Teoria dell'Informazione e Crittografia

Ilaria Battiston

Anno scolastico 2018-2019

Contents

1 Codifica dei messaggi

La teoria dell'informazione (Claude E. Shannon) ha come principale obiettivo la trasmissione efficiente di messaggi tramite un canale di comunicazione. Dalla sorgente al canale avviene una codifica del messaggio in un formato prestabilito, che poi verrà decodificato prima di arrivare al ricevente. Nei canali è presente il **rumore**.

Gli elementi del sistema di comunicazione sono mittente, canale e ricevente. La trasmissione avviene nel tempo, e le informazioni viaggiano attraverso circuiti, cavi, campi elettromagnetici ecc.

La codifica (decodifica) si identifica come:

- **Codifica sorgente**, trasformazione di formato. Ha come obiettivo la rappresentazione efficiente (che occupa meno spazio) dei messaggi, quindi effettua la compressione dei dati;
- **Codifica di canale**, con eventuale correzione degli errori: aggiunge informazioni (ridondanza) in modo che il ricevente riesca a stabilire la correttezza del messaggio.

La memorizzazione dei dati è un'altra parte importante del processo, che visualizza il canale come una memoria su cui la sorgente può scrivere. Questa rappresentazione è utile perché anche su qualsiasi mezzo fisico di memorizzazione è presente rumore.

La teoria dell'informazione non riguarda il significato dei messaggi: le informazioni ottenute dalla trasmissione non riguardano il contenuto ma l'aspetto tecnico delle nozioni.
La teoria dei codici, invece, studia metodi per codificare i messaggi in modo compatto e rimediare al rumore.

1.1 Sorgenti

Le sorgenti considerate sono discrete: consistono in una serie di simboli $S = \{s_1, s_2, \dots, s_q\}$ con $|S| = q$, dei quali ciascuno viene prodotto in un colpo di clock. La generazione è sincrona (esattamente un simbolo alla volta) e casuale: se così non fosse ci sarebbe un processo deterministico che consente di prevedere l'output.

Il concetto chiave è che gli eventi rari permettono di avere una quantità maggiore di informazioni, quindi gli algoritmi deterministici sono poco utili.

Molte sorgenti sono memoryless, cioè i dati trasmessi nel passato non sono salvati. Un'eccezione riguarda per esempio il testo, in cui l'occorrenza di un carattere può essere più o meno probabile a seconda del precedente.

Si definisce codifica una funzione che prende in input un simbolo in S e produce una stringa, quindi $Cod \to S : \Gamma*$ dove $\Gamma = \{0, 1\}$. Si ha che $Cod(s_i) = w_i$, dove w_i si definisce codeword e $|w_i| = l_i$ (stringhe di lunghezza finita).

Ogni simbolo s_i ha associata una probabilità p_i, la cui distribuzione rispetta le seguenti condizioni (sorgente casuale):

- $0 < p_i < 1$;
- $\sum_{i=1}^{q} p_i = 1$.

La prima condizione ha la disuguaglianza stretta perché probabilità come 0 o 1 non permettono di ricavare informazioni.

La codifica sorgente ha come scopo la minimizzazione della quantità $L = \sum_{i=1}^{q} p_i l_i$, cioè la media del prodotto tra probabilità e lunghezza delle codeword. Questo risulta in una memorizzazione più compatta possibile.

1.2 Codifica di sorgente

La codifica a blocchi ha lunghezza dell'input uguale a quella dell'output, in contrapposizione alla codifica a lunghezza variabile. Quest'ultima è utile in caso di simboli molto probabili, per avere stringhe generalmente più corte.

Un esempio di codifica a lunghezza variabile è il codice Morse, che si basa su punti, linee e spazi ad ognuno dei quali viene associato un numero di unità di tempo la lettera E ha una probabilità del 12% e viene codificata con ·, mentre la Y con - · - -.

Esempi di codifiche di sorgente a lunghezza fissa (codeword più facili da riconoscere) sono 3-su-7 (Van Duuren) e 2-su-5.

Il codice di Van Duuren consiste nell'associazione di stringhe nelle quali esiste un numero prefissato di 1, disponibili in $\binom{7}{3}$ combinazioni. In altre parole, le parole hanno 7 bit di cui 3 sono 1. Se non ci sono 1 nelle posizioni previste è possibile individuare un errore.

$\binom{5}{2}$ funziona analogamente: un esempio pesato si basa sui valori 0-1-2-4-7 associati a ogni posizione: ogni numero naturale (fino a 10, essendoci 5 bit) è rappresentato come combinazione di somme, e lo 0 è un'eccezione indicata come $7 + 4$.
Esempio: $3 \to 01100$, cioè $2 + 1$.

1.3 Codifica di canale

La codifica di sorgente restituisce un output a cui vengono aggiunti bit per l'individuazione degli errori, la quale può avvenire a due livelli:

1. Error-detecting codes, che si limitano a riconoscere la presenza di errori;
2. Error-correcting codes, che li correggono.

1.3.1 Controllo di parità

Ogni stringa di n bit ha $n-1$ bit che compongono il messaggio e l'ultimo bit y come check ×: y può essere ricavato per esempio effettuando lo XOR tra i caratteri precedenti, quindi $x_1 \oplus x_2 \oplus \cdots \oplus x_{n-1}$.

Questa operazione è rappresentabile come il modulo 2 della somma di tutti i caratteri, tale che valga 0 se il numero di 1 è pari e viceversa.

$$\sum_{i=1}^{n-1} x_i \mod 2$$

Se il numero degli 1 è prestabilito pari e il check è 1, c'è un errore: questa metodologia è pertanto in grado di riconoscere errori, ma non correggerli. Il problema è che se il rumore altera 2 bit e il check è comunque pari, il messaggio ricevuto viene interpretato come valido quando non lo è.

Questa funzione è comunque utile quando la probabilità che più di un bit venga alterato è trascurabile, a seconda del modello. Il più semplice di essi è il **rumore bianco**.

Lo XOR viene utilizzato perché ha molti vantaggi: è già implementato in caso di messaggi che non superano il registro, è associativo quindi può essere calcolato dividendo la stringa, e nei circuiti elettronici è rappresentabile per mezzo di alberi o tramite simulazione di automi.

Con stringhe di n bit è possibile inviare $n-1$ bit, si ha che:

$$R = \frac{\text{n. totale di cifre trasmesse}}{\text{n. di cifre del messaggio}} = \frac{\text{msg + check}}{\text{msg}} \leq 1$$

Dove R sta per **ridondanza**. Questo valore è sempre minore o uguale a 1, e la quantità $\frac{\text{check}}{\text{msg}}$ viene definita **eccesso di ridondanza**, da minimizzare.

Per avere una ridondanza bassa bisogna utilizzare messaggi lunghi, ma ciò aumenta la probabilità di errori nella trasmissione: occorre trovare un giusto compromesso.

1.4 Rumore bianco

Il rumore bianco è un modello che descrive il comportamento del rumore all'interno del canale, ed è il più semplice in assoluto. Definisce la probabilità che avvengano k errori in un messaggio.

Con n bit $\in \{0,1\}^n$, c'è una probabilità p che il simbolo sia alterato: il rumore bianco si basa sull'assunzione che p sia uguale per ogni bit in ogni posizione. Per convenzione, $0 < p < 1/2$.

Le posizioni sono indipendenti: nel caso in cui le probabilità siano dipendenti dalla posizione, si parla di **burst**.

Queste condizioni permettono di rappresentare le stringhe in un sottospazio vettoriale isotropo.

La probabilità che avvenga un errore in una specifica posizione è $p(1-p)^{n-1}$. Applicando questa formula a n posizioni si ottiene $\binom{n}{1}p(1-p)^{n-1} = np(1-p)^{n-1} \approx np$.

In generale, la probabilità che si verifichino k errori su n bit è:

$$\binom{n}{k}p^k(1-p)^{n-k} \quad \forall\ k \in \{0,\dots,n\}$$

La probabilità che un controllo di parità fallisca (k pari con $k \neq 0$) è $\frac{1+(1-2p)^n}{2} - (1-p)^n$.

1.4.1 Burst

Un burst è la concentrazione di più errori in un breve intervallo di tempo: è necessario stimarne la lunghezza massima, detta L. Si suppone che L sia uguale alla lunghezza delle word, si può calcolare la parità direttamente sulle word.

Prima si inviano al destinario alcune word costituite solo da bit di messaggio, e poi una di bit di controllo. I controlli vengono effettuati sui bit di messaggio nella stessa posizione di ciascuna word, quindi ci sono L codici di parità indipendenti. Non possono verificarsi due errori nella stessa posizione coperta dallo stesso controllo.

Un'altra tecnica per prevenire i burst consiste nel codificare ogni singolo carattere con n bit e sommarli, per poi confrontare la somma dell'input con la somma dell'output. Tramite incroci è poi possibile capire dove è avvenuto l'errore.

1.5 Codici pesati

Nel caso di una trasmissione seriale (cifre trasmesse una di seguito all'altra), possono verificarsi errori come bit aggiunti, scambiati o mancanti. Per individuare queste tipologie, è possibile ricorrere ai codici pesati.

Questi codici assegnano un peso a ciascuna posizione del messaggio: con $m_1, m_2, \ldots, m_n$ cifre che compongono un messaggio, con c bit di controllo, si associano pesi alle $n+1$ posizioni:

2 Codici di Hamming

Il codice di Hamming è un codice ottimale che serve per correggere un errore nella trasmissione di un messaggio. Si basa sul modello del rumore bianco, ed è una soluzione algebrica.

Su n bit trasmessi, ce ne sono k che costituiscono il messaggio e m di controllo, con $n = k+m$. Ci sono 2^m configurazioni relative a $n+1$ eventi, quindi per coprire tutte le possibilità $2^m \geq n+1 = m+k+1$.

Il numero di bit si vuole minimizzare, quindi si cercano i minimi valori in grado di rispettare la disequazione: è necessario capire quante cifre di messaggio possono essere coperte dalle cifre di controllo. Dato che il codice è ottimale, esso userà il numero massimo di cifre di messaggio permesse.

Le configurazioni veramente ottimali sono quelle in cui la disuguaglianza è in realtà un'uguaglianza: $n = 2^m - 1$. Le cifre di controllo vanno quindi disposte in modo da coprire perfettamente tutti i bit del messaggio, ma in alcuni casi potrebbero esserci sovrapposizioni. Le equazioni di parità dovrebbero essere indipendenti tra loro.

Esempio in $\{0,1\}^9$:
$c_1 = 1, 2, 5, 7, c_2 = 57, 8, 9, c_3 = 1, 2, 8, 9$
I risultati vengono ottenuti tramite XOR, il quale permette di ottenere combinazioni lineari grazie alle somme modulo 2 e la selezione degli elementi con 1 (moltiplicazione per 1). Essendo c_3 composto da bit in posizioni già conosciuti, esso non dà informazioni aggiuntive: le equazioni devono quindi essere linearmente indipendenti.

Ogni configurazione dei bit di controllo (2^m) introduce una sindrome, a cui va aggiunto il valore 000 che indica l'assenza di errori. Lo scopo della sindrome è il riconoscimento della posizione dell'errore: per esempio, 110 rappresenta un errore nella posizione 6.

Le posizioni per calcolare le varie cifre di controllo sono quelli dove è presente un 1, considerando che le altre non cambiano. I bit delle posizioni con 1 per ogni cifra di controllo sono linearmente indipendenti.

Esempio: $k = 4, m = 3 \rightarrow n = 7$
Messaggio: $c_1c_2 1 c_3 011$
Il numero di 1 dei bit coperti da ogni c dev'essere pari, e ogni c copre al più k bit, quindi la codifica del messaggio sarà 0110011.
Se il messaggio viene alterato, l'equazione di parità non viene più rispettata e il bit compromesso viene individuato dalla sindrome.

Esempio: 0110111, si ha che c_1 ha 1 pari, c_2 anche, ma c_3 ne ha 3, quindi l'errore è nella posizione 5. Questo valore è stato ottenuto sommando i valori di c.

Le posizioni di c vengono scelte tra quelle che devono avere necessariamente 1 e che non hanno già un ruolo nel calcolo delle precedenti, oppure non hanno già bit coperti. In generale, i controlli si trovano nelle posizioni $2^0, 2^1, 2^2, 2^3, 2^4, \ldots$

Un vantaggio di questo metodo è che i calcoli possono essere eseguiti in parallelo, quindi più velocemente. Se la sindrome è 0 non ci sono stati errori: il numero di 1 nei bit di messaggio è pari, quindi $c_i = 0$.