Performance Modeling of Computer Systems and Networks

Prof. Vittoria de Nitto Personè

Generating Discrete Random Variates

Università degli studi di Roma Tor Vergata
Department of Civil Engineering and Computer Science Engineering

1

Prerequisite

We assume the knowledge of discrete random variables (sect.6.1).
In particular:

- *Equilakely*(a,b)
- *Geometric*(p)
- *Bernoulli*(p)
- *Binomial*(n,p)
- *Pascal*(n,p)
- *Poisson*(μ)

2

sis2 è inventory system, usa equilikely.

sis2.c

```
#include <stdio.h>
#include "rng.h"

#define MINIMUM   20
#define MAXIMUM   80
#define STOP     100     /* 100 weeks = about 2 years*/
#define sqr(x)    ((x) * (x))

long Equilikely(long a, long b)
{  return (a + (long) ((b - a + 1) * Random()));}

long GetDemand(void)
{
        return (Equilikely(10, 50)); }
```

3

Noi generiamo un valore tra 0 e 1, e lo trasformiamo a seconda della variabile.

ssq2.c distribution-driven simulation

```
#include <stdio.h>
#include <math.h>
#include "rng.h"
#define LAST          10000L    /* number of jobs processed */
#define START         0.0

double Exponential(double m)          /* ------*
{return (-m * log(1.0 - Random())); }    m > 0.0
                                        ------ */
double Uniform(double a, double b)     /* ------*
{return (a + (b - a) * Random());    }    a < b
                                       * ------*/
          double GetArrival(void)
        {static double arrival = START;
          arrival += Exponential(2.0);
               return (arrival);}

          double GetService(void)
        {return (Uniform(1.0, 2.0));}
```

Prof. Vittoria de Nitto Personè 4

4

X è v.a., F è la cumulativa (funzione di distribuzione).
Esiste F* che sarebbe "funzione inversa", ma formalmente non lo è. Perchè?
Perchè se voglio passare dal continuo al discreto, molti punti continui potrebbero convergere nello stesso valore discreto.

Preliminary Definitions

X random variable, $F(\cdot)$ is the cdf of X

The *inverse distribution function* (idf) of X is the function

$F^* : (0, 1) \rightarrow \chi$, $\forall\, u \in (0, 1)$

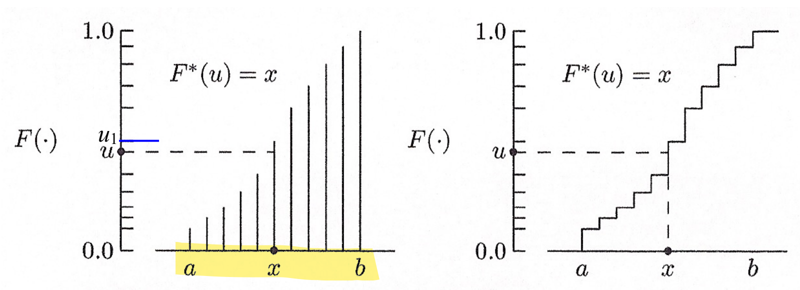
$$F^*(u) = \min_x \{x : u < F(x)\}$$

that is, if $F^*(u)=x$, x is the smallest possible value of X for which $F(x)$ is greater than u

5

Sull'asse 'x' ho insieme discreto. Prendo un x, faccio F(x) = u1 > u. Ora faccio l'inverso,
F*(u) = "x più piccola tale che F(x) > u ". Ciò è vero se prendo sulle ascisse proprio 'x', perchè F(x) = u1 > u.
In pratica, quando faccio l'inverso di un certo "u", devo prendere la x sulle ascisse più piccola tale che, F(x) < u minore stretto.

- $\chi = \{a, a + 1, \ldots, b\}$, where b may be ∞, $F(\cdot)$ is the cdf of X,
- $F(x)=\text{Prob}\{X \leq x\}=u_1 > u$ $\qquad F^*(u) = \min_x \{x : u < F(x)\}$



Theorem
- if $u < F(a)$, $F^*(u) = a$
- else $F^*(u) = x$ where $x \in \chi$ is the unique possible value of X for which $F(x - 1) \leq u < F(x)$

6

Il teorema ci dice che, presa un'ascissa "a" (che non è per forza il primo punto sull'asse x).
Se F(a) > u, allora F*(u) = a.
Altrimenti prendo un'ascissa tale che F(x' -1) <= u < F(x)

L'algoritmo 1 presenta un modo "dummy" per calcolare questa inversa.
Sostanzialmente, partendo dal minimo sulle ascisse 'a', io incremento finchè non trovo il primo valore di x tale che F(x) > u. Allora ritorno questa 'x', ovvero x = F*(u).
L'aspetto negativo è la lentezza delle iterazioni, poichè parto sempre dall'inizio.

Algorithm 1

```
x = a;
while (F(x) <= u)
        x++;
return x;            /*x is F*(u)*/
```

Average case analysis:
- let Y be the number of `while loop` passes
- $Y = X - a$
- $\mathrm{E}[Y] = \mathrm{E}[X{-}a] = \mathrm{E}[X] - a = \mu - a$

Linear search algorithm !

qui partiamo dalla "moda" (un valore più centrale), muovendoci con ricerca binaria.

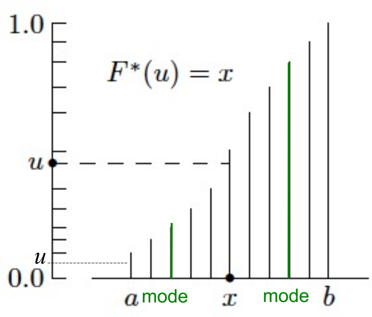
Idea: start at a more likely point

For $\chi=\{a, a+1, \ldots, b\}$, a more efficient linear search algorithm defines $F^*(u)$

Algorithm 2

```
x = mode;           /*initialize with the mode of X */
if (F(x) <= u)
    while (F(x) <= u)
        x++;    avanti
else if (F(a) <= u)
        while (F(x-1) > u)
          x--;  indietro
    else
        x = a;
return x;           /*x is F*(u)*/
```

For large χ, consider binary search



alcuni esempi di inversi calcolate.

Idf Examples

- In some cases $F^*(u)$ can be determined explicitly
- If X is *Bernoulli*(*p*) and $F(x) = u$,
  then x=0 iff $0 < u < 1$-p

$$F^*(u) = \begin{cases} 0 & 0 < u < 1 - p \\ 1 & 1 - p \le u < 1 \end{cases}$$

Random Variate Generation By Inversion

- X is a discrete random variable with idf $F^*(\cdot)$ deve essere esplicita
- continuous random variable U is *Uniform*(0,1)
- Z is the discrete random variable defined by $Z = F^*(U)$

Theorem
Z and X are identically distributed

this Theorem allows any discrete random variable (with known idf) to be generated with one call to `Random()`

Algorithm 3

```
u = Random();
return F*(u);
```

Ho questa funzione densità f(x).

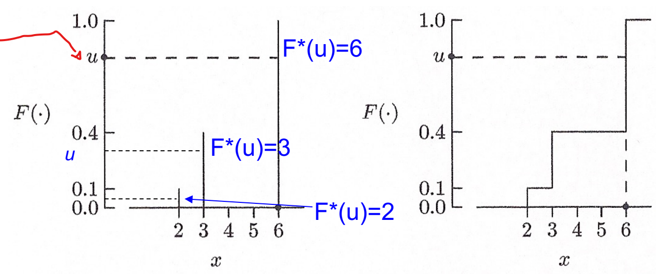
Inversion Examples

- Consider X with pdf

$$f(x) = \begin{cases} 0.1 & x = 2 \\ 0.3 & x = 3 \\ 0.6 & x = 6 \end{cases}$$

- The cdf for X is plotted using two formats

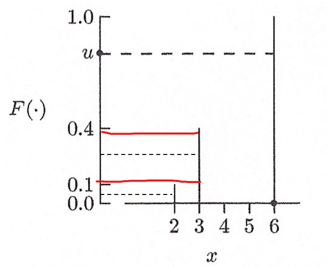
Nel grafico a sinistra, parto dall'asse y e ritorna il valore sulle x che si interseca con la 'u'. Ad esempio, per "u" vicino a 1.0 interseca F*(u) = 6



11


```
if (u < 0.1)
    return 2;
else if (u < 0.4)
        return 3;
    else
        return 6;
```

I valori tra 0.1 e 0.4 vanno tutti in x = 3



This algorithm returns
2 with probability 0.1,
3 with probability 0.3 and
6 with probability 0.6.
This corresponds to the pdf of X.

more efficiency: check the ranges for *u* associated with $x = 6$ first (the mode), then $x = 3$, then $x = 2$

- problems may arise when $|\mathcal{X}|$ is large or infinite

12

More inversion examples

Generating a *Bernoulli*(*p*) random variate

```
u = Random();
if (u < 1-p)
        return 0;
else
        return 1;
```

Generating an *Equilikely*(*a*,*b*) random variate

```
u = Random();
return a + (long) (u * (b - a + 1));
```


Library `rvgs`

Parliamo di "variate", non "variabili", perchè le generiamo noi. (Pseudo random by Lehmer)

- Includes 6 discrete random variate generators (as below) and 7 continuous random variate generators
  - `long Bernoulli(double p)`
  - `long Binomial(long n, double p)`
  - `long Equilikely(long a, long b)`
  - `long Geometric(double p)`
  - `long Pascal(long n, double p)`
  - `long Poisson(double μ)`
- Functions Bernoulli, Equilikely, Geometric use inversion; essentially ideal
- Functions Binomial, Pascal, Poisson do not use inversion

Library `rvms`

densità, cumulativa, inversa

- Provides accurate pdf, cdf, idf functions for many random variates
- Idfs can be used to generate random variates by inversion
- Functions `idfBinomial`, `idfPascal`, `idfPoisson` may have high marginal execution times
- Not recommended when many observations are needed due to time inefficiency
- Array of cdf values with inversion may be preferred

Truncation

Sometimes, the realistic values of a variable are restricted to a subset

fino a infinito

X random variable with possible values $\chi=\{0, 1, 2, \ldots\}$ and cdf $F(x)=\Pr(X \leq x)$

- want to restrict X to the finite range $0 \leq a \leq x \leq b < \infty$
- if $a > 0$, $\alpha = \Pr(X < a), \quad \beta = \Pr(X > b)$ che cosa sto tagliando fuori? alfa coda sinistra, beta destra.

$$\alpha = \Pr(X < a) = \Pr(X \leq a\text{-}1) = F(a\text{-}1)$$

$$\beta = \Pr(X > b) = 1\text{-}\Pr(X \leq b) = 1\text{-}F(b)$$

$$\Pr(a \leq X \leq b) = \Pr(X \leq b) - \Pr(X < a) = F(b) - F(a\text{-}1)$$

essentially, always true iff $F(b) \cong 1.0$ and $F(a\text{-}1) \cong 0.0$

la differenza tra variabile teorica e troncata potrebbe essere minima, se "non taglio nulla".

Specifying truncation points

- if a and b are specified

  Ho l'argomento, devo calcolare la cumulativa per quell'argomento.

  Left-tail, right-tail probabilities α and β obtained using cdf

  $\alpha = \Pr(X < a) = F(a\text{-}1)$ and $\beta = \Pr(X > b) = 1\text{-}F(b)$

  transformation is exact

- if α and β are specified

  Ho i valori della probabilità, devo trovare l'argomento però.

  idf can be used to obtain a and b

  $a = F^*(\alpha)$ and $b = F^*(1\text{-}\beta)$

  transformation is not exact because X is discrete

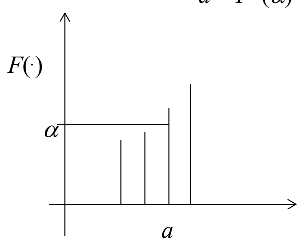
  $\Pr(X < a) \leq \alpha$ and $\Pr(X > b) < \beta$

$F(x-1) \leq u < F(x)$

Specifying truncation points

- if α and β are specified

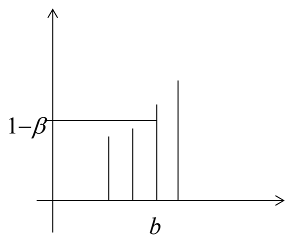
$a = F^*(\alpha)$



$F(a) > \alpha$

$\Pr(X < a) \leq \alpha$

$b = F^*(1\text{-}\beta)$



$F(b) > 1\text{-}\beta$

$\Pr(X \leq b) > 1\text{-}\beta$

$-\Pr(X \leq b) < \beta - 1$

$1\text{-}\Pr(X \leq b) < \beta$

$\Pr(X > b) < \beta$

io conosco 'alfa' (valore cumulativa) asse y.
Su asse x, incremento 'a', calcolo F(a) e il primo
a che mi eccede alfa è l'argomento cercato.

Effects of truncation

sometimes truncation is insignificant:
truncated and un-truncated random variables have (essentially) the same distribution

Truncation is useful for efficiency:

- When idf is complex, inversion requires cdf search
- cdf values are typically stored in an array
- Small range gives improved space/time efficiency

Truncation is useful for realism:

- Prevents arbitrarily large values possible from some variates

In some applications, truncation is significant

- Produces a new random variable
- Must be done correctly !

19

Se "taglio poco", il troncamento non porta effetti significativi.
Se "taglio molto", potrei avere una nuova variabile random!
Devo sempre ponderare queste scelte.