Appunti di Elaborazione di Segnali Multimediali
a.a. 2012/2013

Compattazione dati

G.Poggi, L.Verdoliva

La compattazione (detta anche *codifica entropica*) è una forma di codifica che permette di rappresentare i messaggi prodotti dalla sorgente in modo più efficiente, ma senza alcuna perdita di informazione, in modo da poter risalire senza errori dai dati codificati a quelli originari. Tipicamente viene applicata al flusso di dati già compressi allo scopo di ridurre ulteriormente i simboli da trasmettere senza introdurre perdite. Le strategie di codifica alla base della compattazione dati richiedono strumenti teorici che vengono sviluppati nell'ambito della Teoria dell'Informazione e che fanno uso del concetto di entropia.

1 Introduzione

Il concetto fondamentale che è alla base del progetto di una buona tecnica di codifica è la regolarità statistica della sorgente. Per poter rappresentare efficientemente i dati bisogna portare alla luce queste regolarità e quindi sfruttarle; a volte questo si può fare in modo automatico (algoritmico), altre volte, laddove gli algoritmi non possono arrivare per ragioni di complessità, l'analisi umana dei dati è in grado di individuare meglio queste regolarità. In questo senso la compressione è a metà fra scienza ed artigianato, è quindi importante lo studio dei concetti base e delle diverse tecniche di codifica in modo da fornire gli strumenti per affrontare efficacemente i diversi problemi specifici che si possono incontrare. Individuare le regolarità significa adottare un modello dei dati che possa spiegarne in maniera sintetica il comportamento. Il modello sintetizza la sorgente nelle sue caratteristiche principali, ingloba la conoscenza (a priori o posteriori che sia) disponibile su di essa. Una volta noto il modello basta descrivere gli scostamenti dei dati dal modello. Tale modello può avere a volte un'origine fisica oppure si può ricavare esclusivamente dall'osservazione dei dati senza conoscere la loro semantica.

Uno dei primi a sfruttare la ridondanza (e quindi la struttura) della lingua inglese per effettuare codifica, è stato Samuel Morse, che nella metà del diciannovesimo secolo, ha sviluppato un codice per ridurre il tempo medio necessario a trasmettere un messaggio attraverso un telegrafo. Morse notò che alcune lettere dell'alfabeto si presentano più spesso di altre, e quindi, assegnò sequenze di punti e linee più brevi alle lettere che si presentavano più frequentemente, come $e\,(\cdot)$ ed $a\,(\cdot-)$, e più lunghe a quelle che si presentavano meno frequentemente, come $q\,(--\cdot-)$ e $j\,(\cdot---)$. Questa stessa semplice idea è alla base di algoritmi attualmente usati per la compressione di file di dati (codifica di Huffmann). Anziché far riferimento alle lettere si può pensare all'occorrenza delle parole, così come accade nel codice Braille, sviluppato sempre nello stesso periodo e che permette una riduzione del venti per cento dello spazio occupato. La

diversa frequenza di occorrenza delle lettere dell'alfabeto, o delle parole del vocabolario, rappresenta una struttura della sorgente d'informazione, la cui conoscenza rende possibile effettuare la compressione. In altri termini, nella sorgente esiste una certa regolarità, o *ridondanza statistica*, che permette di ottenere una sua rappresentazione meno dispendiosa.

Se si vogliono confrontare le prestazioni di tecniche differenti di compattazione dati si usa il *tasso di codifica* R, pari al numero medio di bit necessario a codificare un simbolo di sorgente. Una misura ad esso correlata è il *rapporto di compressione*: detto n_s il numero medio di cifre binarie speso per rappresentarte i simboli di uscita della sorgente, e n_c il numero medio di bit necessario per rappresentare gli stessi simboli dopo un'operazione di codifica, il rapporto di compressione si può definire come

$$CR = \frac{n_s}{n_c} \tag{1}$$

Questa quantità dà un'indicazione dell'efficienza di codifica e permette di confrontare immediatamente le prestazioni di metodi differenti. Per sorgenti di interesse comune (file dati, segnali vocali, immagini) il rapporto di compressione generalmente assume valori non maggiori di 2.

2 Entropia

Abbiamo detto che la compattazione dati si prefigge lo scopo di ridurre la quantità di dati necessaria a rappresentare una sorgente senza però rimuovere l'informazione presente. A tal fine, è necessario definire e misurare il contenuto informativo trasportato da un segnale. Shannon, per primo, ha attribuito un senso rigoroso e operativo al concetto astratto di informazione, introducendo l'autoinformazione $I(\cdot)$ associata ad un evento. La nozione intuitiva che un evento è tanto più informativo quanto più improbabile viene formalizzata dall'espressione

$$I(A) = \log \frac{1}{P(A)} = -\log P(A))$$

con $P(A)$ probabilità che si verifichi l'evento A. Anzitutto è chiaro che l'informazione che ci porta un evento (o se vogliamo la notizia del verificarsi dell'evento) è tanto maggiore quanto più improbabile è l'evento, e questo giustifica il fatto che $I(A)$ dipenda dal reciproco di $P(A)$. Per giustificare intuitivamente la presenza del logaritmo possiamo fare due osservazioni; anzitutto, il verificarsi di un evento certo (domani sorge il sole) non porta informazione, ed infatti $\log 1 = 0$, mentre il verificarsi di un evento a probabilità 0 (o quasi: domani non sorge il sole) porta informazione infinita. Inoltre, se un evento è il prodotto di due eventi indipendenti, l'autoinformazione associata risulta essere la somma delle due autoinformazioni

$$P(AB) = P(A)P(B) \Rightarrow I(AB) = I(A) + I(B)$$

Si noti che la base del logaritmo è arbitraria e scelte diverse produrranno semplicemente unità di misura diverse: bits ($\log_2$), nats ($\log_e$), hartleys ($\log_{10}$).

Ora concentriamo l'attenzione non già su un singolo evento, ma su un esperimento caratterizzato da diversi eventi ognuno con la sua probabilità. Per conservare il parallelo giornalistico, le due notizie (domani il sole sorge) e (domani il sole non sorge) sono i due possibili risultati di una inchiesta. E' evidente che i due eventi portano quantità d'informazione radicalmente diverse, ma qual è l'informazione media associata all'esperimento o, in altri termini, conviene

spendere un cronista per indagare su questo problema? La risposta intuitiva è no e questa è anche la risposta che viene dalla teoria dell'informazione. A tal fine definiamo l'informazione media associata ad un esperimento costituito da $A_1, A_2, \cdots, A_n$ eventi:

$$E[I(A_i)] = \sum_i P(A_i) I(A_i) = -\sum_i p_i \log p_i = H(\mathbf{p})$$

dove per semplicità abbiamo posto $P(A_i) \stackrel{\triangle}{=} p_i$ e $\mathbf{p} = [p_1, p_2, \cdots, p_n]$. $H(\mathbf{p})$ è detta *entropia* associata all'esperimento e dipende unicamente dal modello probabilistico della sorgente. Riguardo l'esempio precedente costituito da soli due eventi, detta p la probabilità che il sole non sorga, abbiamo

$$H(\mathbf{p}) = -(1-p)\log(1-p) - p\log p = 1\log 1 + 0\log 0 = 0$$

avendo assunto per continuità che $0 \log 0 = \lim_x x \log x = 0$. L'entropia dunque misura l'informazione media portata dall'osservazione dell'esperimento o anche, che è lo stesso, l'incertezza a priori sull'esito dell'osservazione.

2.1 Entropia del I ordine

A partire dai concetti appena esposti possiamo caratterizzare il contenuto informativo dei simboli emessi da una sorgente. Sia $\mathcal{S}$ la sorgente discreta di informazione e s_i il generico simobolo emesso scelto nell'alfabeto $\{\xi_1, \xi_2, \ldots, \xi_M\}$, cui associamo una variabile aleatoria $X_i : X_i(\xi_k) = x_k$. Si definisce allora entropia dell'i-esimo simbolo di sorgente la quantità

$$H(X_i) = \sum P(X_i = x_k) \log \frac{1}{P(X_i = x_k)} \tag{2}$$

cioè l'autoinformazione media associata al simbolo, detta anche *entropia del I ordine*. Nell'ipotesi di sorgente stazionaria possiamo liberarci dalla dipendenza da i e, introdotta $p_k \stackrel{\triangle}{=} \Pr(X = x_k)$, scrivere più semplicemente

$$H(X) = \sum p_k \log \frac{1}{p_k} \tag{3}$$

L'entropia rappresenta il numero di bit di informazione mediamente convogliati da un simbolo e misura perciò il contenuto di informazione di una sorgente. Se i simboli sono equiprobabili, l'entropia assume il suo valore massimo, pari a $\log_2 M$: si ha allora il massimo grado di incertezza relativa all'emissione del simbolo, ma anche il massimo contenuto informativo trasportato dal simbolo stesso. Se invece è presente un unico simbolo con probabilità unitaria, non si ha alcuna incertezza sulla sua trasmissione (che d'altra parte non fornisce alcuna informazione) e l'entropia è nulla. Più in generale risulta:

$$0 \leq H(X) \leq \log_2 M$$

2.1.1 Esempio

Si consideri la seguente sequenza:

1 2 3 2 3 4 5 4 5 6 7 8 9 8 9 10

Assumendo che la frequenza di occorrenza di ogni numero si possa stimare tramite il numero di volte che appare nella sequenza, possiamo valutare la probabilità di occorrenza di ogni simbolo come:

$$P(1) = P(6) = P(7) = P(10) = \frac{1}{16}$$

$$P(2) = P(3) = P(4) = P(5) = P(8) = P(9) = \frac{2}{16}$$

L'entropia del I ordine è:

$$H = -\sum_{i=1}^{10} P(i) \log_2 P(i) = 3.25 \text{ bit}$$

Si noti come questo valore sia inferiore a $\log_2 16 = 4$, che è il numero di bit necessario a codificare i valori usando un codice a lunghezza fissa.

2.2 Entropia congiunta

Data una coppia di variabili aleatorie, X_1 e X_2, si può valutare l'*entropia congiunta* (o del *II ordine*) definita come:

$$\begin{aligned} H(X_1, X_2) &\triangleq \sum_i \sum_j P(X_1 = x_i, X_2 = x_j) \log \frac{1}{P(X_1 = x_i, X_2 = x_j)} = \\ &= -\sum_i \sum_j p_{ij} \log p_{ij} \end{aligned} \tag{4}$$

dove abbiamo definito $P(X_1 = x_i, X_2 = x_j) \triangleq p_{ij}$. Rielaborando la (4) si ha:

$$\begin{aligned} H(X_1, X_2) &= -\sum_i \sum_j p_{ij} \log p_{ij} = -\sum_i \sum_j p_{ij} \log \left[p_j \, p_{i|j}\right] = \\ &= -\sum_i \sum_j p_{ij} \log p_j - \sum_i \sum_j p_{ij} \log p_{i|j} = \\ &= -\sum_j \log p_j \sum_i p_{ij} + H(X_1|X_2) \end{aligned}$$

dove abbiamo definito entropia condizionale $H(X_1|X_2)$ la seguente quantità:

$$\begin{aligned} H(X_1|X_2) &= \sum_i \sum_j p_{ij} \log p_{i|j} = \\ &= \sum_i \sum_j p_j p_{i|j} \log p_{i|j} = \sum_j P(X_2 = x_j) H(X_1|X_2 = x_j) \end{aligned}$$

In definitiva, si ottiene

$$\begin{aligned} H(X_1, X_2) &= -\sum_i p_{i|j} \sum_j p_j \log p_j + H(X_1|X_2) = \\ &= H(X_2) + H(X_1|X_2) \end{aligned}$$

dato che risulta $\sum_i p_{i|j} = 1$. Abbiamo così trovato l'importante relazione

$$H(X_1, X_2) = H(X_2) + H(X_1|X_2)$$

Essa si interpreta dicendo che l'incertezza sull'esperimento che coinvolge congiuntamente le variabili aleatorie X_1 e X_2 è uguale all'incertezza su X_2 più quella su X_1 avendo già osservato X_2; oppure dicendo che $H(X_1|X_2)$ è la quantità di incertezza restante su X_1 dopo avere osservato X_2. Se le variabili X_1 e X_2 sono indipendenti allora il condizionamento non opera e l'entropia congiunta risulta essere pari alla somma delle due entropie marginali.

2.2.1 Esempio

Consideriamo le variabili aleatorie X ed Y con $X = \{1, 2, 3, 4\}$, la cui matrice delle probabilità congiunte $P(X = x_i, Y = y_i)$ con x_i ed y_i è riportata in tabella 1. Risulta

$$H(X|Y = 3) = -\sum_{i=1}^{4} P(X = i|Y = 3) \log P(X = i|Y = 3) = -\frac{1}{4} \sum_{i=1}^{4} \log \frac{1}{4} = 2$$

Infatti $P(Y = 3) = \sum_{i=1}^{4} P(Y = 3|X = i)P(X = i) = \sum_{i=1}^{4} P(Y = 3, X = i) = 4 \cdot 1/16 = \frac{1}{4}$, ed inoltre

$$P(X = i|Y = 3) = \frac{P(X = i, Y = 3)}{P(Y = 3)} = \frac{1}{16} \cdot 4 = \frac{1}{4} \quad i \in \{1, 2, 3, 4\}$$

Proviamo ora a calcolare $H(X|Y = 4)$. Abbiamo $P(Y = 4) = \sum_{i=1}^{4} P(Y = 4, X = i) = \frac{1}{4}$, e quindi

$$P(X = i|Y = 4) = \frac{P(X = i, Y = 4)}{P(Y = 4)} = \begin{cases} \frac{1}{4} \cdot 4 = 1 & \text{se } i = 1 \\ 0 & \text{se } i \in \{2, 3, 4\} \end{cases}$$

Questo ci basta per concludere che è $H(X|Y = 4) = 0$ in quanto se $Y = 4$ risulta certamente $X = 1$: dato $Y = 4$ non c'è incertezza sulla variabile aleatoria X.

Vogliamo ora far vedere come *non* valga in generale la relazione $H(X, Y) = H(X) + H(Y)$. Infatti, come si può verificare, risulta $0 \leq H(X, Y) = \frac{27}{8} \leq \log 16$ bit (si otterrebbe $H(X, Y) = \log 16$ solo se i 16 eventi elementari del nostro esperimento fossero equiprobabili). Abbiamo

$Y \backslash X$	1	2	3	4
1	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{32}$
2	$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{32}$	$\frac{1}{32}$
3	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$
4	$\frac{1}{4}$	0	0	0

Tabella 1: Matrice delle probabilità congiunte

inoltre

$$P(X=1) = \sum_{i=1}^{4} \mathbf{P}(X=1, Y=i) = \frac{1}{8} + \frac{1}{16} + \frac{1}{16} + \frac{1}{4} = \frac{1}{2}$$
$$P(X=2) = \frac{1}{16} + \frac{1}{8} + \frac{1}{16} + 0 = \frac{1}{4}$$
$$P(X=3) = \frac{1}{32} + \frac{1}{32} + \frac{1}{16} + 0 = \frac{1}{8}$$
$$P(X=4) = \frac{1}{32} + \frac{1}{16} + \frac{1}{16} + 0 = \frac{1}{8}$$

da cui $H(X) = 1.75 = \frac{14}{8}$. Analogamente, essendo

$$P(Y=1) = \sum_{i=1}^{4} P(Y=1, X=i) = \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{32} = \frac{1}{4}$$
$$P(Y=2) = \frac{1}{16} + \frac{1}{8} + \frac{1}{32} + \frac{1}{32} = \frac{1}{4}$$
$$P(Y=3) = \frac{1}{16} + \frac{1}{16} + \frac{1}{16} + \frac{1}{16} = \frac{1}{4}$$
$$P(Y=4) = \frac{1}{4} + 0 + 0 + 0 = \frac{1}{4}$$

Y è uniformemente distribuita e perciò si ha $H(Y) = \log 4 = 2$. In definitiva risulta $H(X,Y) \neq H(X) + H(Y)$ e ciò perché le due variabili aleatorie *non* sono indipendenti. Dalla relazione $H(X,Y) = H(X) + H(Y|X)$ si ottiene

$$H(Y|X) = H(X,Y) - H(X) = \left(\frac{27}{8} - \frac{14}{8}\right) = \frac{13}{8}$$

Si comprende dunque che, essendo $H(Y) - H(Y|X) = (2 - \frac{13}{8}) = \frac{3}{8}$, l'osservazione di X porta $\frac{3}{8}$ di bit di informazione su Y. Analogamente, dalla relazione $H(X,Y) = H(Y) + H(X|Y)$ si ottiene $H(X|Y) = H(X,Y) - H(Y) = \frac{11}{8}$, quindi, essendo $H(X) = \frac{14}{8}$, anche l'osservazione di Y porta $\frac{3}{8}$ di informazione su X.

Conoscere una variabile aleatoria può diminuire o al più lasciare inalterata l'informazione media (incertezza) associata ad un'altra variabile aleatoria. É importante osservare che *ciò vale in media* e *non* puntualmente, cioè vale per $H(X|Y)$ (abbiamo dimostrato che è $H(X|Y) \leq H(X)$) e non per $H(X|Y = y_i)$; infatti nell'esempio precedente risulta $H(X|Y=3) = 2$ che è maggiore di $H(X) = 1.75$.

2.3 Tasso entropico

Dal concetto di entropia si passa poi a quello di tasso entropico, ben definito per sorgenti stazionarie, mediante le distribuzioni di probabilità su n simboli con n comunque elevato, cioè

$$H(\mathcal{S}) = \lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, \ldots, X_n) = \lim_{n\to\infty} H(X_n|X_1, X_2, \ldots, X_{n-1}) \tag{5}$$

con $\{X_1, X_2, \ldots, X_n\}$ sequenza di lunghezza n della sorgente. Mentre $H(X)$ rappresenta il contenuto informativo di un singolo simbolo di sorgente, o entropia del I ordine, il tasso entropico è il contenuto informativo medio di una sequenza di simboli lunga a piacere, e si dimostra che $H(S) \leq H(X)$ con uguaglianza se e solo se i simboli sono indipendenti.

Purtroppo il calcolo del tasso entropico richiede la conoscenza di distribuzioni di probabilità di ordine comunque elevato. Dato che è difficile conoscere il modello probabilistico della sorgente, non è possibile valutarne esattamente il contenuto informativo, ma se ne può fornire solo una stima. La stima dell'entropia è più o meno accurata a seconda di quanto le ipotesi fatte sulla struttura della sequenza emessa dalla sorgente si discostano dalla realtà. Quando non si ha alcuna conoscenza sulla sorgente si adotta un modello molto semplice (ignorance model), assumendo che i simboli emessi costituiscano una sequenza di variabili aleatorie indipendenti ed identicamente distribuite. In tal caso il calcolo dell'entropia si riduce alla stima delle sole probabilità del primo ordine. Se però l'ipotesi di indipendenza non è verificata, tale stima non è accurata perché il tasso entropico effettivo è significativamente minore. Purtroppo, quando si devono calcolare entropie di ordine superiore al primo, le stime diventano subito poco affidabili; per tener conto allora della struttura presente nei dati, è necessario costruirsi un modello matematico che descriva le dipendenze tra campioni vicini e poi codificare i dati rispetto a quel modello.

Uno dei modi più popolari per rappresentare la dipendenza è usare dei modelli di Markov, in particolare modelli del primo ordine, in cui la conoscenza dell'ultimo simbolo equivale alla conoscenza di tutta la storia passata del processo:

$$P(X_n|X_{n-1}) = P(X_n|X_{n-1}, X_{n-2}, X_{n-3}, \ldots)$$

Si suppone cioè che ci sia dipendenza solo tra simboli adiacenti, per cui il tasso entropico si valuta molto semplicemente come:

$$H(S) = H(X_2|X_1)$$

I modelli di Markov risultano particolarmente utili nella compressione di testi, dove la probabilità di una lettera è fortemente influenzata da quella precedente. L'uso dei modelli di Markov per l'inglese scritto appare nel lavoro originario di Shannon del 1948. Shannon ha utilizzato un modello del secondo ordine per il testo inglese (definito su un alfabeto di 26 lettere e uno spazio) e ha trovato un valore di entropia di 3.1 bit/lettera (più bassa dell'entropia del primo ordine pari a 4.03). Attraverso un modello più complicato in cui i simboli in uscita erano parole, anziché lettere, l'entropia si è ridotta a 2.4; tramite predizioni effettuate da persone, e non modelli statistici, è stato possibile stimare un limite inferiore e superiore all'entropia di 0.6 e 1.3 bit/lettera, rispettivamente.

Un altro modello che descrive le dipendenze tra i campioni è basato sulle conoscenze disponibili a priori sul processo di generazione dei dati. Infatti questa informazione può essere usata per formulare delle predizioni sui valori assunti dal segnale. In questo modo, si può codificare la differenza tra i valori reali e quelli predetti (codifica predittiva), che tipicamente possiede un'entropia del primo ordine più bassa e permette quindi di ottenere facilmente un rapporto di compressione più alto. Poiché è difficile possedere conoscenze a priori, è possibile sviluppare un modello basato sulle osservazioni empiriche delle statistiche dei dati.

2.3.1 Esempio

Supponiamo di avere un'immagine binaria (per esempio un fax) e che la probabilità che il pixel sia bianco $P(B) = 30/31$ risulti maggiore della probabilità che il pixel sia nero $P(N) = 1/31$. Se avessiamo a disposizione solo queste probabilità e ipotizzassimo l'indipendenza tra i pixel l'entropia dell'immagine la stimeremmo come l'entropia del I ordine pari a $H(30/31, 1/31) = 0.206$ bit.

In realtà, è ragionevole ritenere che ci sia dipendenza tra i pixel. Nell'ipotesi di considerare solo quelle tra pixel vicini (modello di Markov del I ordine) e trascurare le altre, si può ritenere che una buona stima dell'entropia della sorgente sia:

$$H(S) = H(X_2|X_1) = P(X_1 = B)H(X_2|X_1 = B) + P(X_2 = N)H(X_2|X_1 = N) =$$

Per calcolare le due entropie condizionali è necessario conoscere le probabilità condizionali, che supponiamo essere le seguenti: $P(B|B) = 0.99$, $P(N|B) = 0.01$, $P(N|N) = 0.7$ e $P(B|N) = 0.3$. Quindi l'entropia risulta:

$$H(S) = \frac{30}{31}H(0.99, 0.01) + \frac{1}{31}H(0.7, 0.3) = 0.107$$

pari circa alla metà di quella calcolata usando l'entropia del I ordine.

3 La compattazione dati

E' il processo di codifica che permette di rappresentare in modo più efficiente, ma senza alcuna perdita di informazione, i messaggi prodotti dalla sorgente in modo da poter risalire senza errori dai dati codificati a quelli originari. E' fondamentale in questo processo definire il *codice*, cioè un vocabolario che traduca ogni simbolo emesso dalla sorgente in una sequenza (o *stringa*) di simboli (tipicamente binari) del nuovo alfabeto. Detta X una variabile aleatoria discreta a valori nell'insieme $\mathcal{X} = \{x_1, x_2, \ldots, x_M\}$. Indicato con $\mathcal{D} = \{0, 1\}$ l'insieme ti tutte le stringhe finite di un alfabeto binario, prende il nome di codice per X ogni funzione

$$C\colon x_i \in \mathcal{X} \mapsto C(x_i) \in \mathcal{D}$$

Consideriamo adesso alcuni esempi di codice e valutiamone le proprietà nell'ipotesi in cui $\mathcal{X} = \{A, B, C, D\}$.

$\mathcal{X}$	C_1	C_2	C_3	C_4
A	0	0	10	0
B	1	010	00	10
C	1	01	11	110
D	1	10	110	111

- Il codice C_1 è un *codice singolare*, cioè è un codice che a simboli diversi associa la stessa parola codice; esso non consente di capire, in decodifica, quale simbolo tra B, C e D è stato trasmesso; noi vogliamo perciò codici non singolari;

- il codice C_2 è *non* singolare, ma non è decodificabile in maniera univoca. Ad esempio in fase di decodifica la stringa di simboli di alfabeto 010 può essere interpretata come il simbolo B oppure come la sequenza dei simboli C ed A: esistono per questo codice molte possibili decodifiche di sequenze di bit; un codice (non singolare) non è univocamente decifrabile se estendendo la sorgente si ottiene una codifica singolare, cioè se la corrispondenza $C : (x_i, x_j) \in \mathcal{X} \times \mathcal{X} \mapsto C(x_i, x_j) = C(x_i)C(x_j) \in \mathcal{D}$ è un codice singolare; noi vogliamo codici che siano *univocamente decifrabili* (u.d). Notiamo che i codici u.d. devono esserlo per ogni estensione della sorgente (a due, a tre, quattro simboli e così via; i codici non singolari garantiscono che le parole codice sono distinte per loro singolo uso). Quindi un codice u.d. è un codice non singolare tale che tutte le sue estensioni sono ancora non singolari;

- consideriamo adesso il codice C_3: se riceviamo la stringa 1100 questa la decodifichiamo con la sequenza di simboli C, B, ma se riceviamo la stringa seguente

$$\overbrace{11}^{AB}\overbrace{00}\overbrace{00}\overbrace{00}\overbrace{00}01\ldots$$

  non sappiamo decidere tra le due possibili decodifiche fino a quando non arriva il bit 1 che ci fa scegliere per la sequenza di simboli esatta, ciò perché la sequenza 01 non è presente nelle parole codice del nostro vocabolario: il codice C_3 è u.d. ma *non è istantaneo*. Un codice è istantaneo se nessuna parola codice è prefisso di un'altra; il codice C_4 rispetta la regola del prefisso quindi è certamente un codice istantaneo, ad esempio la stringa 01100100110111 si decodifica subito nella sequenza di simboli A, C, A, B, A, C, D. Notiamo che un codice a prefisso (istantaneo) è anche u.d. (un codice è u.d. quando l'estensione del codice, ovvero il codice applicato alla sorgente estesa, è non singolare).

I codici possono essere convenientemente rappresentate su di un albero. Ad esempio il codice C_4 può essere rappresentato mediante l'albero riportato in figura 1. Dato che il codice è binario ogni nodo dell'albero presenta al più due rami (*albero binario*). In figura 2 è mostrato l'albero del codice C_3, che, come abbiamo visto, *non* è istantaneo. É possibile dimostrare che un codice C è istantaneo se e solo se le parole codice si trovano unicamente sui nodi terminali dell'albero (*foglie*), cioè se percorrendo l'albero lungo un cammino che va dalla radice ad una parola codice, non si passa mai per un'altra parola codice.

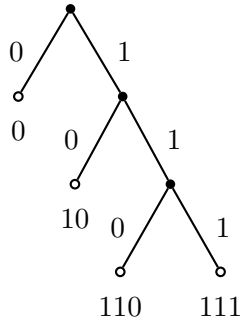


Figura 1: L'albero corrispondente al codice C_4

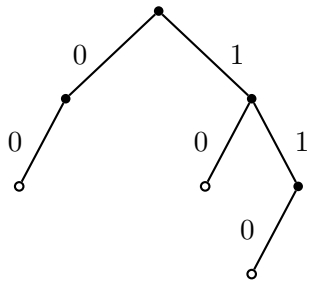
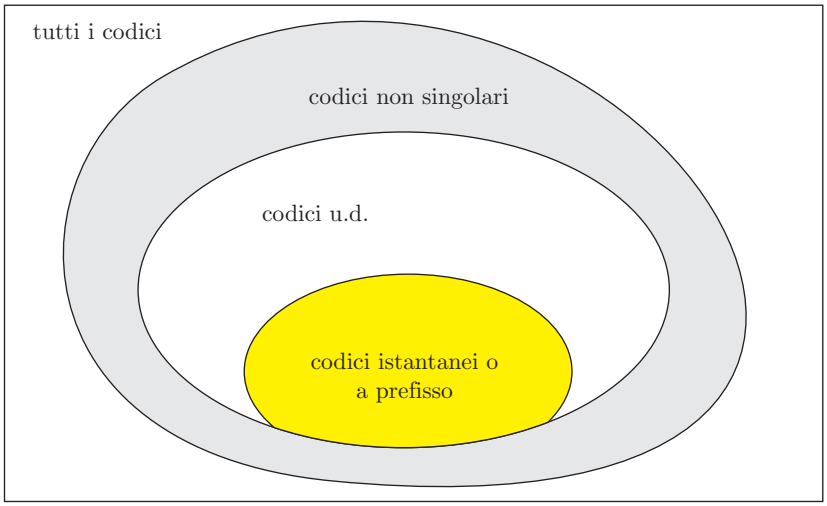


Figura 2: L'albero del codice C_3

La classificazione dei codici appena esposta può essere rappresentata mediante il seguente diagramma di Venn:



Per un codice C si definisce *lunghezza media di codifica* la quantità

$$L \triangleq \sum_{i=1}^{M} p_i l_i$$

dove l_i è la lunghezza della parola codice associata ad x_i che si verifica con probabilità p_i. Obiettivo ultimo della codifica è minimizzare la lunghezza media di codifica, e l'idea comune a tutti i codici per la compattazione dati consiste nell'associare parole-codice brevi a simboli più probabili e viceversa. Naturalmente non si possono prendere parole-codice piccole a piacere altrimenti si rischia di creare dei codici inutilizzabili, nei quali non si riesce a decodificare univocamente una sequenza di simboli emessi dalla sorgente. A tale proposito vale la condizione di *Kraft-McMillan*: per una sorgente M-aria è possibile realizzare un codice univoco purchè le lunghezze l_i delle parole codice rispettino la condizione

$$\sum_{i} 2^{-l_i} \leq 1$$

Al contrario, se tale condizione non è rispettata, nessun codice corrispondente è univocamente decodificabile. E' facile convincersi (in modo non troppo formale) della validità di questa condizione basandosi sulla visualizzazione dei codici mediante alberi (binari).

La conoscenza dell'entropia è molto utile per la compattazione, dato che fornisce, per il I Teorema di Shannon, un limite inferiore al numero medio di simboli binari necessari a codificare un simbolo di sorgente

$$L \geq H(\mathcal{S})$$

L'obiettivo delle tecniche di compattazione dati quindi è quello di associare alle parole emesse dalla sorgente parole codice che, pur preservando tutto il contenuto informativo dei dati, esibiscano un numero medio di bit/simbolo di sorgente che si attesti quanto più vicino possibile

all'entropia. Poichè le tecniche di compattazione si basano sul modello probabilistico assunto per la sorgente, appare evidente l'importanza di possedere una buona conoscenza delle statistiche per raggiungere elevati fattori di compressione. L'entropia di una sorgente costituisce un limite inferiore effettivo alla lunghezza media di codifica e quindi, se non è presente una sufficiente ridondanza statistica il rapporto di compressione può assumere valori anche molto bassi.

Le prestazioni di codifica possono essere migliorate, cioè la lunghezza media per simbolo diminuita, se si codificano congiuntamente blocchi di simboli. Detto $\mathbf{X}_n$ un blocco di n simboli, e P_n la sua distribuzione di probabilità, si dimostra che $H(\mathbf{X}_n) \leq \sum_i H(X_i)$, cioè l'entropia congiunta di n simboli è minore della somma delle entropie dei simboli considerati separatamente, con uguaglianza per simboli indipendenti. Più in generale, $H(\mathbf{X}_n)/n$ è monotonicamente decrescente con n. Detta poi L_n la lunghezza media per simbolo del codice che tratta blocchi di n simboli di sorgente, si dimostra che per ogni n vale

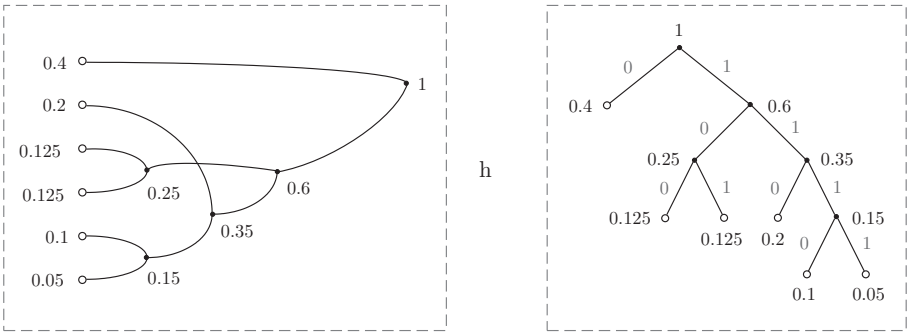
$$L_n \leq \frac{H(\mathbf{X}_n)}{n} + \frac{1}{n}$$

da cui si vede che all'aumentare di n la lunghezza media è maggiorata da una grandezza sempre più piccola, sia a causa del termine $1/n$ che per la decrescenza dell'entropia media $H(\mathbf{X}_n)/n$. Va detto, tuttavia, che la codifica congiunta di un numero elevato di simboli comporta notevoli problemi computazionali, per cui nella pratica ci si limita a blocchi di pochi simboli per volta.

3.1 Codifica di Huffman

L'algoritmo di Huffman permette di costruire il codice ottimo per una data sorgente (cioè per assegnate probabilità). Per sfruttare le statistiche della sorgente si utilizza un codice a lunghezza variabile, in cui le parole codice sono composte da un diverso numero di bit. L'idea è quella di assegnare ai simboli che si presentano più frequentemente un numero di bit minore rispetto a quelli che si presentano più raramente, allo scopo di minimizzare la lunghezza media delle parole codice utilizzate. Il codice si costruisce attraverso un albero binario in cui ad ogni simbolo corrisponde un nodo terminale e quindi una stringa di bit (la parola codice) che lo individua univocamente nell'albero. Sia la struttura dell'albero che l'associazione dei simboli ai nodi dipendono dalle probabilità d'emissione dei simboli stessi.

Le probabilità dei simboli vanno ancora ordinate in modo decrescente, l'albero del codice si costruisce facendo in modo che le due foglie inizialmente e poi i due nodi aventi le probabilità più basse siano figli di un unico nodo genitore (è un codice istantaneo): (sommando le probabilità



dei nodi non terminali e della radice si ottiene L) la lunghezza media di codifica per il codice di Huffman mostrato in figura è:

$$L = 0.4 + 3(0.25 + 0.2) + 4 \cdot 0.15 = 2.35 \text{ bit};$$

il codice di Huffman è il codice ottimo nel senso che è quello che ci consente di ottenere la lunghezza media di codifica minima. In realtà, la lunghezza media raggiunge il limite inferiore dettato dall'entropia solo nel caso in cui le probabilità sono del tipo $p_i = \frac{1}{2^{\ell_i}}$ (distribuzione diadica), mentre in generale ne rimane abbastanza lontano. Ciò avviene in particolar modo quando la sorgente ha pochi simboli e questi hanno probabilità molto sbilanciate. Ad esempio, se un simbolo ha probabilità 0.9 (e quindi autoinformazione 0.152 bit), per portare la lunghezza media di codifica al livello dell'entropia bisognerebbe poterlo codificare appunto con 0.152 bit, mentre si è ovviamente costretti ad usare ogni volta 1 bit.

Per ovviare a questo inconveniente, ed avvicinare effettivamente il limite dettato dall'entropia, basta però procedere alla codifica congiunta di più simboli della sorgente. Consideriamo ad esempio una sorgente iid (cioè di variabili aleatorie indipendenti ed identicamente distribuite) binaria con p=0.9, come nell'esempio precedente. Se codifichiamo congiuntamente due simboli successivi cioè una coppia scelta nell'alfabeto esteso (00, 01, 10, 11), troviamo che il simbolo 11 ha probabilità 0.81 e quindi autoinformazione 0.304 (ricordiamo che per eventi indipendenti l'autoinformazione è additiva) per cui dedicando un bit a tale simbolo esteso sprechiamo solo 0.696 bit in totale, e quindi 0.348 bit per simbolo originario.

Teniamo presente soltanto il fatto che spesso le probabilità dei simboli di sorgente non sono note a priori oppure variano nel tempo. In tal caso tali probabilità vanno stimate al volo (running estimates) in modo da costruire il codice di Huffman corrispondente o adattarne (con opportuni algoritmi veloci) uno già in uso. Svincolandosi dalle informazioni a priori si perviene ad un algoritmo di codifica "universale", cioè utilizzabile per una qualsiasi sorgente. Esso sarà un po' meno efficiente a causa della scarsa accuratezza iniziale delle stime, ma per file sufficientemente lunghi questa perdita diventa trascurabile. Questo limite induce ad utilizzare tecniche di codifica sub-ottime, come la codifica aritmetica o l'algoritmo di Lempel e Ziv, in cui sia possibile costruire il codice attraverso algoritmi iterativi.

3.2 Codifica aritmetica (Shannon-Fano-Elias)

La codifica aritmetica cerca di risolvere i problemi dell'algoritmo di Huffman, legati al fatto che al crescere della domensione del blocco la complessità cresce in modo esponenziale. Nella codifica aritmetica (almeno dal punto di vista concettuale) si divide l'intervallo unitario in M sottointervalli, tanti quanti sono i simboli della sorgente, ognuno di lunghezza pari alla probabilità del simbolo associato. Quindi, ad esempio

i	p_i	I_i	v_i
1	0.5	0.0–0.5	0.25
2	0.2	0.5–0.7	0.60
3	0.2	0.7–0.9	0.80
4	0.1	0.9–1.0	0.95

Associazione tra simbolo e valore in (0,1)

Di conseguenza, ogni simbolo è associato ad un particolare intervallo, per cui se riusciamo ad individuare univocamente un certo intervallo attraverso una stringa di bit (come al solito consideriamo codici binari) questa stringa codifica il simbolo corrispondente. Un modo di procedere potrebbe essere quello di prendere il centro di ogni intervallo e codificarne il valore in binario, ma tale soluzione presenta un ovvio inconveniente; si pensi ad esempio ad un intervallo il cui centro sia pari a 1/3: servirebbero infinite cifre binarie per codificarlo esattamente. Bisogna però tenere presente che non interessa individuare con precisione un punto, ma soltanto fare in modo che la stringa di codifica sia associata in modo non ambiguo ad un certo intervallo. In effetti, dando una successione di cifre binarie dopo la virgola, non si individua un punto ma un intervallo: perchè l'identificazione sia univoca ed istantanea basta che l'intero intervallo individuato dalla sequenza di bit cada all'interno dell'intervallo associato al simbolo. Ad esempio, la sequenza di cifre binarie 0.0 non indica il numero decimale 0 (dovrei scrivere 0.0-STOP) ma l'intero intervallo [0,0.5[, cioè quello compreso fra il più piccolo numero la cui rappresentazione binaria comincia con 0.0 (cioè appunto 0), e l'estremo superiore dei numeri la cui rappresentazione binaria comincia con 0.0, (infatti 0.0111111... < 0.5 decimale). Analogamente, 0.01 indica l'intervallo [0.25,0.5[, ecc. ecc. Allora, ci basta codificare un certo punto associato a un simbolo solo fino a quando l'intervallo d'incertezza nella rappresentazione binaria non ricade interamente nell'intervallo associato al simbolo.

In effetti, nella codifica aritmetica scegliamo di codificare proprio il centro dell'intervallo di simbolo, ma solo fino ad una opportuna precisione finita. A tal proposito, conviene sottolineare le seguenti cose:

1. nel codificare in binario un numero, ci si avvicina ad esso dal basso, ad esempio, se vogliamo codificare 0.80 avremo la sequenza

stringa	punto	intervallo
0.1	0.50	[0.50000–1.00000[
0.11	0.75	[0.75000–1.00000[
0.110	0.75	[0.75000–0.87500[
0.1100	0.75	[0.75000–0.81250[
0.11001	0.78125	[0.78125–0.81250[

2. con ogni nuovo bit, anche quando non varia il numero codificato, si dimezza l'incertezza, cioè l'intervallo cui il numero finale potrebbe appartenere (vedi sopra)

3. di conseguenza, quando si codifica il centro di un intervallo di simbolo posso essere sicuro di aver rimosso ogni ambiguità quando l'intervallo d'incertezza ha ampiezza minore o uguale alla metà dell'intervallo di simbolo; ad esempio, se voglio codificare 0.8, centro dell'intervallo 0.7-0.9 di ampiezza 0.2, basta che l'intervallo di incertezza nella codifica sia minore di 0.1 perchè si trovi tutto compreso in 0.7-0.9, per cui la rappresentazione corretta e sufficiente è 0.110 (ed in pratica la stringa 110).

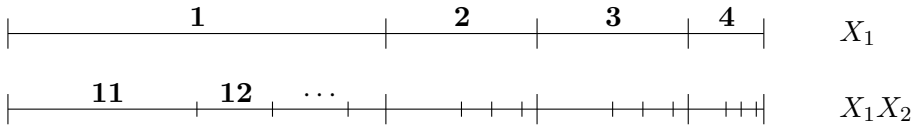
Si noti che il numero di bit necessari a specificare un intervallo è tanto maggiore quanto più piccolo è l'intervallo (in accordo con quanto suggerisce l'intuizione per un codice efficiente). In particolare, tenendo presente quanto detto prima, si può ricavare la lunghezza $l(x)$ della parola codice necessaria, dato che deve essere $2^{-l(x)} \leq p(x)/2$ e quindi $l(x) \geq \log(1/p(x)) + 1$. Quindi in definitiva avremo $l(x) = \lceil -\log p(x) \rceil + 1$.

A questo punto sono chiari i passi necessari per costruire (almeno in linea di principio, l'implementazione pratica è più complessa) un codice di Elias. Note le p_i, si individuano sul segmento [0,1] gli intervalli corrispondenti ai vari simboli ed i loro centri; dopodichè si codificano in binario tali centri con una precisione data da $l_i = \lceil -\log p_i \rceil + 1$. Ad esempio, per la sorgente considerata prima

i	p_i	I_i	v_i	l_i	c_i	I'_i
1	0.5	0.0–0.5	0.25	2	01	0.2500–0.5000
2	0.2	0.5–0.7	0.60	4	1001	0.5625–0.6250
3	0.2	0.7–0.9	0.80	4	1100	0.7500–0.8125
4	0.1	0.9–1.0	0.95	5	11110	0.9375–0.9687

Codice di Elias

E' evidente che il codice così costruito soddisfa la condizione di Kraft-McMillan, ed è anche chiaro che si tratta di un codice a prefisso, dato che le varie parole-codice rappresentano valori appartenenti ad intervalli disgiunti. La lunghezza media di codifica però è piuttosto elevata, $L = H(X) + 2$, e quindi qual è il vantaggio di usare questo tipo di codice invece, per esempio, di quello di Huffman? Tale vantaggio diventa chiaro quando si passano a considerare sorgenti estese, cioè si codificano congiuntamente blocchi di simboli iid, $\mathbf{X}^n = (X_1, \ldots, X_n)$. Prendendo per esempio il caso di due simboli, il primo simbolo divide in M parti l'intervallo [0–1], mentre il secondo simbolo divide in M parti, sempre proporzionali alle probabilità, ognuno dei primi intervalli.



Naturalmente, nel caso di più simboli si itera il procedimento. Anzitutto, procedendo come per il codice di Huffman, è immediato verificare che $L < H(X) + 2/n$ e quindi il codice di Elias è asintoticamente efficiente. Inoltre, e questa è la cosa più importante, data una lunghezza di blocco n, non è necessario generare a priori tutte le M^n parole codice possibili, in quanto esse si possono generare al volo, Mediante un algoritmo estremamente semplice, che si basa solo sulle probabilità dei successivi simboli emessi dalla sorgente. Così, ad esempio, se viene emesso il simbolo "2" l'intervallo corrispondente sarà [0.5–0.7], e il valore da codificare (con 4=$\lceil -\log 0.2 \rceil$ + 1 bit) sarà 0.6. Se poi viene successivamente emesso il simbolo "3" bisognerà aggiornare l'intervallo a [0.5+0.2x0.7–0.5+0.2x0.9], il valore da codificare a 0.66, e la precisione a 7 bit. In modo analogo si procede per i simboli successivi. In definitiva, non c'è bisogno di immagazzinare grandi codici nè di calcolare nulla a priori, tutto può essere realizzato in corso d'opera.

Questa descrizione in realtà è un pò semplificata e nasconde una lunga serie di problemi realizzativi legati per lo più alla precisione finita dei calcolatori. La codifica aritmetica è l'implementazione concreta, su macchine a precisione finita ed aritmetica intera, dei concetti già tutti presenti nella codifica di Elias. La codifica aritmetica è alla base dello standard JBIG (Joint Bi-level Image-processing Group) per la codifica di immagini a due toni (es facsimile).

Riferimenti bibliografici

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, 1948.

[2] T. Cover: *Elements of Information Theory*, Second Edition, Wiley Series in Telecommunications and Signal Processing, 2006.

[3] K. Sayood: *Introduction to Data Compression*, Second Edition, Morgan Kaufmann, 2000.