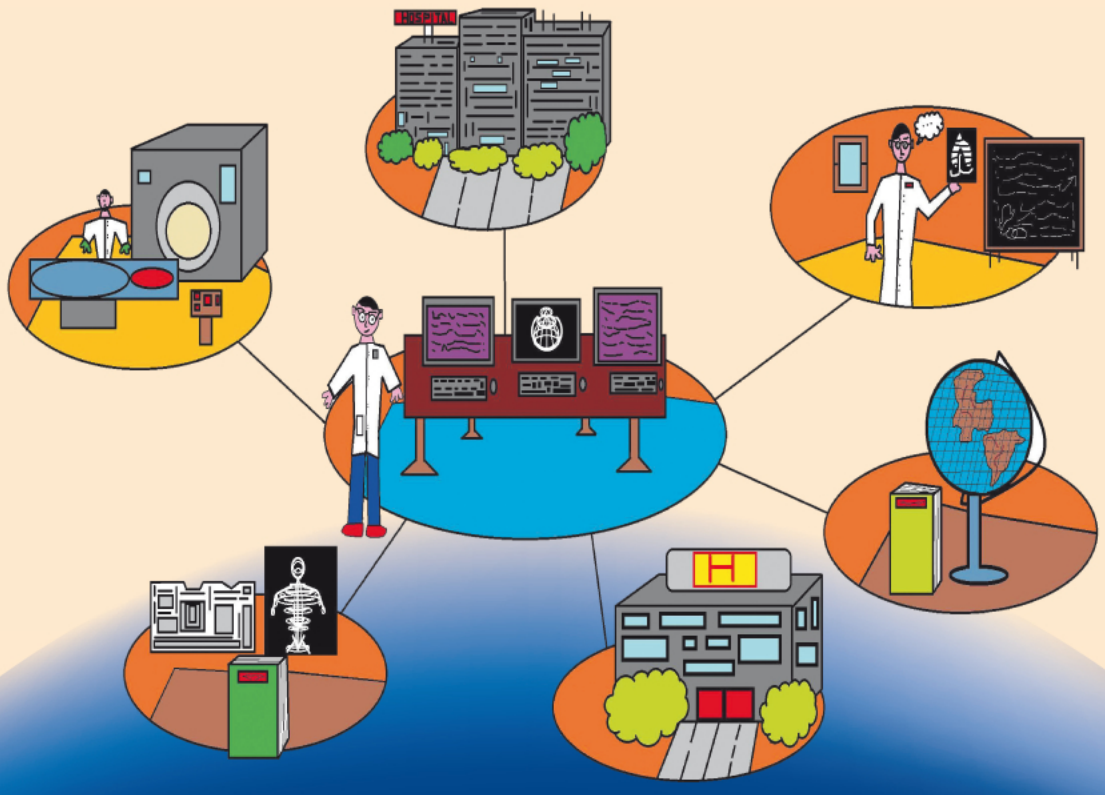
ROBERTO GRASSI • GIOVANNI PINTO • NICOLA SERRA

Sistemi per l'elaborazione dell'informazione



EdiSES

ROBERTO GRASSI GIOVANNI PINTO

NICOLA SERRA

SISTEMI PER L'ELABORAZIONE DELL'INFORMAZIONE

Raccolta del materiale didattico elaborato per gli studenti del corso di Laurea in Tecniche Radiologiche per Immagini e Radioterapia

Roberto Grassi, Giovanni Pinto, Nicola Serra
Sistemi per l'elaborazione dell'informazione

9 8 7 6 5 4 3 2 1
2019 2018 2017 2016

Le cifre sulla destra indicano il numero e l'anno dell'ultima ristampa effettuata

Copertina realizzata da Lorena Merchione su disegno di Daniele Pinto

Stampato presso la: Print Sprint srl - Napoli

Per conto della:

EdiSES srl – Piazza Dante 89 – 80135 Napoli

Tel. 081/7441706-07 Fax. 081/7441705

http://www.edises.it E-mail: info@edises.it

ISBN 978 88 7959 9047

Prefazione

La conoscenza dell'informatica rappresenta un importante bagaglio culturale che ogni studente oggigiorno deve possedere. In particolare con l'avvento di Internet, la conoscenza delle reti, della loro struttura e del loro funzionamento è diventato di fondamentale importanza anche in ambio medico.
Lo scopo di questo libro è presentare attraverso una descrizione completa delle reti e delle sue componenti, il loro utilizzo nel campo della Diagnostica per Immagini. Ovviamente per poter descrivere ciò, sono stati considerati tutti gli aspetti legati alle immagini analogiche e digitali, compreso la loro gestione. In particolare in merito alla condivisione tramite rete delle immagini digitali si sono affrontate problematiche legate alla loro compressione, codifica e ricostruzione, attraverso la descrizione dei più noti algoritmi in questo settore. Infine vengono descritti tutti quei problemi legati alla sicurezza informatica, come le password di autenticazione, la firma digitale, il virus informatico e le sue classificazioni. Concludendo vengono descritti anche gli aspetti legali, legati alla archiviazione digitale descrivendo gli standard di sicurezza dall'ITSEC ai Common Criteria.
Il testo è stato principalmente sviluppato per gli studenti Facoltà di Medicina e Chirurgia ed in particolare per gli specializzandi in radiologia e per gli studenti del corso di laurea in Tecniche Radiologiche per Immagini e Radioterapia, per i quali sono state descritte le applicazioni delle reti nel campo dell'imaging diagnostico. Questo testo potrebbe essere adottato anche per studenti di ingegneria biomedica, vista l'ampia e profonda descrizione nell'analisi dei sistemi per l'elaborazione delle informazioni.

Gli Autori

Roberto Grassi
Giovanni Pinto
Nicola Serra

Autori

Roberto Grassi

professore di ruolo di 1° fascia ordinario in Radiologia, presso la Facoltà di Medicina e Chirurgia, Seconda Università degli Studi di Napoli, Dipartimento Medico - Chirurgico di Internistica Clinica e Sperimentale "F. Magrassi e A. Lanzara", direttore della scuola di specializzazione in Radioterapia e presidente del corso di laurea triennale in Tecnico Sanitario di Radiologia presso la SUN. E' Autore/coautore di 46 libri e di 292 pubblicazioni sia su riviste nazionali che internazionali, con 2847 citazioni e un impact points pari a 428.54. Responsabile di 18 progetti di ricerca per il MIUR, il CNR, il II Ateneo di Napoli e la Regione Campania. Ha svolto attività di insegnamento presso l'Università dal 1994 ad oggi, sia in Italia che all'estero nell'ambito della diagnostica per immagini e radioterapia.

Ha prestato servizio: dal 1980 al 1992 presso il II Servizio di Radiologia della Facoltà di Medicina e Chirurgia dell'Università Federico II di Napoli. dal 1993 al 1998 presso il II Servizio di Radiologia dell'Azienda Ospedaliera di Rilievo Nazionale A. Cardarelli di Napoli. dal 1998 ad oggi presso l'Azienda Universitaria della Facoltà di Medicina e Chirurgia della Seconda Università di Napoli, ricoprendo l'incarico di responsabile U.O.S. di TC e RM.

Ha svolto soggiorni di approfondimento in USA, in Giappone e Germania. L'attività scientifica e di ricerca è rivolta principalmente alle patologie addominali, le applicazioni digitali, le problematiche gestionali e le patologie d'urgenza e recentemente per studi longitudinali su piccoli animali investigati con apparecchi ad alta risoluzione.

Giovanni Pinto

Funzionario Programmatore Esperto - Centro Elaborazione Dati - dell'Azienda Ospedaliera "San Giuseppe Moscati"- Avellino. Conoscenze acquisite nel campo della programmazione e gestione dei siti Internet in linguaggio HTML ed in particolare Referente e Responsabile del sito Internet dell'Azienda Ospedaliera S. G. Moscati. Organizzatore e relatore dei Progetti Formativi Aziendali ECM per tutto il personale Sanitario ed Amministrativo sull'uso pratico nell'utilizzo del computer e delle procedure informatiche utilizzate nell'Azienda: uso della Firma Digitale, uso della PEC, utilizzo dell'Intranet Ospedaliera. Nomina Responsabile Aziendale della Conservazione - progetto SURAFS. Ha partecipato a vari convegni come relatore ECM. Docente a contratto presso la Facoltà di Medicina e Chirurgia, Seconda Università degli Studi di Napoli, Dipartimento Medico-Chirurgico di Internistica Clinica e Sperimentale "F. Magrassi e A. Lanzara" al Corso di Laurea in Tecniche Radiologiche per Immagini e Radioterapia. Dall'anno 2001/2002 a tutt'oggi ha ricevuto l'incarico di insegnamento al Corso di Laurea in Tecniche Radiologiche per Immagini e Radioterapia presso il polo didattico di Avellino dell'Azienda Ospedaliera S. G. Moscati - attivato presso la Seconda Università degli Studi di Napoli. E' coautore di 1 libro con la casa Editrice Springer e di tre pubblicazioni. Ha conseguito presso l'Università degli Studi di Salerno la Laurea in "Scienze dell'Informazione" della Facoltà Scienze Matematiche, Fisiche e Naturali. (Equipollenza della Laurea in Scienze dell'Informazione alla Laurea in Informatica. (GU n.101 del 3.5.2000) e Laurea Triennale in Tecniche di Radiologia Medica, per Immagini e Radioterapia conseguito presso l'Università degli Studi di Napoli Federico II. Dal 1988 al 2001 ha prestato Servizio presso L'Azienda Ospedaliera San Giuseppe Moscati - Avellino - Dipartimento Diagnostica per Immagini con esperienza lavorativa su Tomografia Computerizzata e Risonanza Magnetica. Dal 2001 a tutt'oggi presta servizio con competenze acquisite nel trattamento delle informazioni e dei software presso il Centro Elaborazione Dati dell'Azienda Ospedaliera S. G. Moscati - Avellino.

Nicola Serra

Ricercatore a contratto presso la Facoltà di Medicina e Chirurgia, Seconda Università degli Studi di Napoli, Dipartimento Medico-Chirurgico di Internistica Clinica e Sperimentale "F. Magrassi e A. Lanzara". Laureato in matematica e dottorato in Biologia Computazionale. E' autore/coautore di 12 articoli sia su riviste nazionali che internazionali con 24 citazioni, un impact points pari a 13.21 e di 5 atti a convegno. Ha svolto per l'Università, per enti pubblici e privati, attività di ricerca e di didattica a partire dal 2001 ad oggi, in ambito Matematico, Statitico e Bioinformatico, in merito a problematiche riguardanti la Biomedicina, il Protein Folding, i Grandi Database, le Tecniche di Digitalizzazione e di Ricostruzione delle Immagini Digitali, le reti neurali, i data mining, la Teoria delle reti e dei sistemi.

15 La compressione Lossless delle immagini

Indice dei contenuti

15.1 Introduzione

Si verificano diversi casi in cui dall'informazione compressa, sia possibile ricostruire l'informazione originaria, senza che questa venga minimamente alterata, ossia il flusso di bit prima del processo di compressione deve risultare esattamente identico al flusso di bit dopo il processo di decompressione. Le tecniche di compressione, che operano in questo modo vengono pertanto chiamate *reversibili* (con riferimento al fatto che il processo di compressione è reversibile) o più comunemente *lossless* (senza perdita). Nel caso di compressione lossless quindi, l'unica possibilità di un'alterazione dei bit nel flusso di dati binario decompresso rispetto a quello originale, può essere attribuita al rumore introdotto nel canale di comunicazione, contemporaneamente ad una non efficace codifica di canale. La compressione lossless basata sulle tecniche di codifica entropy encoding, opera sopprimendo, per quanto possibile, le ridondanze presenti nei flusso di bit che costituiscono il messaggio sorgente.

15.2 Proprietà dei codici

Per *codice a blocchi* si intende un codice in cui i simboli che costituiscono il messaggio sono organizzati in blocchi di *lunghezza prefissata*. Nei *codici a lunghezza variabile*, la lunghezza di ogni simbolo non è fissata, ma può variare da simbolo a simbolo. Gli algoritmi di compressione basati sulla codifica **entropy encoding**, si basano su trasformazioni (attraverso una mappatura uno a uno) di codici a blocchi in codici a lunghezza variabile. La compressione si ottiene quindi, quando il messaggio codificato contiene meno bit del messaggio sorgente. Dalla decompressione del messaggio codificato è possibile ricostruire esattamente il messaggio sorgente.

Simboli del messaggio sorgente	Simboli del messaggio codificato
000	**0**
010	**10**
110	**11**

Figura 15.1. Esempio di mappatura di un codice a blocchi in un codice a lunghezza variabile. Il numero di bit complessivo del codice a lunghezza variabile è minore rispetto a quello del codice a blocchi.

Nei codici a blocchi risulta abbastanza agevole distinguere un simbolo rispetto al successivo, poiche nel flusso di dati ogni simbolo si ripresenta ad intervalli di lunghezza prefissata. Per i codici a lunghezza variabile, potrebbe invece risultare estremamente difficile, in termini di tempo di elaborazione o addirittura impossibile discernere un simbolo dal successivo.
I codici detti ***univocamente decodificabili*** godono della proprietà secondo la quale, ad ogni possibile sequenza di simboli del messaggio sorgente, deve corrispondere un'unica e diversa sequenza di simboli nel messaggio codificato. Questa condizione (necessaria e sufficiente), non è però applicabile nella pratica. Ad esempio, il codice composto dai simboli:

$$S_1 = 0,\ S_2 = 01,\ S_3 = 11 \text{ ed } S_4 = 00,$$

non è univocamente decodificabile in quanto la sequenza 0011 può essere interpretata sia come successione dei simboli S_4 ed S_3, che come successione dei simboli S_1, S_2, S_3.
Una seconda fondamentale proprietà è la ***decodificabilità istantanea***. Nei codici che soddisfano questa proprietà, dal flusso di dati si può sempre stabilire quando si è ricevuto completamente un simbolo, senza dover aspettare di riceverne altri. Condizione necessaria e sufficiente affinché un codice sia istantaneamente decodificabile è che nessun simbolo sia il prefisso di un altro simbolo. Il codice:

$$S_1 = 0,\ S_2 = 10,\ S_3 = 110,\ S_4 = 1110 \text{ ed } S_5 = 1111,$$

è un esempio di codice istantaneamente decodificabile. Se un codice è ***istantaneamente decodificabile*****,** si può dimostrare che è anche ***univocamente decodificabile*****.**

15.3 Il codice di Huffmann

Questo algoritmo non distruttivo, realizzato nel 1952 dal matematico *David A. Huffman*, permette di attribuire una parola di codice binario, ai diversi simboli da comprimere (pixel o caratteri ad esempio). La lunghezza di ogni parola del codice non è identica per tutti i simboli, infatti secondo l'algoritmo i simboli più frequenti sono codificati con delle piccole parole di codice, mentre i simboli più rari ricevono dei codici binari più lunghi. Si parla allora di codifica a lunghezza variabile prefissata, per identificare questo tipo di codifica, dato che nessun codice è il prefisso di un altro. Così, la serie finale di parole codificate a lunghezza variabile sarà in media più piccolo rispetto ad un codice di dimensione costante. La costruzione dell'albero (vedi Figura 15.3) delle associazioni avviene attraverso i seguenti cinque passi:

1. Viene analizzato e conteggiato il **numero di ricorrenze** degli elementi di base del *file* da comprimere: i singoli caratteri in un *file* di testo, i *pixel* in un *file* grafico;
2. I due elementi meno frequenti sono accomunati in una categoria che li rappresenta entrambi. Così ad esempio se X ricorre 8 volte e Y 7 volte, viene creata la categoria XY, dotata di 15 ricorrenze. Intanto i componenti X e Y ricevono ciascuno un differente **marcatore** che li identifica come elementi entrati in un'**associazione**;
3. Vengono identificati i due successivi elementi meno frequenti nel file e li si riunisce in una nuova categoria, usando lo stesso procedimento descritto al punto 2. Il gruppo XY può a sua volta entrare in nuove associazioni e costituire, ad esempio, la categoria XYZ. Quando ciò accade, la X e la Y ricevono un nuovo **identificatore di associazione,** che finisce con l'allungare il codice ed identificherà univocamente ciascuna delle due lettere nel **file compresso** che verrà generato;
4. Viene creato quindi, per passaggi successivi, un **albero** costituito da una serie di **ramificazioni binarie**, all'interno del quale appaiono con maggiore frequenza ed in combinazioni successive gli elementi più rari all'interno del file, mentre appaiono più raramente gli elementi più frequenti. In base al meccanismo descritto, ciò fa sì che gli elementi rari all'interno del **file non compresso,** siano associati ad un **codice identificativo** lungo, che si accresce di un elemento ad ogni nuova **associazione**. Gli elementi invece che si ripetono più spesso nel *file originale*, sono anche i meno presenti nell'**albero delle associazioni**, sicché il loro **codice identificativo** sarà il più breve possibile;
5. Viene generato il **file compresso**, sostituendo a ciascun elemento del **file originale** il relativo codice prodotto al termine della **catena di associazioni,** basata sulla frequenza di quell'elemento nel documento di partenza.

Il **guadagno di spazio** al termine della compressione è dovuto al fatto che gli elementi che si ripetono frequentemente sono identificati da un codice breve, che occupa meno spazio di quanto ne occuperebbe la loro codifica normale. Viceversa gli elementi rari nel **file originale** ricevono nel **file compresso** una codifica lunga, che può richiedere, per ciascuno di essi, uno spazio anche notevolmente maggiore di quello occupato nel **file non compresso**.
Dalla somma algebrica dello spazio guadagnato con la codifica breve degli elementi più frequenti e dello spazio perduto con la codifica lunga degli elementi più rari, si ottiene il **coefficiente di compressione** prodotto dall'**algoritmo di Huffman**. Da quanto detto si deduce che questo tipo di compressione è tanto più efficace quanto più ampie sono le differenze di frequenza degli elementi che costituiscono il **file originale**, mentre scarsi sono i risultati che si ottengono quando la distribuzione degli elementi è uniforme.
Un'esempio dimostrativo del funzionamento di questo algoritmo è visionabile su Internet all'indirizzo **http://www.cs.sfu.ca/CC/365/li/squeeze/Huffman.html**. Qui si troverà un'applet java in grado di eseguire la generazione dell'albero delle associazioni, di produrre il codice compresso e di calcolare il coefficiente finale di compressione.
Quindi è ragionevole aspettarsi che i simboli che costituiscono il messaggio sorgente non abbiano tutti la stessa frequenza, ma che alcuni simboli si presentino con una frequenza maggiore. I codici di Huffman

traggono vantaggio dal fatto che i simboli del messaggio sorgente con frequenza maggiore, vengono codificati con simboli di lunghezza minore nel messaggio codificato, mantenendo la proprietà di decodificabilità istantanea. Per cui un messaggio costituito da ***n*** simboli si indica con

$$P_i = \frac{\text{numero di occorrenze del simbolo i-esimo}}{\text{numero totale di simboli}}, \qquad i = (1,...,n)$$

in cui P_i è la probabilità dell'i-esimo simbolo. Senza perdere di generalità, si può supporre che le probabilità di occorrenza dei simboli P_i siano ordinate in ordine decrescente: $P_1 \geq P_2 \geq P_3 \geq ... \geq P_n$ e definendo la lunghezza media del messaggio la quantità:

$$L_{av} = \sum_{i=1}^{n} P_i l_i$$

in cui l_i indica la lunghezza del simbolo. In particolare diremo che un codice è **efficiente** se verificherà contemporaneamente le due disuguaglianze:

1) $P_1 \geq P_2 \geq P_3 \geq ... \geq P_n$

2) $l_1 \leq l_2 \leq l_3 \leq ... \leq l_n$

In questo caso L_{av} assumerà il valore minimo. La costruzione di un codice di Huffman avviene attraverso i **seguenti passi** successivi:

PRIMO PASSO

Il messaggio sorgente deve essere letto una prima volta, in modo da determinare le probabilità associate ad ogni simbolo. Si prenda ad esempio un messaggio sorgente composto da una data successione di simboli di 3 bit con le corrispondenti probabilità come indicato nella Figura 15.2

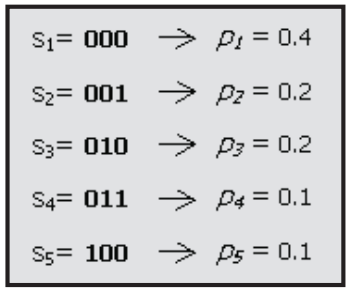


Figura 15.2. Probabilità associate a ciascun simbolo

SECONDO PASSO

Successivamente si costruisce una tabella in cui nella prima colonna, i simboli del messaggio sorgente sono disposti in ordine decrescente di probabilità. Si raggruppano quindi i due simboli meno probabili sommando le loro probabilità e riportando il risultato in una nuova colonna, riordinando se necessario, in modo che le probabilità siano sempre in ordine decrescente. Il processo viene ripetuto fino a quando non restano solamente due valori di probabilità, come mostrato nella seguente Figura 15.3. Tale processo viene detto di processo di riduzione.

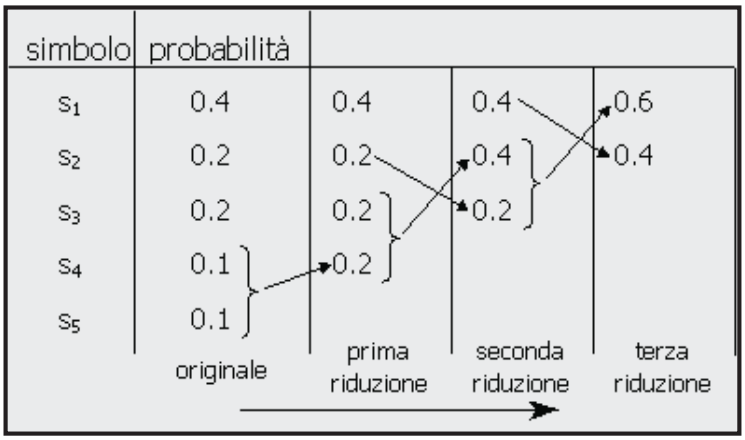


Figura 15.3. Processo di riduzione nella costruzione di un codice di Huffman

TERZO PASSO

Il codice viene infine effettivamente generato nel processo detto di separazione. Nell'ultima colonna dalla tabella generata al punto precedente erano rimasti due soli valori, questi si possono codificare semplicemente attribuendogli i bit 0 e 1. Passando, alla colonna immediatamente a sinistra (ultima colonna), i valori di probabilità presenti sono nuovamente separati in due e la codifica avviene aggiungendo all'ultima il bit 0 al primo e il bit 1 al secondo, come indicato nella Figura 15.4.

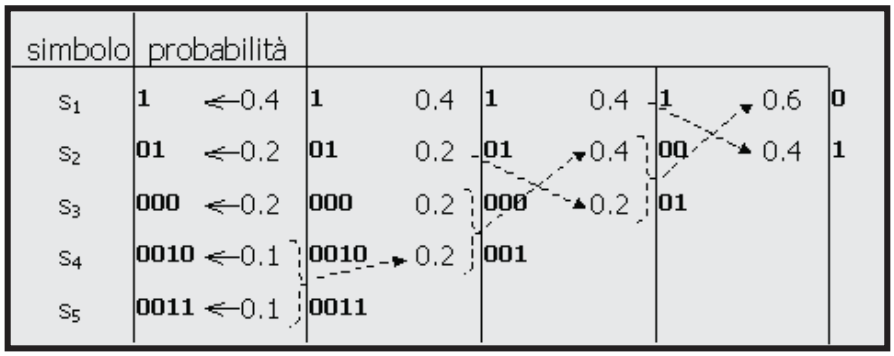


Figura. 15.4. Processo di separazione nella costruzione di un codice di Huffman

Al termine di questo processo, dal messaggio sorgente composto dai cinque simboli S_1, …,S_5 viene generato il codice composto dai cinque simboli S_1, …,S_5, codificati in codice binario con un minor numero di bit (Figura 15.4). Per cui avremo che:

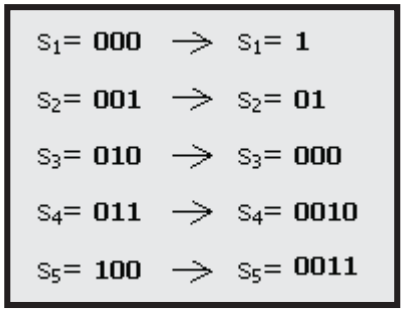


Figura 15.5

Codici di Huffmann adattivi

Il processo descritto sopra per la costruzione di un codice di Huffman, viene detto statico, in quanto richiede che un intero blocco di dati venga esaminato in modo da determinare le probabilità di ciascun simbolo, prima di poter procedere alla costruzione del codice vero e proprio. I codici di Huffman ***adattivi***, pur continuando a basarsi sulla attribuzione di codici di lunghezza minore ai simboli che risultano avere una maggiore probabilità di occorrenza, vengono generati man mano che i simboli del messaggio sorgente vengono "letti" dal codificatore. Nei codici di Huffman *adattivi* infatti, le probabilità dei simboli non vengono calcolate esattamente, ma sono predette, ed eventualmente "adattate" contemporaneamente alla generazione del codice. Nella seguente Figura 15.6 è descritto il sistema che porta alla generazione di un codice di *Huffman adattivo*

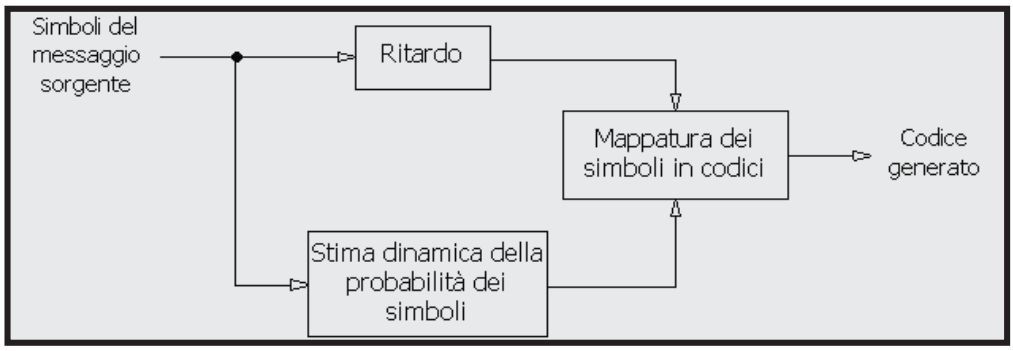


Figura. 15.6. Generazione di un codice di Huffman adattivo

Gli algoritmi utilizzati per la costruzione di un codice di Huffman adattivo comportano maggiori difficoltà rispetto a quelli statici. I codici di Huffman adattivi sono comunque importanti, in quanto, pur garantendo livelli di efficienza paragonabili a quelli dei codici di Huffman statici, presentano il vantaggio poter codificare un flusso continuo di informazioni digitali, quale ad esempio, quello prodotto dall'acquisizione e digitalizzazione dell'audio o del filmato video, quando sia necessaria la compressione in tempo reale come nel caso della lezione multimediale interattiva a distanza.

Decodifica dei codici di Huffman

La decodifica dei codici di Huffman può avvenire in **due** diversi modi:

Il primo attraverso la decodifica detta bit-seriale, consente di realizzare un decodificatore con un tasso di lettura dei bit del messaggio codificato costante, ma con un tasso di uscita dei simboli decodificati variabile a seconda della lunghezza dei codici.

Il secondo modo consiste nel generare una tabella di 2L righe, dove L è la lunghezza del codice più lungo del messaggio codificato. Il vantaggio di questo modo di procedere, consiste nel fatto che il decodificare ha un tasso costante di uscita dei simboli decodificati, ma non è però costante il tasso di lettura dei bit del messaggio codificato.

15.4 Codifica Aritmetica e codifica RLE (Run Length Encoding)

Oltre ai codici di Huffman, esistono svariate altre tecniche per comprimere in modo lossless l'informazione digitale. Tra queste, la codifica aritmetica e la codifica ***RLE*** (***Run Length Encoding***) sono frequentemente usate nella compressione dell'informazione multimediale. La codifica aritmetica trae vantaggio nel raggruppare e codificare con un singolo codice, un intero blocco di simboli dei dati da comprimere. La RLE è un processo di codifica nel quale vengono identificate nel flusso di dati da comprimere, sequenze di uno stesso simbolo, le quali vengono successivamente codificate con un unico codice che deve anche contenere

ovviamente, il numero di volte che il simbolo si ripete. Proviamo ora a vedere come funziona un sistema di compressione non distruttivo RLE, acronimo di Run Lenght Encoding, che potrebbe essere tradotto in italiano con *codifica della lunghezza di stringa*. In questo tipo di compressione, ogni serie ripetuta di caratteri (o run, in inglese), viene codificata usando solo due byte: il primo è utilizzato come contatore e serve per memorizzare quanto è lunga la stringa, il secondo contiene invece l'elemento ripetitivo che costituisce la stringa. Si supponga di voler comprimere in questo formato un file grafico contenente un grande sfondo di un solo colore uniforme. Tutte le volte che l'analisi sequenziale del file s'imbatterà in stringhe di caratteri uguali e ciò accadrà spesso nella scansione dello sfondo omogeneo, le serie ripetitive potranno essere ridotte a due caratteri soltanto: uno che esprime il numero delle ripetizioni, il secondo il valore che si ripete. Il risparmio di spazio sarà direttamente proporzionale al livello di uniformità presente nell'immagine. Provate invece ad usare il sistema RLE su una foto piena di colori differenti e di transizioni sfumate, il risparmio di spazio sarà notevole, perché poche saranno le stringhe di ripetizioni che l'algoritmo riuscirà a trovare leggendo sequenzialmente il file. Pensate infine, al caso limite di un'immagine creata artificialmente, come quella riportato in Figura 15.7, contenente una serie di pixel tutti differenti l'uno dall'altro nei valori cromatici. In questo caso, l'uso della compressione RLE, si dimostra addirittura controproducente.

Figura 15.7. Immagine ingrandita di un file di 16 × 16 pixel, costituito da 256 colori unici differenti

Questo file, salvato in formato BMP non compresso, occupa 822 byte. Salvato invece sempre in formato BMP, ma utilizzando l'algoritmo RLE, occupa 1400 byte, cioè 1,7 volte la sua grandezza originale.

15.5 Codifica Differenziale

Un'altra tecnica molto usata nella compressione Lossless delle immagini è la codifica Differenziale. Sfruttando la forte correlazione tra i pixel adiacenti, soprattutto nelle zone con tonalità grossomodo uniforme, si può diminuire il valore dell'entropia del messaggio sorgente e quindi rendere la distribuzione statistica dei simboli, passibile di rapporti di compressione più elevati, se consideriamo come simboli da codificare, non il valore associato ad ogni pixel, ma la differenza tra il valore associato ad un pixel e il successivo. Se i valori dei pixel di un'immagine sono: $x_1, x_2, x_3, \ldots, x_n$, i simboli da codificare diverranno:

$$y_i = x_i - x_{i-1} \quad \text{con} \quad i = 1,...,n \quad \text{ed} \quad x_0 = 0.$$

e i nuovi simboli così ottenuti sono chiamati *residui di predizione* degli x_i. Una tipica distribuzione dei simboli x_i e dei corrispondenti residui di predizione y_i è mostrata in Figura 15.8.

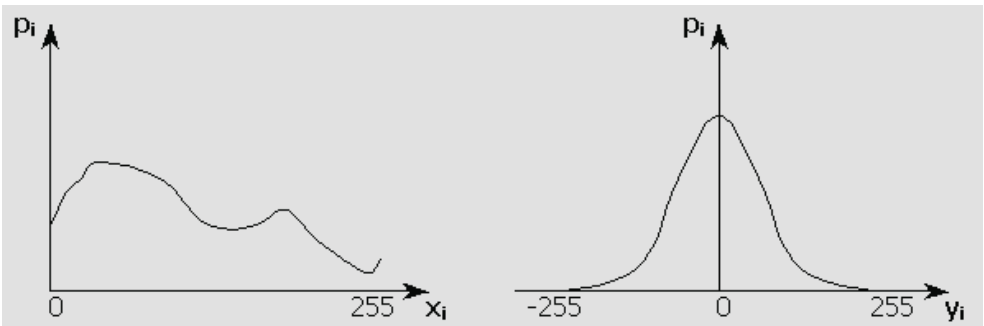


Figura 15.8. Rappresentazione tipica di una distribuzione di probabilità dei valori dei pixel di un immagine x_i presi in sequenza e dei corrispondenti residui di predizione y_i.

Si può osservare dalla Figura 15.8 che l'entropia dei residui di predizione è minore rispetto a quella del valore dei pixel. In particolare gli intervalli di x_i ed y_i sono rispettivamente [0,255] e [-255, 255].
La codifica differenziale in se stessa non comporta una compressione dell'informazione, ma rende solo più efficace la codifica successiva. La codifica differenziale è quindi un esempio di source encoding nel quale viene ridotta l'entropia del messaggio sorgente e al quale segue sempre un processo di entropy encoding.
Infine un'altro algoritmo non distruttivo, che va sotto il nome di LZW, è il risultato delle modifiche apportate nel 1984 da Terry Welch ai due algoritmi sviluppati nel 1977 e nel 1978 da **Jacob Ziv** e **Abraham Lempel**, e chiamati rispettivamente LZ77 e LZ78.

Nota. *Definizione di entropia della sorgente.*
Si definisce entropia di una sorgente i cui simboli sono S_i con i = 1, …, n, la quantità:

$$H(S) = \sum_{i=1}^{n} P_i \log_2 \left(\frac{1}{P_i} \right)$$

dove P_i denota la probabilità di occorrenza del simbolo i-esimo. L'entropia fornisce una misura della quantità di informazione contenuta nel flusso di bit, che deve essere sottoposto a compressione.

16 La compressione Lossy delle immagini

Indice dei contenuti

16.1 Introduzione

Nella compressione Lossy, l'informazione digitale viene codificata in modo irreversibile, vale a dire che dopo la compressione non è più possibile ricostruire esattamente la stessa informazione iniziale, che era contenuta nel flusso di dati binario. La compressione Lossy rientra nella categoria denominata **source encoding** in quanto gli algoritmi adottati traggono tutti vantaggio dalle correlazioni che, a seconda del tipo di informazione da comprimere (immagini, filmati, ecc.), sono presenti all'interno del flusso di dati binario. In questo paragrafo verranno prese in esame alcune tecniche di compressione lossy per le immagini fisse utilizzate nello standard di compressione ISO **JPEG** (Joint Photographic Experts Group). Nelle immagini fisse si ha una correlazione di tipo spaziale tra i punti (pixel) adiacenti che la definiscono. Molte immagini infatti, presentano delle zone più o meno vaste, con una tonalità uniforme. Se in queste zone il valore del pixel varia di poco, accade che le corrispondenti differenze nell'immagine visualizzata sullo schermo, siano praticamente impercettibili. D'altra parte, uniformando il valore dei pixel di queste zone dell'immagine al loro valore medio, diventa possibile codificare un'intera zona dell'immagine con un unico valore. La tecnica utilizzata nello standard JPEG si basa sulla trasformata **DCT** (Discrete Cosine Transform), che verrà discussa in questa sezione.

16.2 Tecniche di compressione a campioni e codifica DPCM (Modulazione Codificata di Impulsi)

Prima di passare alla descrizione delle tecniche basate sulla trasformata DCT, che rientrano nella categoria della **codifica a blocchi**, in quanto l'immagine viene codificata analizzando un intero blocco di dimensioni generalmente pari 8 × 8 pixel, verrà accennata la codifica DPCM (Differenzial Pulse Code Modulation), che rappresenta un esempio di **codifica a campioni**.
Nelle tecniche di codifica a campioni l'immagine viene compressa codificando un pixel alla volta. La codifica a campioni ha quindi il vantaggio di essere concettualmente più semplice rispetto alla codifica a blocchi e di conseguenza consente un'implementazione meno complessa, con l'ulteriore vantaggio di una maggiore velocità di elaborazione, che viene generalmente espressa come numero di campioni codificati (e decodificati) per unità di tempo. Lo svantaggio invece, consiste nel fatto che il rapporto di compressione della codifica a campioni è minore di quello raggiungibile con la codifica a blocchi.
Nella tecnica di compressione DCPM, (DIFFERENTIAL PULSE CODE MODULATION) dal flusso di dati binario che contiene i campioni dell'immagine, viene letto un campione alla volta. Per ciascun campione $X_{i,j}$, dove i e j sono le coordinate del campione, è generato un segnale residuo, denotato con $e_{i,j}$ ottenuto sottraendo a $X_{i,j}$ il valore di predizione $P_{i,j}$. Nella seguente Figura 16.1, è rappresentato uno schema a blocchi della tecnica di compressione DCPM.

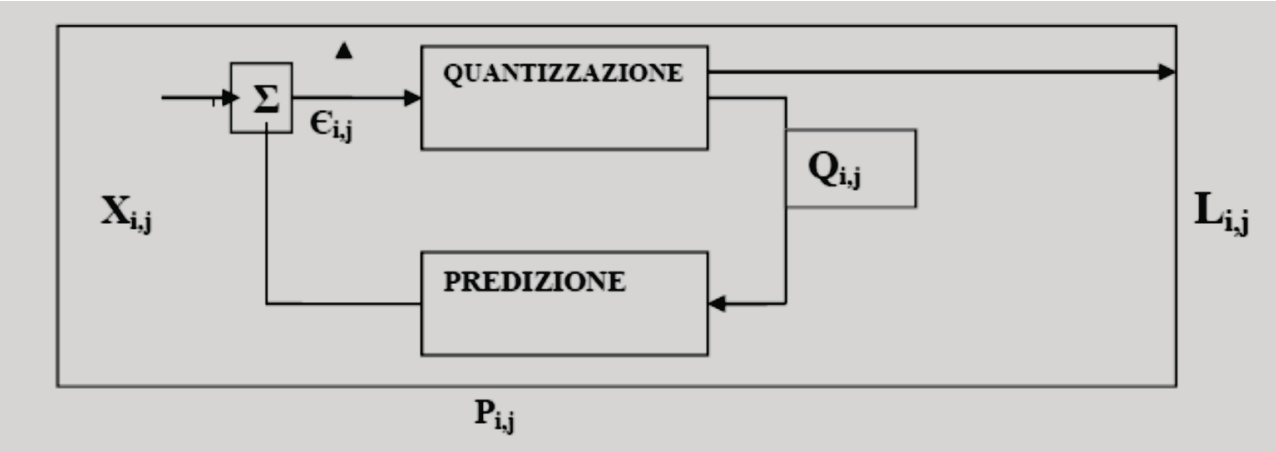


Figura 16.1. Schema a blocchi della tecnica di compressione DPCM

Il valore di predizione $P_{i,j}$ si può calcolare dal valore dei campioni che si trovano nelle immediate vicinanze di $X_{i,j}$, utilizzando ad esempio una formula del tipo:

$$P_{i,j} = w_1 X_{i,j-1} + w_2 X_{i-1,j-1} + w_3 X_{i-1,j}$$

Dove w_1, w_2, w_3, sono delle costanti. Il processo finora descritto è molto simile alla codifica differenziale, utilizzata per la compressione Lossless delle immagini.
L'irreversibilità viene introdotta dal processo di quantizzazione. I residui $\boldsymbol{e_{i,j}}$, per quelle parti dell'immagine con tonalità pressoché costante, hanno un valore prossimo allo zero. Il processo di quantizzazione riduce questo valore esattamente a zero. In definitiva, nelle zone dell'immagine caratterizzate da una forte correlazione spaziale, i valori in uscita dal codificatore DPCM saranno quasi tutti zero e quindi passibili di elevati rapporti di compressione a causa del ridotto valore di entropia, se si sottoponesse il flusso di dati così ottenuto, ad una successiva compressione Lossless. Nella decodifica il flusso di dati compresso viene sottoposto al processo di quantizzazione inversa. In questo processo i valori che erano stati portati a zero dal quantizzatore, continuano ovviamente a rimanere zero ed è questa la principale causa di irreversibilità della compressione DPCM.

16.3 Drastiche riduzioni di peso con lo standard JPEG

La sigla JPEG identifica una commissione di esperti denominata **Joint Photographic Expert Group**, formata nel 1986 con lo scopo di stabilire uno standard di compressione per le immagini a tono continuo - cioè di tipo fotografico - sia a colori sia in bianco e nero. Il lavoro di questa commissione ha portato alla definizione di una complessa serie di algoritmi, approvata come standard ISO nell'agosto del 1990 e successivamente divenuta la raccomandazione T.81 (9/92) dell'ITU, International Telecommunication Union. Si può leggere la versione integrale in formato PDF dal sito del W3C. L'indirizzo è http://www.w3.org/Graphics/JPEG/itu-t81.pdf. Il documento è composto da 186 pagine fitte di definizioni, procedure, grafici, immagini esemplificative e formule matematiche.

Il JPEG è dunque uno standard industriale e non va confuso con il formato di file JPG, che rappresenta di volta in volta, a seconda della software house che lo implementa, un sottoinsieme variabile e non sempre universalmente compatibile con lo standard di riferimento.

Nota: In ogni caso Jpeg è il nome del formato e jpg l'estensione, per chiarire meglio questa sottile differenza basti pensare alla differenza fra "**.html**" e "**.htm**".

In pochi sanno ad esempio, che le specifiche JPEG descrivono anche un formato di compressione non distruttivo, basato su tecniche differenti da quelle che descriveremo di seguito, del quale si è ormai persa traccia. I meccanismi di compressione risultano particolarmente adatti ad immagini contenenti un elevato numero di colori (immagini True Color), a gradazioni di grigio e per la memorizzazione di immagini fotografiche o di disegni molto sfumati. Negli altri casi, come cartoons, forniscono risultati generalmente peggiori di altri formati, sia in termini di qualità sia di dimensione dei file.

L'algoritmo JPEG ottiene una buona compressione sfruttando le limitazioni conosciute dell'occhio umano. Infatti è stato progettato in modo che i pixel eliminati siano quelli meno percettibili, ad esempio se si trovano due pixel con minima variazione di luminosità vengono conservati, ma se si trovano due pixel con minima variazione di colore se ne conserverà uno solo.

Tuttavia JPEG non usa un metodo fisso di compressione. E' possibile scegliere il grado di compressione che si desidera applicare a un'immagine, determinando in questo modo anche la qualità dell'immagine. Più si comprime un'immagine, più se ne riduce la qualità. JPEG può raggiungere rapporti di compressione molto alti, riducendo le dimensioni delle immagini di circa un centinaio di volte rispetto a quelle originali. Questo è

ROBERTO GRASSI • GIOVANNI PINTO • NICOLA SERRA

Sistemi per l'elaborazione dell'informazione

EdiSES
www.edises.it

ISBN 978–88–7959–904–7

€ 21,00