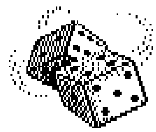
Distribuzioni di probabilità: generazione di campioni

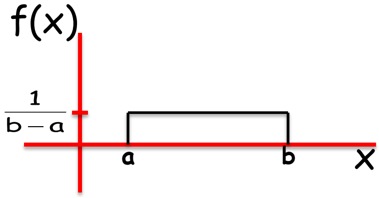


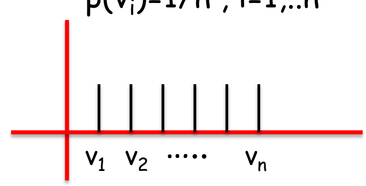
Dario Maio

http://bias.csr.unibo.it/maio/

Distribuzione uniforme

- In quasi tutti i linguaggi di programmazione è disponibile una funzione in grado di generare un numero distribuito uniformemente tra 0 e 1.
- Nel linguaggio C la funzione stdlib.h rand() genera un intero casuale maggiore o uguale a 0 e minore o uguale a RAND_MAX.
- Si possono pertanto facilmente generare interi o float, uniformemente distribuiti all'interno di un intervallo [min,max].





Distribuzione uniforme: esempi

- **Esempio 1:**

```
/* Restituisce un numero pseudo-casuale
   fra 0 e 1 (estremi compresi) */
float rnd()
{
   return (float)rand() / RAND_MAX;
}
```

- **Esempio 2:**

```
/* Restituisce un intero pseudo-casuale
   fra n1 e n2 (estremi compresi) */
int random2(int n1,int n2)
{
  return rand()*(n2-n1+1)/(RAND_MAX+1) + n1;
}
```


Generazione numeri casuali in C#

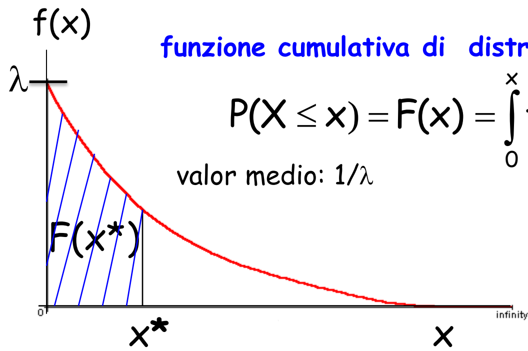
- Si fa uso della classe **Random**, ad esempio:

```
// instanziazione della classe
Random myRandomClass = new Random();
// definizione di un intero
int myRandomNumber;
// ..........................
// ..........................
// genera un intero fra 1 e 2,147,483,647
myRandomNumber = myRandomClass.Next();
// ..........................
// genera un intero >=1 e < 7  (lancio di un dado)
myRandomNumber = myRandomClass.Next(1,7);
// ..........................
// genera un intero >=0 e < 7
myRandomNumber = myRandomClass.Next(7);
```

Distribuzione esponenziale (1)

$$f(x) = \lambda e^{-\lambda x} \quad , x > 0 \qquad (\text{con } \lambda > 0)$$
$$\quad = 0 \qquad , x \le 0$$

funzione densità di probabilità



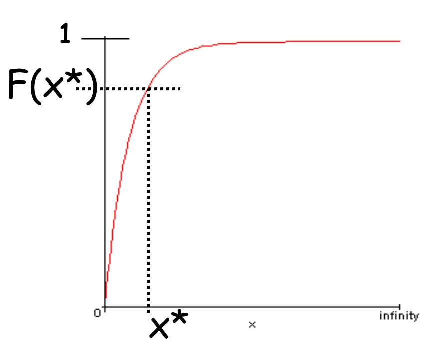
funzione cumulativa di distribuzione di probabilità

$$P(X \le x) = F(x) = \int_0^x f(u)du = 1 - e^{-\lambda x}$$

valor medio: 1/λ

Distribuzione esponenziale (2)

$$F(x) = 1 - e^{-\lambda x}$$



Sia r un numero generato in modo pseudo-casuale nell'intervallo (0,1]

$$r = 1 - e^{-\lambda x^*}$$

$$x^* = -\frac{1}{\lambda} \cdot \ln r$$

generazione di un campione

N.B. Si deve escludere il caso r=0

Il metodo di trasformazione (1)

- Per una variabile aleatoria distribuita uniformemente nell'intervallo (0,1), si ha che la probabilità di generare un numero compreso fra x e x+dx vale:

$$p(x)dx = \begin{cases} dx & 0 < x < 1 \\ 0 & \text{altrimenti} \end{cases}$$

- Si consideri una funzione y(x); si ha: $p(y) = p(x) \cdot \left| \frac{dx}{dy} \right|$

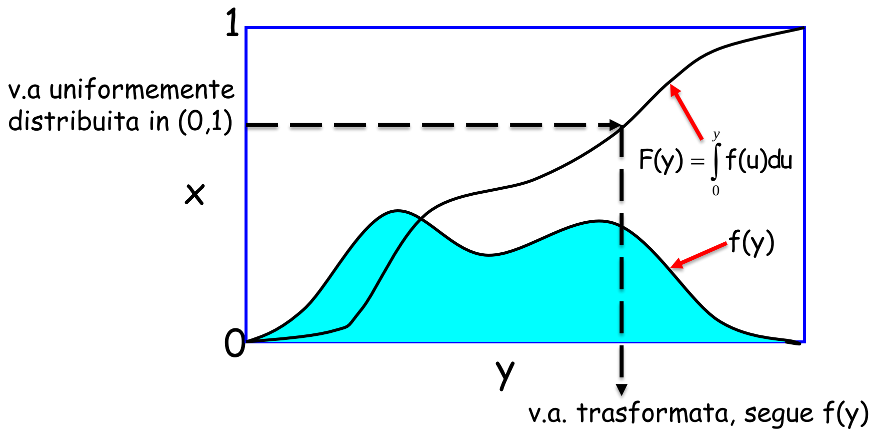
- Nota una funzione densità di probabilità f(y)=p(y), per generare campioni che seguono tale distribuzione, si deve risolvere l'eq. differenziale:

$$\frac{dx}{dy} = f(y)$$

e la soluzione è proprio: $x = F(y) = \int f(y)dy$

- pertanto la trasformazione da una variabile uniformemente distribuita in (0,1) in una variabile distribuita secondo f(y) è la funzione inversa F^{-1}.
- N.B. Questo metodo è applicabile ogni volta che la funzione inversa F^{-1} è calcolabile analiticamente o numericamente.

Il metodo di trasformazione (2)



Distribuzione di Erlang

$$f(x;\alpha,\lambda)=\frac{1}{\Gamma(\alpha)(1/\lambda)^{\alpha}}x^{(\alpha-1)}e^{-\lambda x} \quad x>0$$

valor medio: α/λ

dove α è un intero positivo e $\Gamma(\alpha)$ è la funzione definita come:

$$\Gamma(\alpha)=\int_0^{\infty}u^{\alpha-1}e^{-u}du$$

Per generare un campione è utile ricordare che una v.a. che segue la distribuzione di Erlang è la somma di α v.a. indipendenti che seguono una stessa distribuzione esponenziale.

N.B. quando α =1, si ottiene la distribuzione esponenziale.

Generalizzazione del metodo

- Se $p(x_1, x_2, \ldots, x_n)$ è la densità di probabilità congiunta di n variabili aleatorie e se $y_1, y_2, \ldots y_n$, sono funzioni di tutte le x, allora si ha:

$$p(y_1, y_2, \ldots, y_n)dy_1dy_2 \ldots dy_n = p(x_1, x_2, \ldots, x_n)\left|\frac{\partial(x_1,x_2,..x_n)}{\partial(y_1,y_2,..y_n)}\right| dy_1dy_2 \ldots dy_n$$

essendo $\left|\frac{\partial(x_1,x_2,..x_n)}{\partial(y_1,y_2,..y_n)}\right|$ il determinante Jacobiano.

- Un'importante applicazione è rappresentata dalla **tecnica di Box-Muller** per la generazione di campioni appartenenti alla distribuzione normale.

Distribuzione normale (1)

$$f(y)dy = \frac{1}{2\pi} e^{-y^2/2} dy$$

la funzione cumulativa F(y) non può essere risolta per y in formula chiusa.

- Siano X_1 e X_2 v.a. uniformemente distribuite in (0,1); si considerino le due funzioni y_1, y_2:

$$y_1 = \sqrt{-2\ln x_1}\cos 2\pi x_2$$
$$y_2 = \sqrt{-2\ln x_1}\sin 2\pi x_2$$

$$x_1 = \exp\left[-\frac{1}{2}\left(y_1^2 + y_2^2\right)\right]$$
$$x_2 = \frac{1}{2\pi}\arctan\frac{y_2}{y_1}$$

$$\begin{vmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \end{vmatrix} = -\left[\frac{1}{\sqrt{2\pi}} e^{-y_1^2/2} dy\right]\left[\frac{1}{\sqrt{2\pi}} e^{-y_2^2/2} dy\right]$$

Y_1 e Y_2 sono v.a. indipendenti e seguono una distribuzione normale.

Distribuzione normale (2)

- Conviene adottare un accorgimento: invece che considerare x_1 e x_2 come le coordinate di un punto random nel quadrato di lato unitario, scegliamo v_1 e v_2 rispettivamente come l'ordinata e l'ascissa di un punto random nel cerchio di raggio unitario centrato nell'origine.

- Allora $R^2 = v_1^2 + v_2^2$ è una v.a. uniformemente distribuita che può essere usata per x_1, mentre l'angolo che (v_1, v_2) forma con l'asse v_1 può essere usato come angolo random $2\pi x_2$. Il vantaggio consiste nel fatto che il coseno e il seno nella precedente formula possono essere scritti rispettivamente come:

$$\frac{v_1}{\sqrt{R^2}} \quad \text{e} \quad \frac{v_2}{\sqrt{R^2}}$$ evitando chiamate a funzioni trigonometriche.

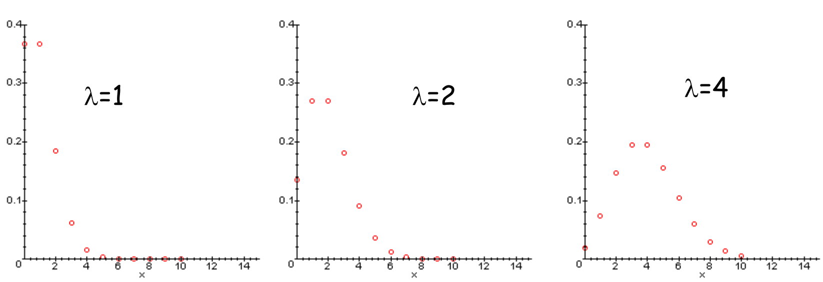
Distribuzione normale (3)

```
#include <math.h>
float gaussdev(long *idum)
/* Restituisce un campione che segue distr. normale con v.m. 0 e varianza 1,
usando ran1(idum)che genera un numero in (0,1) */
{ float ran1(long *idum);
  static int iset=0; static float gset;
  float fac,rsq,v1,v2;
  if (iset == 0)
     {/* è necessario generare */
          do { v1=2.0*ran1(idum)-1.0;
               v2=2.0*ran1(idum)-1.0;
              rsq=v1*v1+v2*v2;
              /* controlla se interno al cerchio */
             } while (rsq >= 1.0 || rsq == 0);
     fac=sqrt(-2.0*log(rsq)/rsq);
     /* applica la trasformazione Box-Muller; restituisci
     un valore e salva l'altro per la prossima chiamata */
     gset=v1*fac; iset=1; /* set flag */
     return v2*fac;
     } else { iset=0; return gset; }
}
```


Distribuzione di Poisson (1)

$$p(x;\lambda) = \begin{cases} \dfrac{e^{-\lambda}\lambda^{x}}{x!} & x=0,1,2\ldots;\ \lambda>0 \\ 0 & \text{altrove} \end{cases}$$

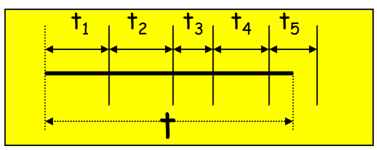
valor medio: λ



Distribuzione di Poisson (2)

- Posto $\lambda = \nu t$, essendo ν la frequenza costante di occorrenze, $p(x;\ \lambda)$ rappresenta la probabilità di avere esattamente x occorrenze in un intervallo di lunghezza t.
- Ricordando che l'intervallo di tempo che intercorre fra due occorrenze indipendenti di Poisson è una v.a. che segue una distribuzione esponenziale, un campione x appartenente alla distribuzione di Poisson può essere ottenuto generando successivamente valori da una distribuzione esponenziale e arrestando la generazione quando la somma di x+1 valori eccede la prescritta lunghezza di tempo t.

Esempio di generazione di un valore x=4



Distribuzione binomiale (1)

$$p(x;n,p) = \frac{n!}{(n-x)!\,x!} p^x (1-p)^{n-x} \quad x = 0,1,2,.., n$$

- Si ricorda che per n=1, degenera nella funzione di probabilità di Bernoulli

$$\begin{aligned} p(x;p) &= p^x (1-p)^{1-x} \quad x = 0,1 \\ &= 0 \quad \text{altrove} \end{aligned}$$

- Una v.a. binomiale può essere vista come la somma di n lanci da un processo di Bernoulli descritto da:

$$Y = \begin{cases} 1 \text{ con probabilità } p \\ 0 \text{ con probabilità } (1-p) \end{cases}$$

Distribuzione binomiale (2)

- Possiamo generare un campione appartenente a una distribuzione binomiale generando n valori della v.a. Y, ciascuno così determinato:

$$Y = \begin{cases} 1 \text{ se } 0 \leq r \leq p \\ 0 \text{ se } p < r \leq 1 \end{cases}$$

dove r è un numero uniformemente distribuito in [0,1].

- La somma degli 1 generati nella sequenza di n valori rappresenta un valore random binomiale.

Distribuzione di Zipf

- Spesso è necessario modellare fenomeni reali con distribuzioni discrete "**skewed**". Si pensi ad esempio alla generazione di valori di un attributo di una relazione, in cui l'ipotesi di uniformità non è sensata.

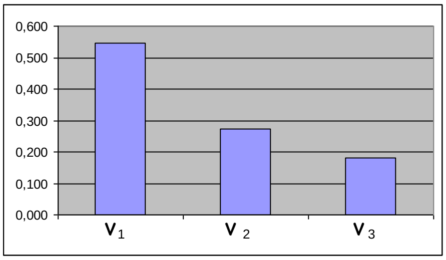
$$p(v_i) = \frac{i^{-z}}{\sum_{j=1}^{n} j^{-z}}$$

- v_1 è il valore più probabile.
- Per z=0 si ottiene la distribuzione uniforme.

Esempi distribuzione di Zipf

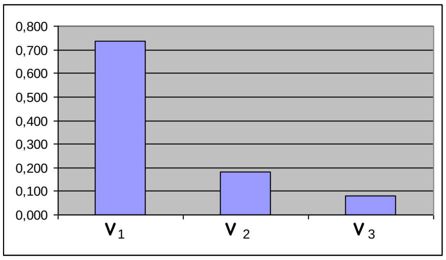
- Esempio z=1, n=3

$$p(v_1) = \frac{1}{1+1/2+1/3} = \frac{6}{11}$$
$$p(v_2) = \frac{1/2}{1+1/2+1/3} = \frac{3}{11}$$
$$p(v_3) = \frac{1/3}{1+1/2+1/3} = \frac{2}{11}$$



- Esempio z=2, n=3

$$p(v_1) = \frac{1}{1+1/4+1/9} = \frac{36}{49}$$
$$p(v_2) = \frac{1/4}{1+1/4+1/9} = \frac{9}{49}$$
$$p(v_3) = \frac{1/9}{1+1/4+1/9} = \frac{4}{49}$$



Generazione distribuzione di Zipf

- Per generare campioni appartenenti a una distribuzione di Zipf con parametri z e n, è sufficiente considerare un vettore F di n+1 numeri, con:

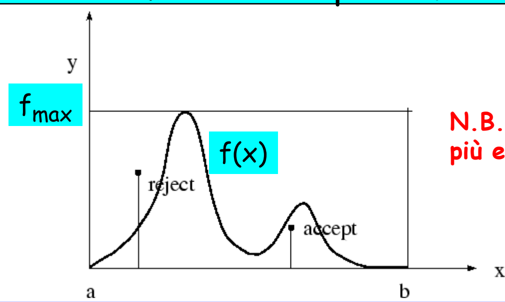
$$F_i = \sum_{j=1}^{i} p(v_j) \quad i = 1,..,n \qquad F_0 = 0$$

- Viene generato un valore r uniformemente distribuito nell'intervallo [0,1], che determina il campione v_i tale per cui $F_{i-1} < r \leq F_i$, avendo posto $F_0=0$.

Rejection method

- Introdotto da von Neumann non richiede il calcolo della inversa F^{-1}, ed è applicabile per lo più a ogni f(x).
- Sia [a,b] l'intervallo di valori della v.a., f_{max} il valore massimo della funzione densità di probabilità f(x). L'algoritmo di generazione è:

1. **Genera una coppia di valori uniformemente distribuiti:**
   $x \in [a,b]$ e $y \in [0, f_{max}]$
2. **Se $y \leq f(x)$ allora accetta x come campione valido altrimenti torna al passo 1.**



N.B.: esistono varianti più efficienti del metodo