Generatori di sequenze pseudocasuali

Manuela Aprile

Maria Chiara Fumi

Indice

- Concetti base e terminologia
- Random bit generator
- Pseudorandom bit generator
- Cenni di statistica
- Test Statistici

Concetti base e terminologia

- Numero **arbitrario** = numero qualunque
- Numero **casuale** = numero estratto da un insieme di valori equiprobabili
- Numero **pseudocasuale** = numero casuale generato da calcolatore

Requisiti di una sequenza casuale

- Periodo lungo
- Ripetibilità
- Sottosequenze non correlate
- Ordinamento interno non uniforme/numeri ben distribuiti

Generatori di random bit

- Random numbers VS random bits
- Hardware based
  - Bias e correlazione
- Software based
  - Maggiori difficoltà implementative
  - Maggiore esposizione ad attacchi
- De-skewing
  - Funzioni Hash

La generazione di "vere" sequenze di numeri casuali non è possibile su di un computer senza uno speciale hardware

Generatori di pseudorandom bit

- Un PRNG può essere definito come una struttura (S;; f; U; g)
- I cambiamenti di stato sono determinati dalla ricorrenza $s_n = f(s_{n-1})$ per n > 0
- L'output al passo n è : $u_n = g(s_n) \varepsilon \mathrm{U}$, i valori u_n sono i nostri "numeri random"
- S(i+P)=S(i)

Algoritmi di generazione sequenze

- Middle Square (John Von Neumann 1946)
- Lineare congruenziale (Lehemer 1951)
- Congruenza quadratica (Knuth 1981)

Middle Square

- a compreso tra 0 ed 1 numero pari di n cifre decimali;
- a^2 (doppia precisione);
- a^2* 10^(n/2);
- b=prime n cifre decimali (secondo elemento della sequenza)

Esempio:

a=0.5772156649 (n=10)

a^2 = 0.33317792380594919201

a^2 *10^5 => 33317.792380594919201

b=0.7923805949

Lineare Congruenziale

$$X_{n+1} = (aX_n + b) \bmod m, \quad n \geq 0$$

- Successione definita per ricorrenza
- Insieme dei valori finiti [0,m-1]

$$y_n = \frac{x_n}{(m-1)}$$

- Sequenza distribuita nell'intervallo [0,1]
- Se b=0 è detto **moltiplicativo**

$$x_i = (a_1 x_{i-1} + \cdots + a_k x_{i-k}) \bmod m,$$

Esempio:

Parametri iniziali: $X0 = 3, m = 9, b = 2, a = 7$

$X1 = (7 \leftarrow 3+2) \ mod \ 9 = \underline{\mathbf{5}}$

$X2 = (7 \leftarrow 5+2) \ mod \ 9 = 1$

$X3 = (7 \leftarrow 1+2) \ mod \ 9 = 0$

$X4 = (7 \leftarrow 0+2) \ mod \ 9 = 2$

$X5 = (7 \leftarrow 2+2) \ mod \ 9 = 7$

$X6 = (7 \leftarrow 7+2) \ mod \ 9 = 6$

$X7 = (7 \leftarrow 6+2) \ mod \ 9 = 8$

Congruenza Quadratica

$$X_n = (aX_{n-1}^2 + bX_{n-1} + c) \ mod \ m$$

<u>Esempio:</u>

Parametri iniziali: $X0$ = 2, a=2, b=3, c=1, m=4

$X1$ = (2*4+3*2+1) mod 4 = <u>3</u>
$X2$ = (2*9+3*3+1) mod 4 = 0
$X3$ = (2*16+3*4+1) mod 4 = 1
$X4$ = (2*1+3*1+1) mod 4 = 2
<u>**$X5$ = (2*4+3*2+1) mod 4 = 3**</u>

Esempi di generatori

- ANSI X9.17
  - Generazione chiavi e inizializzazione vettori per l’algoritmo DES
- FIPS 186
  - Generazione parametri segreti(chiavi e firme) per DSA
  - Funzioni one-way SHA-1 o DES

Applicazioni e librerie

- Simulazione
- Protocolli di comunicazione sicura
- Crittografia
- …

- Java, Matlab, Excel, …

Requisiti di un generatore

- Periodo lungo
- Portabilità
- Efficienza
- Ripetibilità
- Jumping Ahead

Cenni di statistica

- Distribuzione normale
- Distribuzione uniforme
- Altre distribuzioni
- Funzione inversa di distribuzione
- Riduzione alla distribuzione uniforme

Distribuzione Normale

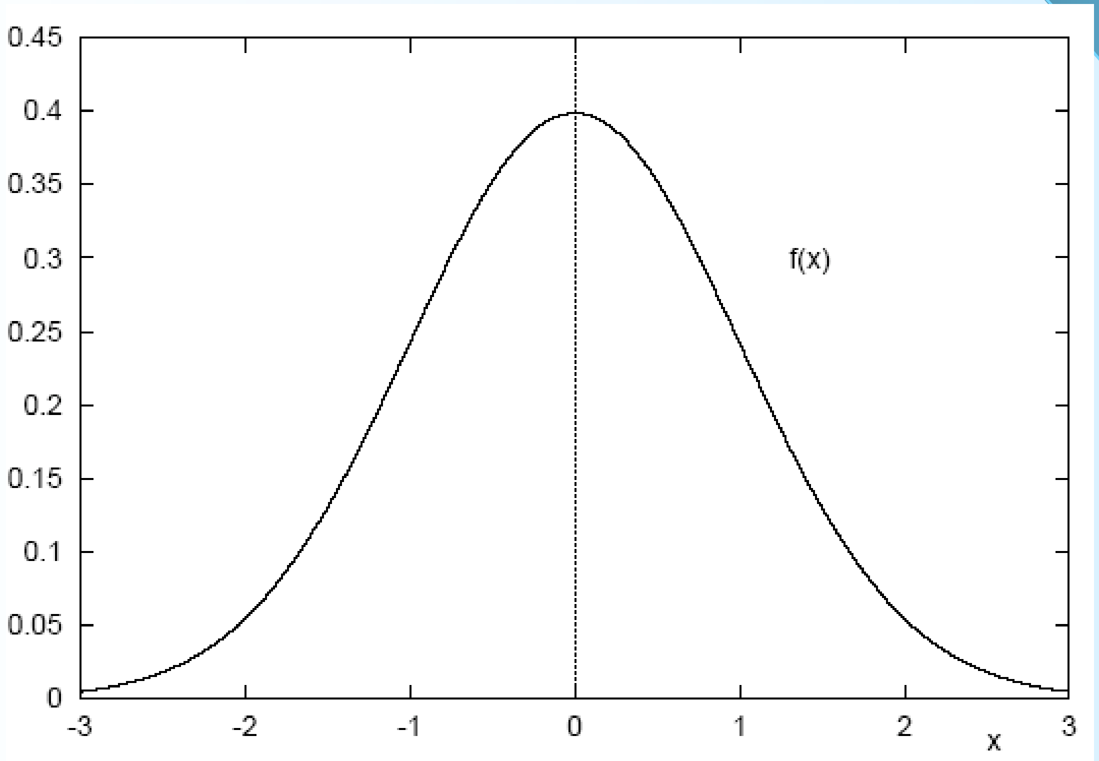
X variabile aleatoria N(μ,__)

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], -\infty < x <$$

Somma di più variabili aleatorie con stesso valor medio e stessa varianza

Distribuzione Uniforme

X variabile aleatoria N(0,1)



Altre distribuzioni

- Chi-square
- Poisson
- Bernoulli
- Esponenziale
- …

Non esistono ancora efficienti algoritmi in grado di generare sequenze di numeri con distribuzioni diverse da quelle **uniformi** senza utilizzarle

Funzione di distribuzione inversa

- Passare da una sequenza con una data distribuzione ad una differente

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

- X = F__(U) monotona non decrescente di U

Riduzione alla distribuzione uniforme

Distribuzione normale $\Longleftrightarrow$ Distribuzione uniforme

$$
\begin{tikzcd}
X \approx N(\mu,\sigma) \arrow[r, bend right, "Z = (X - \mu)/\sigma"'] & Z \approx N(0,1) \arrow[l, bend right, "X = \sigma Z + \mu"']
\end{tikzcd}
$$

Postulati di Golomb

- Primo tentativo di test dei generatori

I. Il numero di 0 e 1differiscono al più di 1 nel ciclo della sequenza s

II. Nel ciclo di S almeno la metà delle *run* hanno lunghezza 1, almeno un quarto hanno lunghezza 2, …

III. L’auto-correlazione C(t) è two-valued

$$N \cdot C(t) = \sum_{i=0}^{N-1} (2s_i - 1) \cdot (2s_{i+t} - 1) = \begin{cases} N, & \text{if } t = 0, \\ K, & \text{if } 1 \le t \le N-1. \end{cases}$$

Test

- Verifica della qualità del generatore
- Consente di accettare o rifiutare l'ipotesi statistica Ho, circa la distribuzione della sequenza
- Significance level _

  probabilità di rifiutare Ho quando questo è vero

  (Type 1 Error, Type 2 Error)
- La conclusione di ogni test è solo probabilistica

- One-sided test:

  - $x_{\square}$ scelta in modo tale che $P(X>x_{\square})=\square$
  - se $X_s>x_{\square}$ la sequenza fallisce il test

- Two-sided test:

  - scelta $x_{\square}$ in modo tale che $P(X>x_{\square})= P(X<-x_{\square})= \square/2$.
  - se $X_s>x_{\square}$ o $X_s<-x_{\square}$ la sequenza fallisce il test

Frequency test (Monobit test)

- Determina se il numero di 0 e 1 è circa lo stesso

$$X_1 = \frac{(n_0 - n_1)^2}{n}$$

- con n0 e n1 indichiamo il numero di 0 e 1

Esempio

S= 11100 01100 01000 10100 11101 11100 10010 01001

n0=84 , n1= 76 X1=0.4

Serial test (Two bit test)

- Determina se il numero di occorrenze di 00, 01, 10, 11 come sottosequenze di S sia pressoché uguale

$$X_2 = \frac{4}{n-1}\left(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2\right) - \frac{2}{n}\left(n_0^2 + n_1^2\right) + 1$$

n00+n01+n10+n11= (n-1)

Esempio

S= 11100 01100 01000 10100 11101 11100 10010 01001

n00=44, n01=40, n10=40, n11=35, X2=0.6252

Poker test

- k sottostringhe di lunghezza m

$$k = \lfloor \frac{n}{m} \rfloor \geq 5 \cdot (2^m)$$

- Si determina se ogni sottosequenza lunga m appare lo stesso numero di volte in s

$$X_3 = \frac{2^m}{k} \left(\sum_{i=1}^{2^m} n_i^2 \right) - k$$

Esempio

S= 11100 01100 01000 10100 11101 11100 10010 01001

m=3, k=53, X3=9.6415

000 → 5, 001 → 10, 010 → 6, …

Run test

- Determina se il numero di *run* di varia lunghezza è compatibile con quanto atteso per una sequenza casuale.

$$X_4 = \sum_{i=1}^{k} \frac{(B_i - e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(G_i - e_i)^2}{e_i}$$

- Numero gap di lunghezza *i* in una sequenza casuale lunga m: $e_i = (n-i+3)/2^{i+2}$

Esempio

S= 11100 01100 01000 10100 11101 11100 10010 01001

e1=20.25, e2=10.0625, e3=5, k=3, X4=31.7913

25 Blocks lunghi 1, 4 lunghi 2, 5 lunghi 3

8 Gaps lunghi 1, 20 lunghi 2, 12 lunghi 3

Autocorrelation test

- Analizza la correlazione tra la sequenza s e la sua copia ritardata

$$X_5 = 2\left(A(d) - \frac{n-d}{2}\right) / \sqrt{n-d}$$

$$A(d) = \sum_{i=0}^{n-d-1} s_i \oplus s_{i+d}.$$

- Valore di shift $1 \leq d \leq \lfloor n/2 \rfloor$

Esempio

S= 11100 01100 01000 10100 11101 11100 10010 01001

d=8, allora A(8)=100, X5=3.8933

FIPS 140-1

- Vengono specificati quali sono gli intervalli di validità che una sequenza deve soddisfare perché il suo generatore superi i test
- Sia s una stringa di 20000 bits
  - monobit test: $9654<n1<10346$
  - poker test: $m=4$, $1.03<X3<57.4$
  - runs test:

Length of run	Required interval
1	2267 – 2733
2	1079 – 1421
3	502 – 748
4	223 – 402
5	90 – 223
6	90 – 223

Maurer’s universal statistical test

- Non si può comprimere l’output di un generatore casuale senza perdita di informazione.
- È in grado di individuare ogni possibile difetto di un generatore.
- Richiede sequenze di output più lunghe.

Conclusioni

- Un PRBG che supera tutti i test è detto *generatore pseudorandom di bit* *crittograficamente sicuro*

"Costruire un generatore che superi tutti i test è un sogno impossibile" (Pierre L'Ecuyer)