X con una FdD F, estraggo $U \sim \mathcal{U}(0,1)$ e sia $X = F^{-1}(U)$. L'algoritmo generale [7] per generare variabili casuali da una FdD F qualsiasi

Algoritmo 3 Metodo della Trasformazione Inversa

1: Genera $U \sim \mathcal{U}(0,1)$
2: Restituisci $X = F^{-1}(U)$.

L'algoritmo molto semplice è di grande importanza perché semplifica notevolmente il processo di generazione. Infatti si basa semplicemente sulla generazione di una sequenza di numeri casuali uniformi a cui poi viene applicata una trasformazione inversa che dipende dalla FdD. Di seguito è riportato un esempio per quanto riguarda il caso della distribuzione esponenziale.

Esempio 2.4.1 (Distribuzione esponenziale)
La distribuzione di probabilità esponenziale, dato un parametro λ è

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

La sua funzione di distribuzione e la sua funzione di distribuzione inversa sono

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} f(t)\,dt = \begin{cases} 1 - e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

$$F^{-1}(u) = \begin{cases} -\frac{1}{\lambda}\ln(1-u) & u \geq 0 \\ 0 & u < 0 \end{cases} \tag{2.1}$$

In questo esempio quindi basta generare una variabile casuale U da una uniforme e applicare la trasformazione precedente 2.1. Di seguito vi è il grafico e il codice realizzato con MATLAB per l'implementazione dell'esempio.

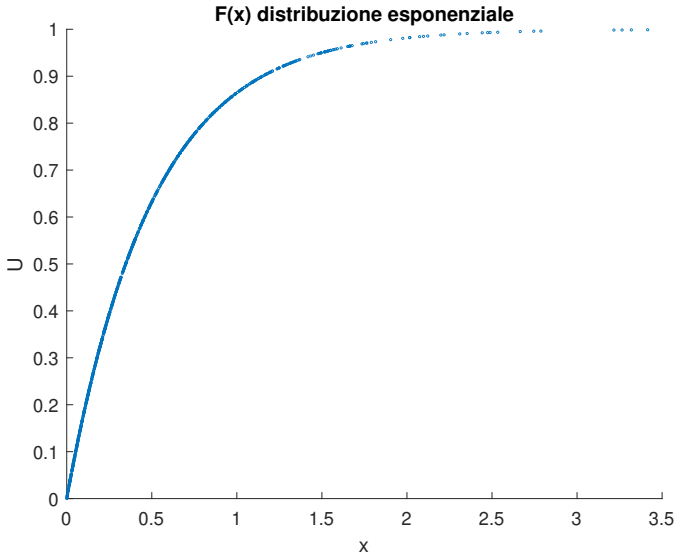


Figura 2.2: Implementazione del metodo della trasformazione inversa per distribuzione esponenziale

```
%Generatore distribuzione esponenziale con metodo trasformazione
    inversa
a = 2 %parametro positivo
%FdD inversa
F_inversa = @(u) -1 / a * log(1-u);
%numero di campioni da generare
n = 1000;
%generazione campioni di U
U = rand(1,n);
F=zeros(1,n);
for i=1 : n
    F(i) = F_inversa(U(i));
end
```

Questo tipo di generatori però ha un limite importante nell'applicabilità: è necessario che F^{-1} sia ricavabile analiticamente o algoritmicamente, altrimenti è impossibile o difficile trovare la trasformazione inversa.

Per un numero crescente di applicazioni Monte Carlo i generatori di variabili casuali esatti, come quello della trasformazione inversa, Algoritmo 3, sono difficili o impossibili da realizzare, bisogna quindi utilizzare dei metodi di approssimazione. Tra i più noti e utilizzati ci sono i metodi Markov Chain Monte Carlo che verrano analizzati nel Capitolo 3.

Capitolo 3

Markov Chain Monte Carlo

Nel primo capitolo, con l'esempio della stima bayesiana 1.1.1, è emerso come la cosiddetta *maledizione della dimensionalità* sia una problematica importante quando si risolvono integrali in spazi di dimensioni elevate. Intuitivamente il volume di uno spazio ad alta dimensione è tanto vasto che il numero di dati a disposizione non è sufficiente per eseguire calcoli esatti sul valore dell'integrale in relazione al volume dello spazio. Per ottenere quindi risultati affidabili e una significatività statistica, il numero di dati dovrebbe crescere esponenzialmente nella dimensione dello spazio, oppure un'alternativa migliore è quella di utilizzare approcci stocastici Monte Carlo. In questo modo si può affrontare il problema per via numerica, generando campioni casuali indipendenti da distribuzioni qualsiasi. Tuttavia tale generazione non è una soluzione immediata e banale, ma è al contrario un problema complicato.
Una soluzione proposta utilizza le Markov Chain, che nel caso in cui rispettino delle particolari proprietà consentono di generare campioni provenienti da una distribuzione generica. Per generare le Markov Chain con le caratteristiche ricercate si utilizza l'algoritmo Metropolis-Hastings.
Viene quindi introdotto prima il concetto di Markov Chain e le proprietà chiave ricercate, per poi descrivere l'algoritmo Metropolis-Hastings. Segue poi un esempio di implementazione per dimostrare l'applicabilità dell'algoritmo e la qualità del campionamento.

3.1 Markov Chain

Le Markov chain sono uno strumento matematico utilizzato per modellare e analizzare sistemi che evolvono nel tempo attraverso una serie di stati discreti. Questi

modelli sono basati su una proprietà, detta **proprietà di Markov** che caratterizza i modelli in cui lo stato del sistema futuro dipende solo dallo stato attuale e non dalla sequenza di stati precedenti. Le Markov chain si applicano a una vasta classe di problemi in cui vi è la necessità di descrivere processi stocastici.

Formalmente, si consideri un insieme di vettori aleatori di dimensione d,

$$\{X_t, t = 0, 1, 2, \dots\}$$

Si dice che è una **collezione di Markov** se, considerando

$$X_t \,|\, X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2}, \dots, X_0 = x_0$$

si ha che:

$$P(X_t \in A|\, X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2}, \dots, X_0 = x_0) = P(X_t \in A \,|\, X_{t-1} = x_{t-1}) \tag{3.1}$$

$$\forall\, A \in \mathscr{F},\ \forall\, t,\ \forall\, x\,, \text{dove}\, \mathscr{F} \text{ è la } \sigma\text{-algebra}$$

Una catena in particolare è descritta in modo cruciale dalla matrice di transizione, che indica le probabilità di passare da uno stato ad un altro.

Una proprietà importante che servirà in seguito delle Markov chain è la stazionarietà. Una Markov chain si dice stazionaria se dopo un determinato numero di passi temporali la distribuzione di probabilità del sistema si stabilizza, perdendo quindi la dipendenza dal tempo. Lo stato di equilibrio deve rimanere tale anche dopo l'applicazione della matrice di transizione del sistema.

Definizione 3.1.1. Una Markov chain si dice *stazionaria* se la densità di probabilità condizionata non varia nel tempo

$$P(X_1 \in A \,|\, X_0 = x) = P(X_T \in A \,|\, X_{t-1} = x) \doteq P(A,\, x)$$

$$\forall\, A \in \mathscr{F},\ \forall\, t,\ \forall\, x$$

Il kernel di transizione è un'altra definizione importante legata alla matrice di transizione. Rappresenta la probabilità che la funzione k si trovi in una certa regione A.

Definizione 3.1.2. Il *kernel di transizione* di una Markov chain stazionaria è

una funzione $k(a, x)$ tale che:

$$P(A,\, x) = \int_A k(a, x)da$$
$$k(\cdot \,|\, \cdot) = p_{X_{t+1}|X_t}(\cdot \,|\, \cdot)$$

Quindi $k(\cdot \,|\, \cdot)$ definisce completamente le leggi di probabilità della Markov chain. Infatti sia π_0 una densità di probabilità iniziale, il kernel di transizione permette di calcolare la densità di probabilità congiunta per ogni n-uple di vettori dalla catena [12].

Esempio 3.1.1

$$\begin{aligned} p_{X_0,X_1,X_2}(x_0, x_1, x_2) &= p_{X_0}(x_0)\, p_{X_1,X_0}(x_1, x_0)\, p_{X_2|X_1,X_0}(x_2|x_1, x_0) \\ &= \pi_0(x_0)\, k(x_1, x_0)\, k(x_2, x_1) \end{aligned}$$

Sia per esempio π_{t-1} la densità di probabilità di X_{t-1} e π_t la densità di probabilità di X_t si ha:

$$\pi_t(a) = \int k(a, x)\, \pi_{t-1}(x)dx \tag{3.2}$$

A ogni stato è associata la densità di probabilità π_t, come viene rappresentato nella Figura seguente.

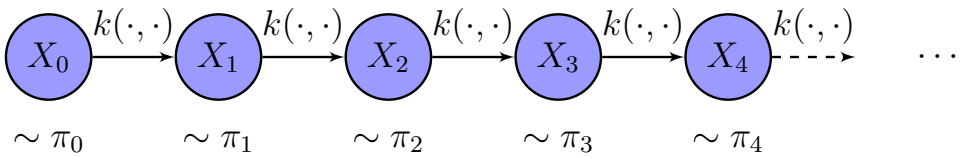


Figura 3.1: Diagramma di stato di una Markov Chain

Inoltre però se π è una densità di probabilità invariante per la Markov chain allora l'Equazione 3.2, diventa

$$\pi(a) = \int k(a, x)\, \pi(x)dx \tag{3.3}$$

E la rappresentazione degli stati si semplifica poiché la densità π non cambia tra i vari stati.

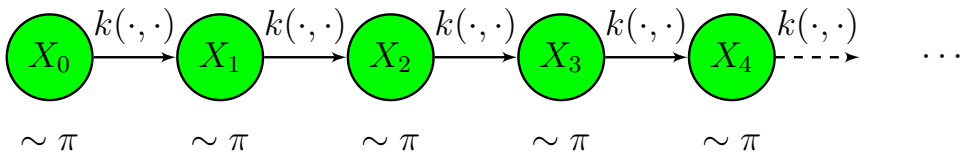


Figura 3.2: Caption

Definizione 3.1.3. Sia π una distribuzione invariante per la catena. Una Markov chain si dice *irriducibile* se:

$$\forall x, A \in \mathscr{F}, \text{con} \int_A \pi(x)\, dx > 0,$$

$$\exists t > 0 \mid \quad P(X_t \in A \mid X_0 = x) > 0 \tag{3.4}$$

L'irriducibilità è quindi la possibilità di visitare tutte le regioni di interesse di π partendo da qualunque stato x. Come rappresentato in Figura 3.3

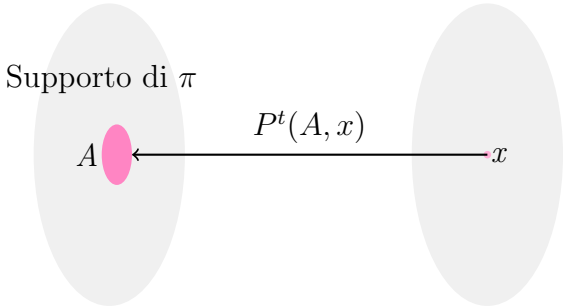


Figura 3.3: Supporto di π

Definizione 3.1.4. Si dice che una Markov chain stazionaria con distribuzione iniziale π è *reversibile* se soddisfa la condizione di *detailed balance*:

$$\pi(x_i)k(x_j, x_i) = \pi(x_j)k(x_i, x_j) \tag{3.5}$$

Lemma 3.1.5. *Sia P una matrice di transizione sullo spazio degli stati numerabile E, e sia π una distribuzione di probabilità su E. Se $\forall i, j \in E$, le condizioni 3.5 di detailed balance sono soddisfatte, allora π è una distribuzione stazionaria di P.*

Il lemma esprime come la proprietà di detailed balance sia condizione sufficiente per l'invarianza della densità. Esistono infatti Markov chain che hanno distribuzini stazionarie che non soddisfano la condizione di detailed balance.

La definizione della Markov Chain e di alcune sue proprietà saranno nell'interesse dell'analisi delle Markov Chain Monte Carlo e dell'algoritmo Metropolis-Hastings che verrano descritte nella prossima sezione.

3.2 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) è un metodo generico per il campionamento approssimativo da una distribuzione arbitraria. Si basa sul costruire una Markov Chain irriducibile 3.1.3 avente densità invariante π pari alla densità che si desidera campionare. Nel caso della stima bayesiana la densità da campionare è la densità a posteriori.
Prima quindi si generano i campioni dalla catena di Markov e successivamente si utilizza l'integrazione Monte Carlo per ottenere le stime.
Data una qualsiasi distribuzione che si vuole campionare, che può essere anche molto complicata o in uno spazio di dimensioni molto elevate, si genera una Markov Chain che esplori lo spazio degli stati, cioè delle possibili soluzioni. La catena è progettata per essere irriducibile, quindi può raggiungere tutti gli stati a partire da uno stato iniziale qualsiasi e convergere verso una distribuzione invariante. Una volta raggiunta la convergenza alla densità invariante, i campioni generati dalla catena possono stimare le quantità di interesse mediante l'integrazione Monte Carlo.

3.2.1 Algoritmo Metropolis-Hastings

L'algoritmo Metropolis-Hastings si applica al seguente contesto: si supponga di voler generare dei campioni da una densità di probabilità arbitraria e multidimensionale, detta distribuzione target (*target distribution*), del tipo

$$\pi(\mathbf{x}) = \frac{p(\mathbf{x})}{Z}, \quad \mathbf{x} \in \mathscr{X}$$

dove $p(\mathbf{x})$ è una funzione positiva e Z è una costante di normalizzazione che può essere nota o meno. Sia inoltre $q(\mathbf{y} \mid \mathbf{x})$ la densità di proposta (*proposal density*). Si tratta di una densità di distribuzione utilizzata per proporre nuovi campioni da esplorare nel processo di campionamento e che influenza l'efficienza dell'algoritmo, poiché definisce la probabilità di muoversi dallo stato $\mathbf{x}$ allo stato $\mathbf{y}$ durante il

campionamento. Quindi intuitivamente più $q(\mathbf{y} \mid \mathbf{x})$ è simile alla distribuzione target $\pi(\mathbf{x})$, più sarà veloce la generazione dei campioni.

L'algoritmo Metropolis-Hastings permette di campionare da distribuzioni complesse senza dover calcolare esplicitamente la normalizzazione di $\pi(x)$ e si basa sul seguente Algoritmo che segue una strategia *trial-and-error*.

Algoritmo 4 Algoritmo Metropolis-Hastings

1: Dato lo stato attuale $X_t = x$, si genera un nuovo campione $\mathbf{Y} \sim q(\mathbf{y} \mid X_t)$
2: Si genera $U \sim \mathcal{U}(0, 1)$ e se:

$$X_{t+1} = \begin{cases} \mathbf{Y} & \text{se } U \leq \alpha(\mathbf{X}_t, \mathbf{Y}) \\ \mathbf{X}_t & \text{altrimenti} \end{cases}, \quad \alpha(\mathbf{x}, \mathbf{y}) = \min \left\{ \frac{\pi(\mathbf{y})\, q(\mathbf{x} \mid \mathbf{y})}{\pi(\mathbf{x})\, q(\mathbf{y} \mid \mathbf{x})}, 1 \right\}$$

La probabilità $\alpha(\mathbf{x}, \mathbf{y})$ è detta probabilità di accettazione (*acceptance probability*) e π diventa la densità invariante della Markov chain generata. Quando si propone un nuovo campione l'accettazione o meno di questo dipende dalla probabilità di accettazione $\alpha(\mathbf{x}, \mathbf{y})$ che è di fatto una scelta progettuale poiché dipende dalla densità di proposta.

Teorema 3.2.1. *Se la probabilità di accettazione è*

$$\alpha(c, x) = \min \left\{ 1, \frac{\pi(c) q(x|c)}{\pi(x) q(c|x)} \right\}$$

allora π dventa la densità invariante della Markov chain generata.

Lemma 3.2.2.

$$\alpha(c, x) = \min \left(1, \frac{\pi(c) q(x|c)}{\pi(x) q(c|x)} \right)$$

implica

$$\pi(X_t) q(X_{t+1}|X_t) \alpha(X_{t+1}, X_t) = \pi(X_{t+1}) q(X_t|X_{t+1}) \alpha(X_t, X_{t+1}) \tag{3.6}$$

Il teorema afferma che la scelta della probabilità di accettazione secondo la formula data garantisce che π sia la distribuzione invariante, mentre il lemma formalizza come questa scelta soddisfi la condizione di detailed balance, assicurando la convergenza alla distribuzione target.

Dimostrazione Lemma

Bisogna dimostrare che l'equivalenza vale per ogni coppia (X_t, X_{t+1}).
Le possibili coppie (X_t, X_{t+1}) si dividono in due gruppi:

$$\text{gruppo 1} \quad \frac{\pi(X_{t+1})q(X_t|X_{t+1})}{\pi(X_t)q(X_{t+1}|X_t)} \leq 1$$

implica che

$$\alpha(X_{t+1}, X_t) = \frac{\pi(X_{t+1})q(X_t|X_{t+1})}{\pi(X_t)q(X_{t+1}|X_t)} \quad \text{e} \quad \alpha(X_t, X_{t+1}) = 1$$

Infatti dalla prima di può moltiplicare il denominatore a destra e moltiplicare per $\alpha(X_t, X_{t+1}) = 1$ per ottere la forma 3.6.

$$\pi(X_t)q(X_{t+1}|X_t)\alpha(X_{t+1}, X_t) = \pi(X_{t+1})q(X_t|X_{t+1})\underbrace{\alpha(X_t, X_{t+1})}_{=1}$$

$$\text{gruppo 2} \quad \frac{\pi(X_{t+1})q(X_t|X_{t+1})}{\pi(X_t)q(X_{t+1}|X_t)} > 1$$

$$\frac{\pi(X_t)q(X_{t+1}|X_t)}{\pi(X_{t+1})q(X_t|X_{t+1})} < 1$$

implica che

$$\alpha(X_t, X_{t+1}) = \frac{\pi(X_t)q(X_{t+1}|X_t)}{\pi(X_{t+1})q(X_t|X_{t+1})} \quad \text{e} \quad \alpha(X_{t+1}, X_t) = 1$$

Come visto per il Gruppo 1, si può riscrivere nella forma 3.6.

$$\pi(X_t)q(X_{t+1}|X_t)\underbrace{\alpha(X_{t+1}, X_t)}_{=1} = \pi(X_{t+1})q(X_t|X_{t+1})\alpha(X_t, X_{t+1})$$

□

Dimostrazione Teorema

Per dimostrare la correttezza del teorema bisogna dimostrare che la distribuzione π sia stazionaria rispetto al kernel di transizione. Sia quindi $k(X_{t+1}|X_t)$ il kernel di transizione che stabilisce la probabilità di passare da uno stato X_t a uno stato X_{t+1}. Lo consideriamo quindi nel caso in cui il nuovo stato X_{t+1} è uguale o diverso dallo stato attuale X_t.

Nel primo caso $k(X_{t+1}|X_t)$ è dato dalla probabilità di proporre il nuovo campione X_{t+1} moltiplicato per la probabilità di accettarlo, $\alpha(X_{t+1}|X_t)$.

Nel secondo caso la probabilità del nuovo campione è data sia dalla probabilità di effettivamente proporlo e accettarlo più la probabilità di non accettare una qualsiasi proposta.

$$k(X_{t+1}|X_t) = \begin{cases} q(X_{t+1}|X_t)\alpha(X_{t+1}|X_t) & X_{t+1} \neq X_t \\ q(X_{t+1}|X_t)\alpha(X_{t+1}|X_t) + \overbrace{\left(1 - \int q(c|X_t)\alpha(c, X_t)dc\right)}^{\text{P[rifiutare qualsiasi campione]}} & X_{t+1} = X_t \end{cases}$$

Il kernel della catena si può riscrivere in maniera compatta usando il delta di Kronecker:

$$k(X_{t+1}|X_t) = q(X_{t+1}|X_t)\alpha(X_{t+1}, X_t) + \delta(X_{t+1} = X_t)\left(1 - \int q(c|X_t)\alpha(c, X_t)dc\right)$$

Per quando appena dimostrato, per il lemma precedente e per la simmetria del termine $\delta(X_{t+1} = X_t)\left(1 - \int q(c|X_t)\alpha(c, X_t)dc\right)$, si ottiene

$$\pi(X_t)k(X_{t+1}|X_t) = \pi(X_{t+1})k(X_t|X_{t+1})$$

Integrando

$$\int \pi(X_t)k(X_{t+1}|X_t)dX_t = \pi(X_{t+1}) \underbrace{\int k(X_t|X_{t+1})dX_t}_{=1 \text{ per definizione}}$$

Si ottiene direttamente l'espressione di $\pi(X_{t+1})$

$$\pi(X_{t+1}) = \int \pi(X_t)k(X_{t+1}|X_t)dX_t$$

Che confrontata con la definizione di distribuzione invariante 3.3 si conclude che π è la densità invariante

□

Osservazioni

- La catena si muove sempre, infatti anche se il campione viene rifiutato il prossimo stato equivale al precedente.
- In generale l'algoritmo può generare campioni da π che sono correlati ma non indipendenti
- La densità di target π è nota eccetto per un fattore di normalizzazione

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\left\{\frac{\pi(\mathbf{y})\, q(\mathbf{x}\,|\,\mathbf{y})}{\pi(\mathbf{x})\, q(\mathbf{y}\,|\,\mathbf{x})}\,,\, 1\right\} \qquad \pi(\mathbf{x}) \propto p_{y|x}(y|x)\, p_x(x)$$

- In teoria, se la catena è irriducibile 3.1.3, l'algoritmo è valido per ogni $q(\cdot|\cdot)$, nella pratica invece la scelta di q è cruciale e deve essere:
  - facile da campionare
  - semplice da valutare punto per punto
  - in grado di esplorare rapidamente il supporto di π

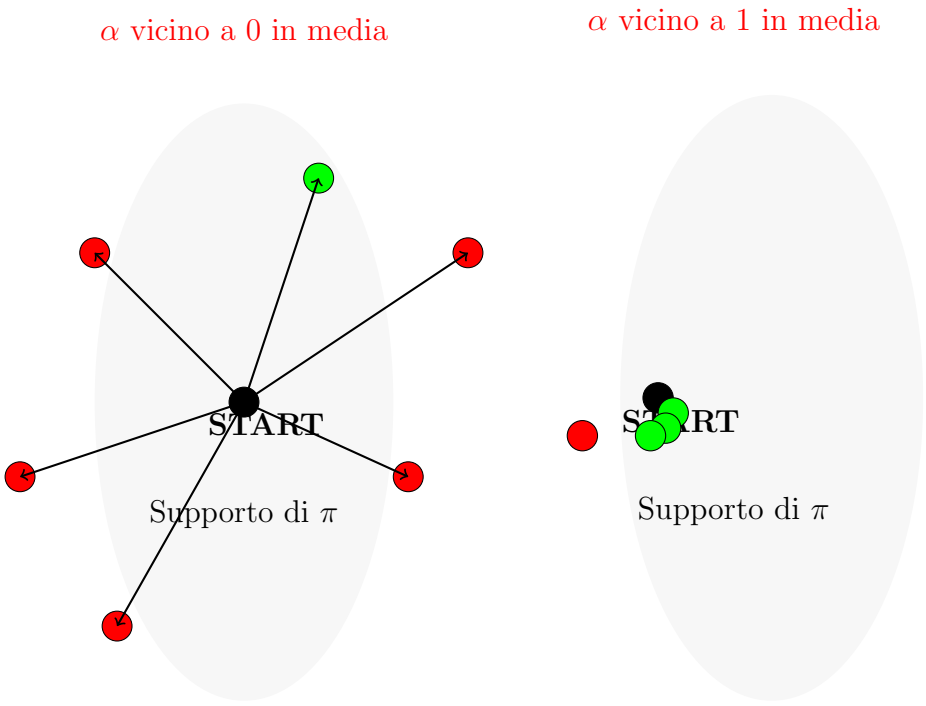


Figura 3.4: α diversi a confronto

Come illustrato in figura se α assume un valore vicino allo zero implica che la maggior parte delle proposte viene rifiutata e comporta un rallentamento nello

sviluppo della catena. Si troveranno quindi dei campioni che presentano un'alta correlazione e comporta che la catena esplori solo di una parte limitata dello spazio degli stati. Al contrario se α assume valori vicini a 1 quasi tutte le proposte vengono accettate.

3.2.2 Random Walk Sampler

In questo algoritmo i nuovi passi proposti derivano da una "random walk", ossia da dei passi aleatori a partire dallo stato precedente. Se inoltre la distribuzione di proposta è simmetrica, cioè $q(\mathbf{y} \mid \mathbf{x}) = q(\mathbf{x} \mid \mathbf{y})$, allora la probabilità di accettazione si semplifica notevolmente. Sia per esempio il passo aleatorio guidato da una distribuzione normale multivariata a media nulla:

$$q(\mathbf{y} \mid \mathbf{x}) = f(|\mathbf{y} - \mathbf{x}|) = q(\mathbf{x} \mid \mathbf{y})$$

$$\begin{cases} \mathbf{y} = X_t + \epsilon \\ \epsilon = \mathcal{N}(0, \Sigma) \end{cases} \quad \rightarrow q(\mathbf{y} \mid \mathbf{x}) = \mathcal{N}(\mathbf{x}, \Sigma)$$

La matrice di covarianza Σ contiene tutte le informazioni relative a come muoversi localmente attorno al punto attuale e la probabilità di accettazione diventa

$$\alpha(\mathbf{x}, \mathbf{y}) = \min \left\{ \frac{\pi(\mathbf{y})}{\pi(\mathbf{x})}, 1 \right\}$$

3.2.3 Burn-in

Un aspetto cruciale per il successo nell'implementazione dell'algoritmo Metropolis-Hastings o di qualsiasi altro campionatore MCMC è il numero di iterazioni o passi necessari affinché la catena raggiunga la stazionarietà. Tipicamente, i primi elementi vengono scartati di un numero che può variare di molto. Infatti se viene fatta una scelta inadeguata dei valori iniziali o della distribuzione proposta, può aumentare notevolmente il tempo di burn-in necessario, e un'area di ricerca attuale si concentra sulla possibilità di trovare un punto di partenza e una distribuzione proposta ottimali [5].
Si veda nella Sezione 3.3 come è stato valutato un valore ottimale del burn-in per una determinata distribuzione da campionare.

3.3 Campionamento da una distribuzione generica utilizzando Metropolis-Hastings

Vediamo ora un esempio implementativo che riguarda l'utilizzo dell'algoritmo Metropolis-Hastings per il campionamento di una distribuzione data dalla somma di due Normali, descritte dalla **funzione di target**

$$\pi(x) = \frac{1}{k}\left(\frac{1}{\sqrt{2\pi\sigma_1^2}}e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} + \frac{1}{\sqrt{2\pi\sigma_2^2}}e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}}\right)$$

Dove k è il fattore di normalizzazione.

$$y_1 \sim \mathcal{N}(\mu_1, \sigma_1^2) \qquad y_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$$

Nell'esempio $\mu_1 = 3$, $\sigma_1^2 = 1$, $\mu_1 = 15$, $\sigma_1^2 = 9$

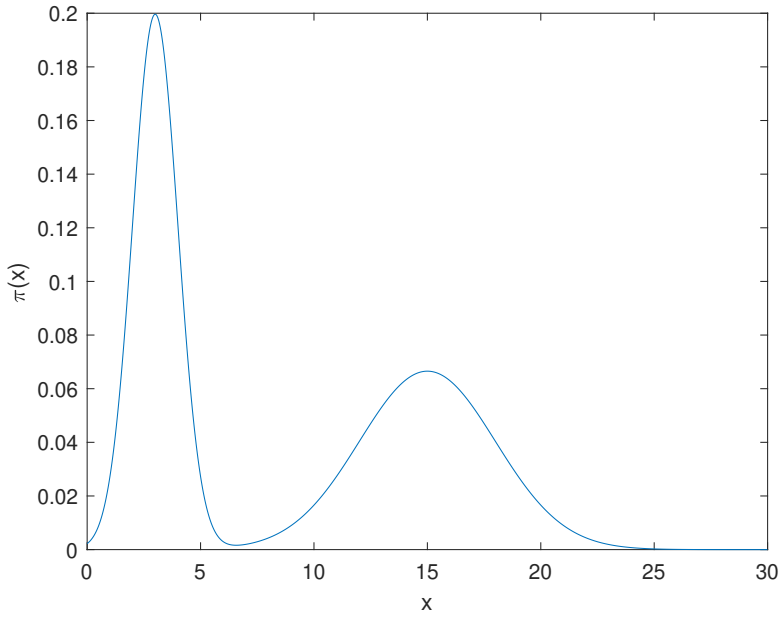


Figura 3.5: Distribuzione target $\pi(x)$

Scelgo poi una **densità di proposta** $q(y \mid x)$ che serve per proporre nuovi campioni nel processo di campionamento. Scelgo quindi un random walk sampler che utilizza una distribuzione uniforme centrata sullo stato precedente.

Si tratta di un random walk sampler perché la nuova proposta è fatta a partire dallo stato attuale a cui si aggiunge un passo aleatorio dato da una variabile

uniforme, definita sull'intervallo $[-b, b]$ dove b è il parametro.

$$\begin{cases} y = X_t + \epsilon \\ \epsilon = \mathcal{U}(-b, b) \end{cases}$$

Definita quindi come

$$q(y \mid x) = \begin{cases} \frac{1}{2b} & \text{se } y \in [-b, b] \\ 0 & \text{altrimenti} \end{cases}$$

Essendo in questo caso la densità di proposta simmetrica, come spiegato nella Sezione 3.2.2, la **probabilità di accettazione** si semplifica e diventa

$$\alpha(x, y) = \min \left\{ \frac{\pi(y)}{\pi(x)}, 1 \right\}$$

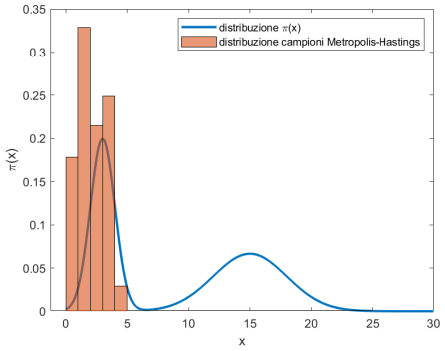
Per avere una visualizzazione della qualità del campionamento si realizza un plot in cui i campioni generati dall'algoritmo Metropolis-Hastings sono rappresentati da degli istogrammi. Nel codice realizzato ci sono due parametri che possono essere variati per valutare la qualità del campionamento: il parametro b della densità di proposta legato alla sua varianza e il valore del *burn-in*.

In primo luogo allora si cerca di trovare quale potrebbe essere il b ottimale per il problema. A tal fine si cerca di avere un tasso di accettazione che sia tra il 15% e il 50% [10]:

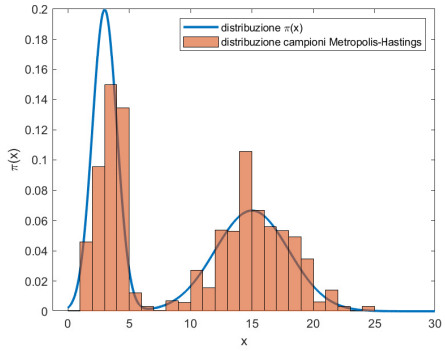
- tasso di accettazione troppo alto $> 50\%$: La catena di Markov fa proposte che sono vicine allo stato attuale, accettandole quasi tutte. L'esplorazione però è lenta perché si muove di piccoli passi e può non coprire l'intero supporto.

- tasso di accettazione troppo basso $< 15\%$: La catena fa proposte che sono lontane dallo stato attuale, rifutandole quasi tutte. Di conseguenza la convergenza della catena sarà lenta.

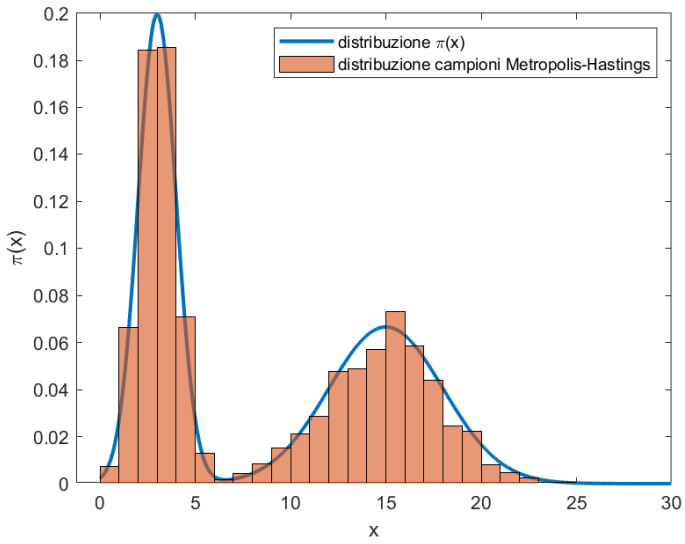
Vediamo allora alcuni esempi mantenendo il valore del burn-in pari a 1000.



(a) $b = 0.1$ rate= 97%



(b) $b = 50$ rate= 11%

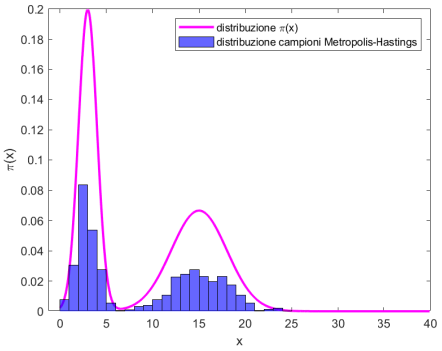


(c) $b = 12$ rate= 35%

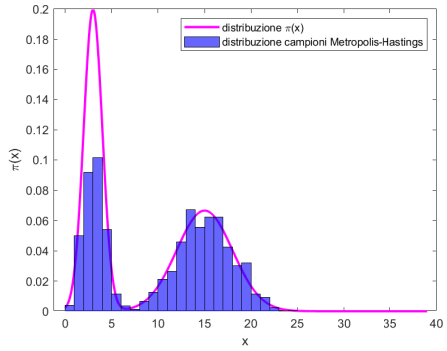
Figura 3.6: ricerca del valore ottimale b

Il valore di $b = 12$ trovato è un buon compromesso tra esplorazione dello spazio degli stati e il tempo di convergenza.

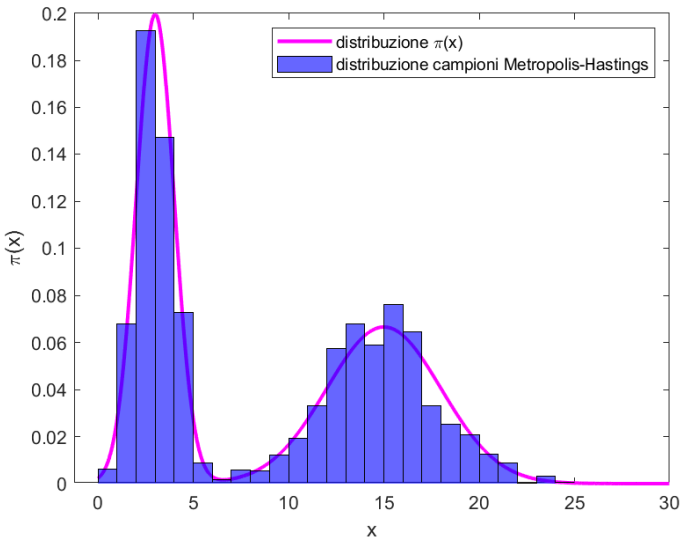
Un altro aspetto da considerare è l'impatto della lunghezza del periodo di burn-in. Tenendo quindi fissato i valore di b e il numero totale di campioni $n = 4000$, è utile esaminare gli effetti di diverse scelte di burn-in impostando valori rispettivamente pari a 10, 100 e 1000. È fondamentale selezionare uno stato iniziale che si discosti significativamente dalla distribuzione target per valutare efficacemente l'effetto del burn-in. Ad esempio, possiamo scegliere lo stato iniziale $X_0 = 300$.



(a) burn-in = 10



(b) burn-in= 100



(c) burn-in= 1000

Figura 3.7: ricerca del valore ottimale del burn-in

Si deduce che adottare un valore di burn-in più conservativo consenta una valutazione più accurata dei campioni generati dalla distribuzione target. Questo perché, se il burn-in è troppo basso e lo stato iniziale è molto distante dalla regione in cui si concentra la maggior parte della densità della distribuzione target, allora i campioni iniziali, che sono ancora vicini dallo stato iniziale, potrebbero non rappresentare adeguatamente la distribuzione target. In altre parole, un burn-in insufficiente può portare a campioni che non sono ancora completamente rappresentativi della distribuzione desiderata.

Segue il codice MATLAB utilizzato per eseguire il campionamento basato sull'implementazione descritta in precedenza.

```
%algoritmo metropolis hasings
%parametri
burn_in = 1000;
b = 10;
num = 4000;
X_0 = 1;
c = num - burn_in;
sigma1 = 1;
sigma2 = 3;
mu1 = 3;
mu2 = 15;
x = 0 : 0.01 : (0.01 *(c-1));
%normalizzazione della funzione di target
y = (1 / (sigma1 * sqrt(2 * pi))) * exp(-((x - mu1).^2) / (2 *
    sigma1^2)) + (1 / (sigma2 * sqrt(2 * pi))) * exp(-((x - mu2)
    .^2) / (2 * sigma2^2))
area = trapz(x, y);
y = y/ area;
%funzioni
target = @(x) 1/area*(1 / (sigma1 * sqrt(2 * pi))) * exp(-((x -
    mu1).^2) / (2 * sigma1^2)) + (1 / (sigma2 * sqrt(2 * pi))) *
     exp(-((x - mu2).^2) / (2 * sigma2^2));
accettazione = @(x) x + (2 * b) * (rand - 0.5);
%inizializzazione
samples = zeros(1, c)
samples(1) = X_0;
current = X_0;
%algoritmo Metropolis-Hastings
for i = 2 : num
    Y = accettazione(current);
    alpha = min(1, target(Y)/target(current));
    if rand < alpha
        current = Y;
         if i > burn_in
            accettate = accettate + 1;
        end
    end
    if i > burn_in
        samples(i-burn_in) = current;
    end
end
%plot
edges = 0 : 1: 25;
```

```
figure;
plot(x, y, 'LineWidth', 2);
hold on
histogram(samples, 30, 'Normalization', 'probability','BinEdges',
      edges);
legend('distribuzione \pi(x)', 'distribuzione campioni Metropolis
    -Hastings', 'Location', 'northeast')
hold off
xlabel('x');
ylabel('\pi(x)')
```

3.4 Esempio di stima bayesiana utilizzando Metropolis-Hastings

La stima bayesiana si applica quando si desidera determinare un parametro θ incerto, trattato come variabile stocastica. Lo scopo di questo esempio è quello di analizzare come applicare l'algoritmo Metropolis-Hastings a un problema di stima bayesiana.
Si consideri per esempio il problema di dover stimare il valore di una resistenza. Il valore certificato della resistenza è $R = 1\text{k}\Omega$ con una varianza di $\sigma = 70\Omega$, modellato con una gaussiana. Questa informazioni rappresenta l'informazione a priori, **prior**, indicata come p(θ):

$$p(\theta) \sim \mathcal{N}(1000, 70^2) \qquad p(\theta) = \frac{1}{\sqrt{2\pi}\,70} e^{-\frac{(\theta-1000)^2}{2\cdot 70^2}}$$

L'informazione rappresentata dal prior, nella stima bayesiana, viene integrata dall'informzaione proveniente dai dati ottenuti tramite un processo di misurazione. Questo processo di misura va modellato tramite la probabilità $p(y|\theta)$. Nell'esempio di interesse la likelihood deriva dall'informazione del valore della tensione ai capi della resitenza ottenuto utilizzando la legge di Ohm $V = R\,I$, dove V è il valore della tensione, R della resistenza e I della corrente. Si supponga nell'esempio che attraverso la resistenza passi una corrente $I = 1\,\text{mA}$ e che la tensione misurata sia affetta da un errore stocastico ϵ , distribuito come una Normale di media nulla e deviazione standar $0.005V$.

$$(y =)V = \theta I + \epsilon \qquad \epsilon \sim \mathcal{N}(0, 0.005^2)$$

Nel seguente esempio la likelihood è quindi data da:

$$p(\theta|y) \sim \mathcal{N}(\theta I, 0.005^2) \qquad p(\theta) = \frac{1}{\sqrt{2\pi}\,0.05} e^{-\frac{(\theta-\theta I)^2}{2\cdot 0.005^2}}$$

La distribuzione a posteriori o **posterior** è data dalla formula di Bayes ed è quindi proporzionale a

$$\pi(\theta|y) \propto p(\theta|y)p(\theta)$$

Essa rappresenta la distribuzione aggiornata del parametro dopo aver considerato i dati. Si ottiene combinando il prior con la likelihood.

Nella stima bayesiana il parametro che voglio stimare può essere ottenuto tramite lo stimatore a minima varianza che è risultato dell'integrazione del posterior

$$\hat{\theta} = \mathbb{E}(\theta|y) = \int \theta\, p_{\theta|y}(\theta|y) d\theta$$

L'algoritmo Metropolis-Hastings consente di stimare lo stesso valore ma senza il calcolo deterministico dell'integrale.

Ritornando all'esempio bisogna simulare il processo di misurazione della tensione ai capi della resistenza e per semplicità consideriamo essere un'unica misura y. Supponiamo che la resistenza in questione abbia un valore effettivo pari a 978Ω. Il processo di una misura può essere simulato come campione derivato da una gaussiana avente media pari a $978I$ e varianza 0.005^2.

Con la conoscenza di y si può procedere utilizzando l'algoritmo Metropolis-Hastings per campionare il posterior e poi ottenere un valore della stima della resistenza tramite Monte Carlo. Nell'esempio si ottiene una stima pari a 976 Ω. Il caso riportato, pur essendo semplice sia nei parametri scelti che nel modello, mostra come è possibile applicare in successione le tecniche affrontate in questa tesi attreverso un esempio dell'ingegneria dell'informazione. In particolare la sostituzione del calcolo deterministico con il metodo Monte Carlo che è reso possibile tramite il campionamento con l'algoritmo Metropolis-Hastings.

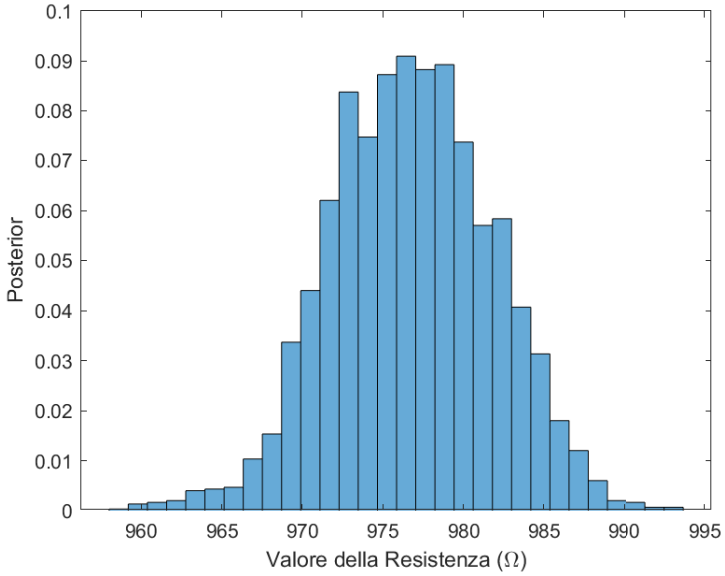


Figura 3.8: Campionamento del posterior tramite Metropolis-Hastings.

```
% Parametri iniziali
burn_in = 1000;
num = 4000;
c = num - burn_in;
 % valore iniziale della resistenza
theta = 1000;
%corrente
I = 0.001;
% Parametri della distribuzione prior e likelihood
mu_p = 1000;
sigma_p = 70;
mu_l = 978 * I;
sigma_l = 0.01;
%calcolo y misura della tensione
y = mvnrnd(mu_l, sigma_l^2, 1);
%funzioni
prior = @(theta) (1 / (sigma_p * sqrt(2 * pi))) * exp(-((theta -
     mu_p).^2) / (2 * sigma_p^2));
likelihood = @(theta) 1/ (sqrt(2 * pi) * 0.05) * exp(-((y - theta
      * I)^2) / (2 * sigma_l^2));
%inizializzazione
samples = zeros(1, c);
samples(1) = theta;
current = theta;
for i = 2 : num
    %densita' di proposta
```

```
25      Y = normrnd(current, 2);
        alpha = min(1, (prior(Y) * likelihood(Y))/(prior(current) *
         likelihood(current)));
        if rand < alpha
            current = Y;
        end
30      if i > burn_in
            samples(i-burn_in) = current;
        end
    end
    media_resistenza = mean(samples)
35  figure;
    histogram(samples, 30, 'Normalization', 'probability');
    xlabel('Valore della Resistenza (\Omega)');
    ylabel('Frequenza');
    title('Stima della Resistenza tramite Metropolis-Hastings');
```

Bibliografia

[1] George Marsaglia. «Random numbers fall mainly in the planes». In: *Proceedings of the National Academy of Sciences* 61.1 (1968), pp. 25–28. DOI: 10.1073/pnas.61.1.25. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.61.1.25. URL: https://www.pnas.org/doi/abs/10.1073/pnas.61.1.25.

[2] F James. «Monte Carlo theory and practice». In: *Reports on Progress in Physics* 43.9 (set. 1980), p. 1145. DOI: 10.1088/0034-4885/43/9/002. URL: https://dx.doi.org/10.1088/0034-4885/43/9/002.

[3] Nick Metropolis. «The beginning of the Monte Carlo Method». In: *Special Edition of Los Alamos Science*. 1987. URL: https://api.semanticscholar.org/CorpusID:18607470.

[4] P. Hellekalek. «Good random number generators are (not so) easy to find». In: *Mathematics and Computers in Simulation* 46.5 (1998), pp. 485–505. ISSN: 0378-4754. DOI: https://doi.org/10.1016/S0378-4754(98)00078-0. URL: https://www.sciencedirect.com/science/article/pii/S0378475498000780.

[5] B. Walsh. *Markov Chain Monte Carlo and Gibbs Sampling*. Apr. 2004. URL: http://nitro.biosci.arizona.edu/courses/EEB581-2004/handouts/Gibbs.pdf.

[6] Leonardo Angelini. *Tecniche Monte Carlo*. 2006.

[7] Reuven Y Rubinstein Dirk P Kroese. *Monte carlo methods*. John Wiley & Sons, Inc., 2012.

[8] Dirk Kroese et al. «Why the Monte Carlo method is so important today». In: *Wiley Interdisciplinary Reviews: Computational Statistics* 6 (nov. 2014). DOI: 10.1002/wics.1314.

[9] L. Finesso. *Lezioni di probabilità*. Progetto Libreria, 2019. ISBN: 9788831901185. URL: https://books.google.it/books?id=nMYCwwEACAAJ.

[10] J.S. Teixeira et al. «A new adaptive approach of the Metropolis-Hastings algorithm applied to structural damage identification using time domain data». In: *Applied Mathematical Modelling* 82 (2020), pp. 587–606. ISSN: 0307-904X. DOI: https://doi.org/10.1016/j.apm.2020.01.021. URL: https://www.sciencedirect.com/science/article/pii/S0307904X20300214.

[11] William L. Dunn e J. Kenneth Shultis. «Chapter 5 - Variance Reduction Techniques». In: *Exploring Monte Carlo Methods (Second Edition)*. A cura di William L. Dunn e J. Kenneth Shultis. Second Edition. Elsevier, 2023, pp. 151–187. ISBN: 978-0-12-819739-4. DOI: https://doi.org/10.1016/B978-0-12-819739-4.00013-5. URL: https://www.sciencedirect.com/science/article/pii/B9780128197394000135.

[12] Gianluigi Pillonetto. «Estimation and Filtering». 2023-2024.