

Sistema binario

Sommario

- informatica
- rappresentare informazioni
- la differenza Analogico/Digitale
- i sistemi di numerazione posizionali
- il sistema binario

Informatica

Definizione

la scienza della rappresentazione e della elaborazione dell'informazione

- L'informazione è costituita dal connubio di:

dati rappresentazioni di entità di interesse

interpretazione regole per comprendere e manipolare i dati

Rappresentazione dell'informazione

Componenti

supporto fisico l'informazione deve essere scritta su un qualche supporto fisico che funga da veicolo (e.g. la carta, un disco magnetico)

linguaggio è un insieme di regole che permette di scrivere e leggere informazione, e di interpretarla (e.g. la lingua Italiana, i sistemi di numerazione posizionali)

Rappresentazione dell'informazione

Linguaggio

- Un linguaggio è tipicamente costituito da:

alfabeto un insieme di simboli appartenenti al linguaggio

regole un insieme di regole che permettono di combinare tali simboli in costrutti del linguaggio, e di interpretarli.

- Perché ogni simbolo contribuisca un minimo di informazione, un alfabeto deve avere almeno due simboli distinti

Rappresentazione dell'informazione

Sistemi automatici

- Per poter elaborare automaticamente informazione, è necessario darle una rappresentazione che sia gestibile da uno strumento automatico
- Esistono fonti di informazione estremamente diverse, e il loro trattamento automatico richiede una uniformazione nella modalità con cui vengono rappresentate
- Come vedremo, tutta l'informazione gestita dal calcolatore viene mantenuta in forma numerica, con il più semplice sistema di numerazione possibile, quello binario (con due soli simboli)
- Inoltre tutti i dati vengono memorizzati tramite numeri interi finiti (possibilmente con un'unità frazionaria)

Rappresentazione numerica

Rappresentazione di dati

- Dal punto di vista della rappresentazione, un dato può essere di vari tipi:

categorico rosso,verde,blu

ordinale orrendo,brutto,bello,fantastico

numerale discreto 10, 159, -10

numerale continuo $1e-13$, $\sqrt{2}$

- Tutti questi tipi possono essere rappresentati o approssimati tramite numeri interi (possibilmente con un'unità frazionaria)

Rappresentazione numerica

Dati categorici come interi

- Si usa il metodo del *dizionario*
- Ad ogni possibile valore viene associato un numero intero, con cui viene rappresentato
- Esempio: rosso \rightarrow 0, verde \rightarrow 1, blu \rightarrow 2
- L'unica operazione definita su questo tipo di dati è l'*uguaglianza*, e la codifica preserva tale operazione (rosso=rosso \rightarrow 0=0)

Rappresentazione numerica

Dati ordinali come interi

- Si usa il metodo dell'*enumerazione*
- I valori sono ordinati, ed ad ognuno viene associato un numero intero crescente
- Esempio: orrendo \rightarrow 0, brutto \rightarrow 1, bello \rightarrow 2 fantastico \rightarrow 3
- La codifica rispetta la relazione di ordine (orrendo $<$ bello \rightarrow 0 $<$ 2)

Rappresentazione numerica

Dati reali come combinazione di interi

- Un reale non può essere rappresentato che come reale, ma può essere *approssimato*
- Un numero razionale può approssimare un numero reale:

$$\sqrt{2} = 1.41421356237309504880168872420969807..$$

- In pratica si sceglie il numero di cifre della rappresentazione, e si approssima il reale con quelle cifre. E.g. con 10 cifre frazionarie:

$$\sqrt{2} \approx 1.4142135624$$

- Il numero risultante può essere scritto come la combinazione di un intero per una unità frazionaria

$$1.4142135624 = 14142135624 * 0.0000000001$$

Rappresentazione numerica discreta

Codica analogica contro codifica digitale

- Come è meglio codificare l'informazione per la sua elaborazione automatica mediante un calcolatore ?
- Grandezze naturali come l'intensità di un suono assumono valori *continui*.
 - Una codifica si dice *analogica* se mantiene un'analogia tra la struttura dell'entità di informazione e struttura della configurazione (possiamo parlare di codifica *continua*).
 - Una codifica si dice *digitale* se impone un numero di configurazioni distinte ammissibili e converte un'entità di informazione in una di queste configurazioni mediante una regola di codifica (possiamo parlare di codifica *discreta*).

Codica analogica contro codifica digitale

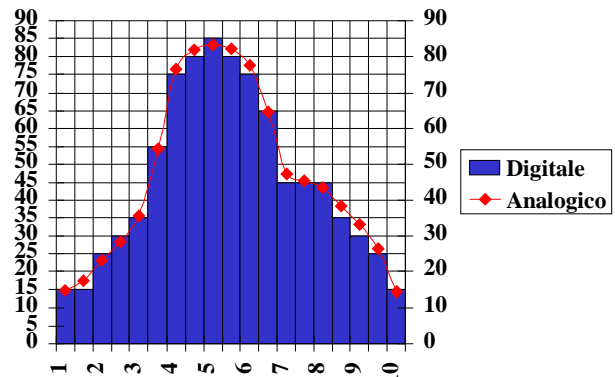
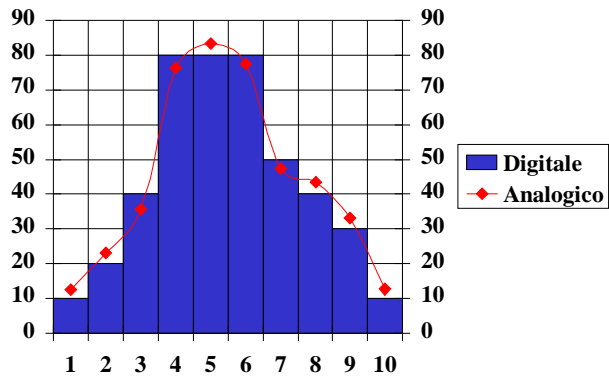
Vantaggi della codifica digitale

- Qualsiasi sistema fisico (incluso il calcolatore) è sottoposto all'influenza dell'ambiente circostante che ne perturba la configurazione introducendo *rumore*.
 - E' necessario rendere il sistema fisico che elabora l'informazione il più immune possibile al rumore.
 - La codifica analogica è fortemente sensibile al rumore, poichè tutte le configurazioni sono lecite e non si può distinguere la componente di informazione dal contributo dovuto al rumore.
 - Minore è il numero di configurazioni possibili maggiore è la possibilità di isolare l'informazione dal rumore.
 - E' questa la ragione principale del successo della codifica *binaria* nei calcolatori elettronici (la più estrema codifica discreta: due soli valori per ogni simbolo)

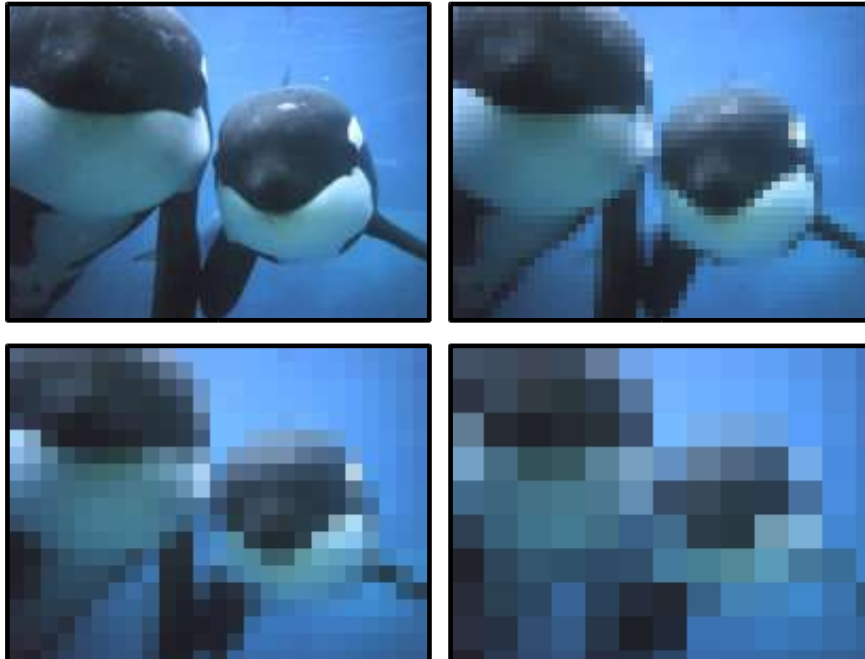
Codifica digitale di segnali analogici

Campionamento

- Permette di approssimare in maniera arbitrariamente accurata informazione continua (analogica).
- Consiste nel:
 1. misurare il segnale ad intervalli regolari (di tempo se evolve nel tempo, di spazio se immagine, entrambi se video)
 2. approssimare ogni misura continua con un valore discreto



Esempio



Rappresentazione dei numeri

Premessa

- Cosa intendiamo quando scriviamo (ad es.) ?

1945732

unmilionenovecentoquarantacinquemilasettecentotrentadue

- Questo si ottiene come:
due + trenta + settecento + cinquemila + \dots + unmilione
- Ogni cifra viene moltiplicata per un peso. Da destra a sinistra i pesi sono 1, 10, 100, 1000, \dots , 1000000, ossia $10^0, 10^1, 10^2, 10^3, \dots, 10^6$
- Il totale si ottiene sommando il valore di ciascuna cifra moltiplicato per il peso corrispondente alla sua posizione.

Sistema di numerazione posizionale

sistema *posizionale*

- Si parla di sistema *posizionale* perché il valore di una cifra dipende dalla sua posizione
- Le posizioni si contano da destra a sinistra a partire da 0
- Il valore della cifra viene moltiplicato per la base (10) elevata alla posizione
- Esempio: 1456

cifra	posizione	valore
6	0	$6 * 10^0 = 6$
5	1	$5 * 10^1 = 50$
4	2	$4 * 10^2 = 400$
1	3	$1 * 10^3 = 1000$

- Tale sistema di numerazione facilita notevolmente le operazioni aritmetiche rispetto ai sistemi non posizionali (e.g. quello dei Romani)

Sistemi di numerazione posizionali: definizione

Ingredienti

- Un numero naturale b detto *base*. (e.g. 10)
- Un insieme ordinato di $b - 1$ simboli distinti detti *cifre* (e.g. 0,1,2,3,4,5,6,7,8,9)
- Un *codice di interpretazione* per determinare il numero rappresentato da una stringa di cifre.
- Un insieme di procedure (algoritmi) per le quattro operazioni aritmetiche $+, -, \times, /$

Sistemi di numerazione posizionali: definizione

Codice di interpretazione

- Un numero intero rappresentato in base b con n cifre è una stringa di cifre:

$$(c_{n-1} \cdots c_0)_b$$

- Ad ogni posizione nella stringa è associato un peso. Da destra a sinistra i pesi sono:

$$b^0, \dots, b^{n-1}$$

- Ogni cifra rappresenta il numero di volte in cui deve essere considerato il peso corrispondente alla posizione in cui si trova la cifra stessa

Forma polinomiale

- Possiamo dunque definire la seguente relazione detta *forma polinomiale*

$$(c_{n-1} \cdots c_0)_b = c_0 \times b^0 + \cdots + c_{n-1} \times b^{n-1}$$

- analogamente possiamo scrivere numeri frazionari:

$$(.c_{-1} \cdots c_{-m})_b = c_{-1} \times b^{-1} + \cdots + c_{-m} \times b^{-m}$$

- o numeri con parte intera e parte frazionaria:

$$(c_{n-1} \cdots c_0.c_{-1} \cdots c_{-m})_b = c_{n-1} \times b^{n-1} + \cdots + c_0 \times b^0 + c_{-1} \times b^{-1} + \cdots + c_{-m} \times b^{-m}$$

Nota

Il sistema di numerazione in base 10, o *sistema decimale*, è il sistema comunemente usato, ed i numeri in base 10 sono rappresentati di norma senza l'indicazione della base.

Sistemi di numerazione con altre basi

Sistema binario (base 2)

- E' il sistema di numerazione con la base più piccola possibile
- In questo caso le cifre sono $\{0,1\}$
- Si parla di cifra binaria (*binary digit* o *bit*)
- Il bit è l'unità minima di informazione.
- Conversione binario \rightarrow decimale tramite la sua forma polinomiale:

$$\begin{aligned}(1011.01)_2 &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} \\ &= (11.25)_{10}\end{aligned}$$

Conversione decimale → binario di numeri interi

- Problema formulabile come segue: dato l'intero decimale N , determinare la stringa di bit $c_{n-1} \cdots c_2 c_1 c_0$ tale che

$$N = c_0 \times 2^0 + c_1 \times 2^1 + c_2 \times 2^2 + \cdots + c_{n-1} \times 2^{n-1}$$

- Si applica il metodo delle *divisioni successive*.
- Dividiamo entrambi i membri dell'uguaglianza per 2:

$$N/2 = c_0 \times 2^{-1} + c_1 \times 2^0 + c_2 \times 2^1 + \cdots + c_{n-1} \times 2^{n-2} = R \times 2^{-1} + Q$$

dove $R = N \bmod 2$ e $Q = \lfloor N/2 \rfloor$ sono rispettivamente resto e quoziente della divisione.

- Si ricava così che:

$$\begin{aligned} c_0 &= R = N \bmod 2 \\ Q &= \lfloor N/2 \rfloor = c_1 \times 2^0 + c_2 \times 2^1 + \cdots + c_{n-1} \times 2^{n-2} \end{aligned}$$

Conversione decimale → binario di numeri interi

- Se $Q \neq 0$ possiamo ripetere la procedura:

$$\begin{aligned} Q &= c_1 \times 2^0 + c_2 \times 2^1 + \cdots + c_{n-1} \times 2^{n-2} \\ Q/2 &= c_1 \times 2^{-1} + c_2 \times 2^0 + \cdots + c_{n-1} \times 2^{n-3} \\ &= R' \times 2^{-1} + Q' \end{aligned}$$

- da cui

$$\begin{aligned} c_1 &= R' = Q \bmod 2 \\ Q' &= \lfloor Q/2 \rfloor = c_2 \times 2^0 + \cdots + c_{n-1} \times 2^{n-3} \end{aligned}$$

- Iterando finché si ottiene una divisione con quoziente nullo.

Esempio

- Convertire in binario il numero decimale $(61)_{10}$

61	$\bmod 2 = 1 = c_0$	(least significant bit),	$\lfloor 61/2 \rfloor =$	30
30	$\bmod 2 = 0 = c_1$		$\lfloor 30/2 \rfloor =$	15
15	$\bmod 2 = 1 = c_2$		$\lfloor 15/2 \rfloor =$	7
7	$\bmod 2 = 1 = c_3$		$\lfloor 7/2 \rfloor =$	3
3	$\bmod 2 = 1 = c_4$		$\lfloor 3/2 \rfloor =$	1
1	$\bmod 2 = 1 = c_5$	(most significant bit)	$\lfloor 1/2 \rfloor =$	0

- quindi $(61)_{10} = (111101)_2$

Conversione decimale → binario di numeri frazionari

- Problema formulabile come segue: dato il numero frazionario decimale F , determinare la stringa di bit $c_{-1}c_{-2}c_{-3} \dots c_{-m}$ tale che

$$F = c_{-1} \times 2^{-1} + c_{-2} \times 2^{-2} + c_{-3} \times 2^{-3} + \dots + c_{-m} \times 2^{-m}$$

- Si applica il metodo delle *moltiplicazioni successive*.
- Moltiplichiamo entrambi i membri dell'uguaglianza per 2:

$$F \times 2 = c_{-1} + c_{-2} \times 2^{-1} + c_{-3} \times 2^{-2} + \dots + c_{-m} \times 2^{-m+1} = N + F'$$

dove $N = c_{-1}$ è la parte intera del risultato ed $F' = c_{-2} \times 2^{-1} + c_{-3} \times 2^{-2} + \dots + c_{-m} \times 2^{-m+1}$ è la parte frazionaria.

Conversione decimale → binario di numeri frazionari

- Si itera il procedimento sulla parte frazionaria finchè si verifica una delle due condizioni:
 1. Si raggiunge il numero massimo di cifre binarie con cui si intende rappresentare il numero frazionario (si ottiene un' *approssimazione per difetto* del numero decimale).
 2. Si ottiene come risultato di una moltiplicazione un numero con parte frazionaria nulla.

Operazioni aritmetiche binarie

- Funzionano in maniera analoga a quelle decimali.
- Tabella per l'addizione binaria:

$$\begin{array}{l} 0 + 0 = 0 \\ 0 + 1 = 1 \\ 1 + 0 = 1 \\ 1 + 1 = 0 \quad \text{con riporto di } 1 \end{array}$$

- Esempio

$$\begin{array}{r} \\ \\ \\ \hline 1 \end{array}$$

Operazioni aritmetiche binarie

- Tabella per la sottrazione binaria:

$$\begin{array}{l} 0 - 0 = 0 \\ 0 - 1 = 1 \quad \text{con prestito di } 1 \\ 1 - 0 = 1 \\ 1 - 1 = 0 \end{array}$$

- Esempio

$$\begin{array}{r} \\ \\ \\ \hline 0 \end{array}$$

Sistemi di numerazione in altre basi

Sistema ottale (base 8)

- Cifre usate: $\{0,1,2,3,4,5,6,7\}$
- Conversione ottale \rightarrow decimale tramite la forma polinomiale
- Conversione decimale \rightarrow ottale tramite divisioni (o moltiplicazioni per la parte frazionaria) successive per la base (8).

Esempi

- Conversione ottale \rightarrow decimale di $(754)_8$

$$(754)_8 = 4 \times 8^0 + 5 \times 8^1 + 7 \times 8^2 = (492)_{10}$$

- Conversione decimale \rightarrow ottale di $(678)_{10}$

678	mod 8 = 6 = c_0	$\lfloor 678/8 \rfloor$	=	84
84	mod 8 = 4 = c_1	$\lfloor 84/8 \rfloor$	=	10
10	mod 8 = 2 = c_2	$\lfloor 10/8 \rfloor$	=	1
1	mod 8 = 1 = c_3	$\lfloor 1/8 \rfloor$	=	0

quindi $(678)_{10} = (1246)_8$

Conversione tra binario ed ottale

- Si noti che ciascuna cifra ottale può essere rappresentata con tre bits:

cifra ottale	numero binario
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Conversione tra binario ed ottale

Conversione binario \rightarrow ottale

1. Si raggruppano i bit a gruppi di tre da destra verso sinistra per la parte intera, da sinistra verso destra per la parte frazionaria.
2. Si aggiungono se necessario bit 0 a sinistra (per la parte intera) ed a destra (per la parte frazionaria) del numero.
3. Si sostituisce ogni gruppo di tre cifre binarie con la corrispondente cifra ottale.

$$\text{e.g. } (1011.1001)_2 \rightarrow (001\ 011 . 100\ 100)_2 \rightarrow (13.44)_8$$

Conversione ottale \rightarrow binario

1. Si sostituisce ogni cifra ottale con il corrispondente gruppo di tre cifre binarie

$$\text{e.g. } (23.14)_8 \rightarrow (010\ 011 . 001\ 100)_2 \rightarrow (10011.0011)_2$$

Sistemi di numerazione in altre basi

Sistema esadecimale (base 16)

- Cifre usate: $\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$
- Dove vale la seguente conversione:

cifra esadecimale	numero decimale
A	10
B	11
C	12
D	13
E	14
F	15

- La conversione tra esadecimali e decimali è analoga a quelle già viste (forma polinomiale per passare a decimale, divisioni o moltiplicazioni per la base 16 per passare a esadecimale).

Esempi

- Conversione esadecimale \rightarrow decimale di $(A54)_{16}$

$$(A54)_{16} = 4 \times 16^0 + 5 \times 16^1 + 10 \times 16^2 = (2644)_{10}$$

- Conversione decimale \rightarrow esadecimale di $(678)_{10}$

$$\begin{array}{rcl} 678 \text{ mod } 16 = 6 = c_0 & [678/16] & = 42 \\ 42 \text{ mod } 16 = 10 = A = c_1 & [42/16] & = 2 \\ 2 \text{ mod } 16 = 2 = c_2 & [2/16] & = 0 \end{array}$$

quindi $(678)_{10} = (2A6)_{16}$

Conversione tra binario ed esadecimale

- Si noti che ciascuna cifra esadecimale può essere rappresentata con quattro bit:

0000 \rightarrow 0	0001 \rightarrow 1	0010 \rightarrow 2	0011 \rightarrow 3
0100 \rightarrow 4	0101 \rightarrow 5	0110 \rightarrow 6	0111 \rightarrow 7
1000 \rightarrow 8	1001 \rightarrow 9	1010 \rightarrow A	1011 \rightarrow B
1100 \rightarrow C	1101 \rightarrow D	1110 \rightarrow E	1111 \rightarrow F

- Le conversioni sono analoghe a quelle viste per la numerazione ottale:

$$- (101111.01)_2 \rightarrow (0010\ 1111\ .\ 0100)_2 \rightarrow (2F.4)_{16}$$

$$- (A3.E)_{16} \rightarrow (1010\ 0011\ .\ 1110)_2 \rightarrow (10100011.111)_2$$

Utilità dei sistemi ottale ed esadecimale

- Servono a rappresentare in maniera leggibile e concisa stringhe di bit.
- La numerazione ottale è stata introdotta in informatica quando i mainframe più diffusi usavano parole di 24 o 36 bit (divisibili per 3).
 - E' ancora diffusa per rappresentare i permessi sui file nei sistemi *Unix*: lettura (4), scrittura (2), esecuzione (1). Si sommano i valori dei permessi che si vogliono garantire (e.g. 6 = lettura e scrittura)
- Con la diffusione dei computer a 16, 32 e 64 bit (divisibili per 4) si è imposta la numerazione esadecimale.
 - Ad esempio si usa in HTML per rappresentare i colori a 24-bit nel formato RGB (#RRGGBB con RR valore della componente rossa, GG della verde e BB della blu, e.g. #FFFF00 = giallo)