Teoria dell'informazione e della trasmissione

Indice

1. Lezione 01

1.1. Introduzione

1.1.1. Storia

La **teoria dell'informazione** nasce nel 1948 grazie a **Claude Shannon** (1916-2001), un impiegato alla "Telecom" americana al quale sono stati commissionati due lavori: data una comunicazione su filo di rame, si voleva sfruttare tutta la capacità del canale, ma al tempo stesso correggere gli errori di trasmissione dovuti al rumore presente.

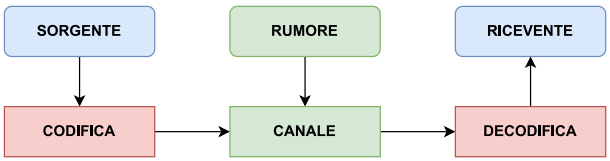
Nel luglio 1948 infatti viene pubblicato l'articolo "*A Mathematical Theory of Communication*" da parte di Bell Labs, dove Shannon pone le basi della teoria dell'informazione.

Ma non è l'unico personaggio che lavora in questo ambito: infatti, nel 1965, tre matematici russi pubblicano degli articoli che vanno a perfezionare il lavoro fatto anni prima da Shannon.

I tre matematici sono Gregory Chaitin (1947-), Ray Solomonoff (1926-2009) e **Andrey Kolmogorov** (1903-1987), ma si considerano solo gli articoli di quest'ultimo poiché ai tempi era molto più famoso dei primi due.

1.1.2. Shannon vs Kolmogorov

La situazione generica che troveremo in quasi la totalità dei nostri studi si può ridurre alla comunicazione tra due entità tramite un **canale affetto da rumore**.



Quello che distingue Shannon da Kolmogorov è l'approccio: il primo propone un **approccio ingegneristico**, ovvero tramite un modello formato da una distribuzione di probabilità si va a definire cosa fa *in media* la sorgente, mentre il secondo propone un **approccio rigoroso e formale**, dove sparisce la nozione di media e si introduce la nozione di sorgente in modo *puntuale*.

In poche parole, dato un messaggio da comprimere:

- Shannon direbbe "lo comprimo *in media* così, e lo comprimerei così anche se il messaggio fosse totalmente diverso";
- Kolmogorov direbbe "lo comprimo *esattamente* così, ma lo comprimerei in modo totalmente diverso se il messaggio fosse diverso".

1.1.3. Obiettivi di Shannon

Gli obiettivi che Shannon vuole perseguire sono due:

- **massimizzare** l'informazione trasmessa *ad ogni utilizzo del canale*;
- **minimizzare** il numero di errori di trasmissione dovuti alla presenza del rumore nel canale.

La parte "*ad ogni utilizzo del canale*" viene inserita per dire che, ogni volta che si accede al canale, deve essere utilizzato tutto, mentre senza questa parte una sorgente potrebbe mandare l'1% del messaggio ad ogni accesso al canale, mandandolo sì tutto ma senza sfruttare a pieno la banda.

Shannon risolverà questi due problemi con due importantissimi teoremi:

- *I°* **teorema di Shannon**, che riguarda la *source coding*, ovvero la ricerca di un codice per rappresentare i messaggi della sorgente che massimizzi l'informazione spedita sul canale, ovvero massimizzi la sua **compressione**;
- *II°* **teorema di Shannon**, che riguarda la *channel coding*, ovvero la ricerca di un codice per rappresentare i messaggi della sorgente che minimizzi gli errori di trasmissione dovuti alla presenza del rumore nel canale.

L'approccio che viene usato è quello *divide-et-impera*, che in questo caso riesce a funzionare bene e riesce ad unire i risultati dei due teoremi di Shannon grazie al **teorema di codifica congiunta sorgente-canale** e ad alcune relazioni che legano i due problemi descritti.

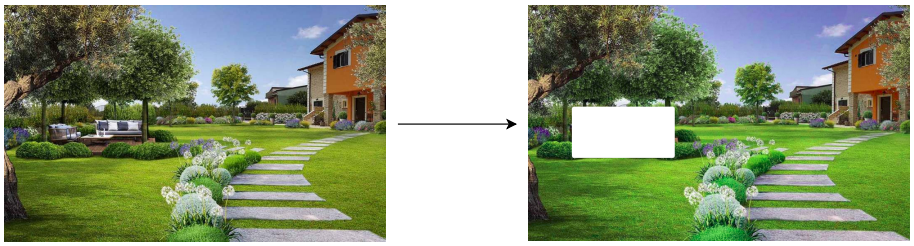
In un caso generale del *divide-et-impera* si ricade in una soluzione sub-ottimale.

1.1.4. Primo teorema di Shannon

Il primo problema da risolvere è il seguente: come è distribuita l'informazione all'interno di un documento?

Vediamo due esempi dove un documento viene spedito su un canale e alcune informazioni vengono perse per colpa del rumore presente nel canale.

In questo primo esempio notiamo che, nonostante l'informazione persa sia sostanziosa, possiamo in qualche modo "risalire" a quello perso per via delle informazioni che troviamo "nelle vicinanze".



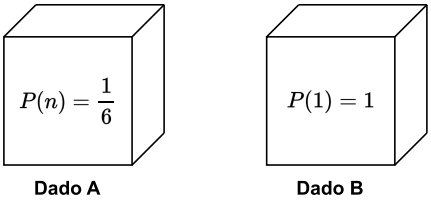
In questo secondo esempio notiamo invece che, nonostante l'informazione persa sia molto meno rispetto a quella precedente, "risalire" al contenuto perso è molto più difficile.

Questi due esempi dimostrano come l'informazione contenuta in un documento **non** è uniforme, e quindi che una distorsione maggiore non implica una perdita maggiore di informazioni.

L'obiettivo del primo teorema di Shannon è eliminare le informazioni inutili e ridondanti, comprimendo il messaggio per poter utilizzare il canale per inviare altre informazioni.

Quello che facciamo è concentrare l'informazione, rendendola **equamente distribuita**, quindi impossibile da ridurre ancora e contenente solo informazioni importanti.

Vediamo un altro esempio: supponiamo di avere due dadi a sei facce, uno *normale* e uno *truccato*, e supponiamo di tirarli assieme per un numero di volte molto grande. Quale dei due dadi mi dà più informazioni?



La risposta è quello *normale*: nel lungo il dado *normale* si "normalizzerà" su una probabilità di circa $\frac{1}{6}$ per ogni possibile faccia, quindi è una sorta di evento **regolare** che mi dà tanta informazione, mentre quello truccato posso sapere già cosa produrrà, quindi è una sorta di evento **prevedibile** che mi dà poca informazione.

In poche parole, massimizzo la probabilità di "ottenere" informazioni se le probabilità di estrarre un simbolo sono uguali.

1.1.5. Secondo teorema di Shannon

Il secondo teorema di Shannon è quello più rognoso, perché si occupa della *channel coding*, ovvero di una codifica che permetta di minimizzare l'informazione persa durante la trasmissione.

Vogliamo questo perché l'informazione che passa sul canale è compressa, quindi qualsiasi bit perso ci fa perdere molte informazioni, non essendoci ridondanza.

Quello che viene fatto quindi è aggiungere **ridondanza**, ovvero più copie delle informazioni da spedire così che, anche perdendo un bit di informazione, lo si possa recuperare usando una delle copie inviate.

La ridondanza che aggiungiamo però è **controllata**, ovvero in base al livello di distorsione del canale utilizzato si inviano un certo numero di copie.

In un **canale ideale** la ridondanza è pari a 0, mentre per canali con rumore viene usata una matrice stocastica, che rappresenta la distribuzione probabilistica degli errori.

IN/OUT	a	b	c	d	e
a	0.7	0.0	0.1	0.1	0.1
b	0.2	0.8	0.0	0.0	0.0
c	0.1	0.0	0.6	0.2	0.1
d	0.0	0.0	0.2	0.5	0.3
e	0.0	0.0	0.0	0.0	1.0

Ogni riga i rappresenta una distribuzione di probabilità che definisce la probabilità che, spedito il carattere i, si ottenga uno dei valori j presenti nelle colonne. Ovviamente, la somma dei valori su ogni riga è uguale a 1.

Se il canale è ideale la matrice risultante è la matrice identità.

1.2. Codifica della sorgente

Andiamo a modellare e formalizzare il primo problema con un modello statistico, utilizzando però un approccio *semplice e bello*: assumiamo che le regole di compressione non siano dipendenti dalle proprietà di un dato linguaggio.

Ad esempio, data la lettera "H" nella lingua italiana, ho più probabilità che esca la lettera "I" piuttosto che la lettera "Z" come successiva di "H", ma questa probabilità nel nostro modello non viene considerata.

Il codice *zip* invece prende in considerazione questo tipo di distribuzione statistica dipendente, e infatti è molto più complicato.

1.2.1. Introduzione matematica

Introduciamo una serie di "personaggi" che saranno utili nella nostra modellazione:

- insieme X: insieme finito di simboli che compongono i messaggi generati dalla sorgente;
- messaggio $\overline{x}$: sequenza di n simboli sorgente; in modo formale,

$$\overline{x} = (x_1, ..., x_n) \in \mathrm{X}^n, \text{ con } x_i \in \mathrm{X} \ \forall i \in \{1, ..., n\};$$

- base D;
- insieme $\{0, ..., D-1\}$: insieme finito dei simboli che compongono il codice scelto;
- insieme $\{0, ..., D-1\}^+$: insieme di tutte le possibili parole di codice esprimibili tramite una sequenza non vuota di simboli di codice; in modo formale,

$$\{0, ..., D-1\}^+ = \bigcup_{n=1}^{\infty} \{0, ..., D-1\}^n.$$

  Un altro nome per le parole di codice è sequenze D-arie;
- funzione c: funzione (nel nostro caso *codice*) che mappa ogni simbolo $x \in \mathrm{X}$ in una parola di codice, ovvero una funzione del tipo

$$c : \mathrm{X} \longrightarrow \{0, ..., D-1\}^+.$$

  Questa funzione va ad effettuare un **mapping indipendente**, ovvero tutto viene codificato assumendo che non esistano relazioni tra due o più estrazioni consecutive.

1.2.2. Prima applicazione

Vogliamo trasmettere sul canale i semi delle carte da poker utilizzando il codice binario.

Andiamo a definire:

- $\mathrm{X} = \{♥, ♦, ♣, ♠\}$ insieme dei simboli sorgente;
- $c : \mathrm{X} \longrightarrow \{0, 1\}^+$ funzione di codifica;
- $c(♥) = 0$;
- $c(♦) = 01$;
- $c(♣) = 010$;
- $c(♠) = 10$.

La codifica proposta è sicuramente plausibile, ma ha due punti deboli:

- *ambiguità*: se ricevo "010" come possibili traduzioni ho:
  - ♣;
  - ♥♠;
  - ♦♥;
- *pessima compressione*: usiamo tre simboli di codice per codificare ♣ e solo uno per codificare ♥.

Quest'ultimo punto debole viene risolto prima introducendo $l_c(x)$ come lunghezza della parola di codice associata ad $x \in \mathrm{X}$, e poi minimizzando la media di tutte le lunghezze al variare di $x \in \mathrm{X}$.

1.2.3. Modello statistico

Per completare il modello abbiamo bisogno di un'altra informazione: la **distribuzione di probabilità**, che definisce la probabilità con la quale i simboli sorgente sono emessi.

Definiamo quindi la sorgente come la coppia $\langle \mathrm{X}, p \rangle$, dove p rappresenta la probabilità prima descritta.

Aggiungiamo qualche "protagonista" a quelli già prima introdotti:

- funzione P_n: non siamo molto interessati ai singoli simboli sorgente, ma vogliamo lavorare con i messaggi, quindi definiamo

$$P_n(\overline{x}) = P_n(x_1, ..., x_n) = \prod_{i=1}^{n} p(x_i).$$

  Possiamo applicare la produttoria perché Shannon assume che ci sia *indipendenza* tra più estrazioni di simboli sorgente;
- variabile aleatoria $\mathbb{X}$: variabile aleatoria $\mathbb{X} : \mathrm{X} \longrightarrow \mathbb{R}$ che rappresenta un'estrazione di un simbolo sorgente;
- insieme $\mathbb{D}$: già definito in precedenza come $\{0, ..., D-1\}$;
- funzione c: già definita in precedenza, ma che riscriviamo come $c : \mathrm{X} \longrightarrow \mathbb{D}^+$.

Con questo modello, fissando X insieme dei simboli sorgente e D base, vogliamo trovare un codice $c : \mathrm{X} \longrightarrow \mathbb{D}^+$ che realizzi la migliore compressione, ovvero che vada a minimizzare il *valore atteso* della lunghezza delle parole di codice, definito come $\mathbb{E}[l_c] = \sum_{x \in \mathbb{X}} l_c(x)p(x)$.

La strategia che viene utilizzata è quella dell'alfabeto Morse: utilizziamo parole di codice corte per i simboli che sono generati spesso dalla sorgente, e parole di codice lunghe per i simboli che sono generatori raramente dalla sorgente.

Il primo problema che incontriamo è quello di evitare la *codifica banale*: se usassi il codice

$$c(x) = 0/1 \quad \forall x \in \mathrm{X}$$

avrei sì una codifica di lunghezza minima ma sarebbe impossibile da decodificare.

Dobbiamo imporre che il codice c sia **iniettivo**, o *non-singolare*.

2. Lezione 02

2.1. Codici

Come detto nella scorsa lezione, andiamo a considerare dei codici non singolari per risolvere la problematica dei due *codici banali*, che minimizzavano sì il valore atteso delle lunghezze delle parole di codice $l_c(x)$, ma rendevano impossibile la decodifica.

Andiamo ora a raffinare ulteriormente i codici singolari presi in considerazione.

2.1.1. Codice univocamente decodificabile

Come detto nella scorsa lezione, siamo interessati alle parole che vengono generate dalla mia sorgente, e non ai singoli caratteri.

Andiamo a definire:
- $\mathrm{X} = \{♥, ♦, ♣, ♠\}$ insieme dei simboli sorgente;
- $c : \mathrm{X} \longrightarrow \{0,1\}^+$ funzione di codifica;
- $c(♥) = 0$;
- $c(♦) = 01$;
- $c(♣) = 010$;
- $c(♠) = 10$.

La codifica c scelta è sicuramente non singolare, però abbiamo delle problematiche a livello di decodifica: infatti, possiamo scrivere 01001 come
- $c(♣, ♦)$;
- $c(♦, ♥, ♦)$;
- $c(♥, ♠, ♦)$

e lato ricevente questo è un problema, perché "unendo" le singole codifiche non otteniamo una parola che è generabile in modo unico.

Introduciamo quindi l'**estensione** di un codice c: essa è un codice $C : \mathrm{X}^+ \longrightarrow \mathbb{D}^+$ definito come $C(x_1, ..., x_n) = c(x_1)...c(x_n)$ che indica la sequenza ottenuta giustapponendo le parole di codice $c(x_1), ..., c(x_n)$.

L'estensione C di un codice c non eredita in automatico la proprietà di non singolarità di c.

Un codice c è **univocamente decodificabile** quando la sua estensione C è non singolare.

Tramite l'**algoritmo di Sardinas-Patterson** siamo in grado di stabilire se un codice è univocamente decodificabile in tempo $O(mL)$, dove $m = |\mathrm{X}|$ e $L = \sum_{x \in \mathrm{X}} l_c(x)$

2.1.2. Codice istantaneo

I codici univocamente decodificabili sono ottimali? Sto minimizzando al meglio $\mathbb{E}[l_c]$?

Prima di chiederci questo vogliamo creare un codice che rispetti un'altra importante proprietà: permetterci di decodificare *subito* quello che mi arriva dal canale senza dover aspettare tutto il flusso.

Andiamo a definire:
- $\mathrm{X} = \{♥, ♦, ♣, ♠\}$ insieme dei simboli sorgente;
- $c : \mathrm{X} \longrightarrow \{0,1\}^+$ funzione di codifica;
- $c(♥) = 10$;
- $c(♦) = 00$;
- $c(♣) = 11$;
- $c(♠) = 110$.

Supponiamo di spedire sul canale la stringa 11000…00, e poniamoci lato ricevente. In base al numero di 0 inviati abbiamo due possibili decodifiche:
- #0 pari: decodifichiamo con ♣♦…♦;
- #0 dispari: decodifichiamo con ♠♦…♦.

Nonostante si riesca perfettamente a decodificare, e questo è dato dal fatto che c è un codice univocamente decodificabile, dobbiamo aspettare di ricevere tutta la stringa per poterla poi decodificare, ma in ambiti come lo *streaming* questa attesa non è possibile.

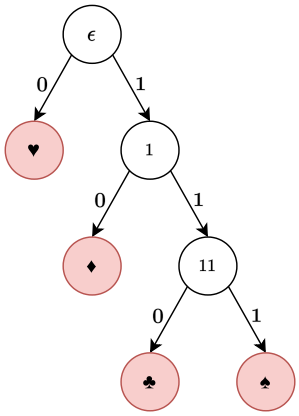
Un altro problema lo riscontriamo a livello di memoria: supponiamo che lato sorgente vengano codificati dei dati dell'ordine dei terabyte, per il ricevente sarà impossibile tenere in memoria una quantità simile di dati per fare la decodifica alla fine della ricezione.

Introduciamo quindi i **codici istantanei**, particolari codici che permettono di decodificare quello che arriva dal canale, appunto, in modo *istantaneo* senza aspettare.

Un codice si dice istantaneo se nessuna parola di codice è *prefissa* di altre.

Andiamo a definire:
- $\mathrm{X} = \{♥, ♦, ♣, ♠\}$ insieme dei simboli sorgente;
- $c : \mathrm{X} \longrightarrow \{0,1\}^+$ funzione di codifica;
- $c(♥) = 0$;
- $c(♦) = 10$;
- $c(♣) = 110$;
- $c(♠) = 111$.

Il seguente codice è istantaneo, e lo notiamo osservando l'*albero* che dà origine a questo codice.



Quello che otteniamo è un albero molto sbilanciato.

2.1.3. Gerarchia dei codici

Cerchiamo infine di definire una **gerarchia** tra i codici analizzati fin'ora: se siamo sicuri che i codici non singolari siano sotto-insieme di tutti i codici (*banale*), e che i codici univocamente decodificabili siano sotto-insieme dei codici non singolari (*banale*), siamo sicuri che i codici istantanei siano un sotto-insieme dei codici univocamente decodificabili?

Lemma Se c è istantaneo allora è anche univocamente decodificabile.

Dimostrazione
Andiamo a dimostrare che se c non è univocamente decodificabile allora non è istantaneo. Visto che c non è univocamente decodificabile (supponiamo sia almeno non singolare) esistono due messaggi distinti $x_1, x_2 \in \mathrm{X}^+$ tali che $C(x_1) = C(x_2)$.

Ci sono solo due modi per avere x_1 e x_2 distinti:
1. un messaggio è prefisso dell'altro: se x_1 è formato da x_2 e altri m caratteri, vuol dire che i restanti m caratteri di x_1 devono essere codificati con la parola vuota, che per definizione di codice non è possibile, e soprattutto la parola vuota sarebbe prefissa di ogni altra parola di codice, quindi il codice c non è istantaneo;
2. esiste almeno una posizione in cui i due messaggi differiscono: sia i la prima posizione dove i due messaggi differiscono, ovvero $x_1[i] \neq x_2[i]$ e $x_1[j] = x_2[j]$ per $1 \leq j \leq i-1$, ma allora $c(x_1[i]) \neq c(x_2[i])$ e $c(x_1[j]) = c(x_2[j])$ perché c è non singolare, quindi sto dicendo che x_1 deve avere x_2 come prefisso (o viceversa), ma, come al punto precedente, otteniamo che c non è istantaneo.

Quindi il codice c non è istantaneo. □

Abbiamo quindi stabilito una gerarchia di questo tipo:

$$\text{codici istantanei} \subset \text{codici univocamente decodificabili} \subset \text{codici non singolari} \subset \text{codici}\ .$$

Una cosa che possiamo notare è come, aumentando di volta in volta le proprietà di un codice, il valore atteso $\mathbb{E}[l_c]$ che vogliamo minimizzare peggiora, o al massimo rimane uguale: questo perché aggiungendo delle proprietà al nostro codice imponiamo dei limiti che aumentano in modo forzato la lunghezza delle nostre parole di codice.

2.2. Disuguaglianza di Kraft

2.2.1. Definizione

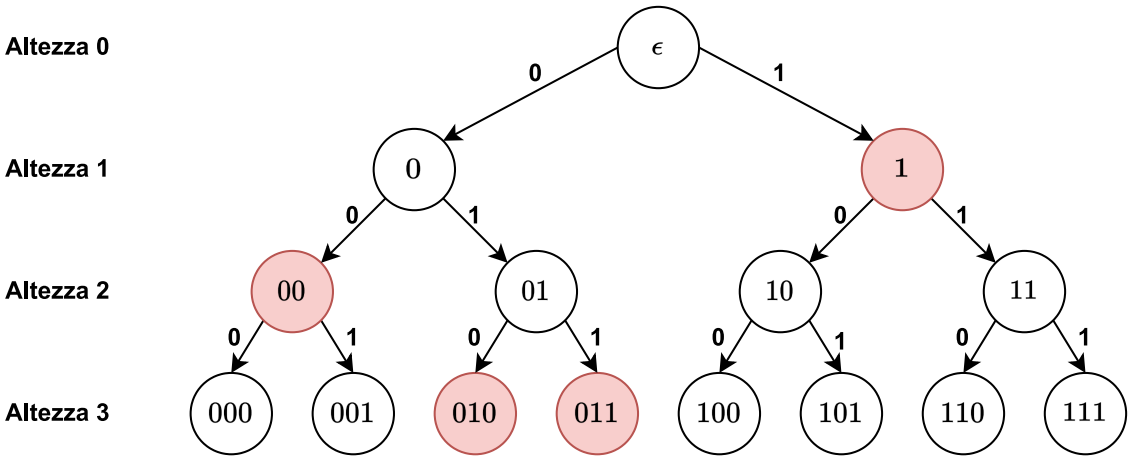
I codici istantanei soddisfano la **disuguaglianza di Kraft**, che ci permette, solo osservando le lunghezze delle parole di codice $l_c(x)$, di dire *se esiste* un codice istantaneo con quelle lunghezze.

Teorema *(Disuguaglianza di Kraft)* Dati $\mathrm{X} = \{x_1, ..., x_n\}$, $D > 1$ e n valori interi positivi $l_1, ..., l_n$, esiste un codice istantaneo $c : \mathrm{X} \to \mathbb{D}^+$ tale che $l_c(x_i) = l_i$ per $i = 1, ..., n$ se e solo se

$$\sum_{i=1}^{n} D^{-l_i} \leq 1.$$

Dimostrazione
($\Longrightarrow$) Sia $l_{\max}$ la lunghezza massima delle parole di c, ovvero $l_{\max} = \max_{i=1,...,n} (l_c(x_i))$. Si consideri l'albero D-ario completo di profondità $l_{\max}$ nel quale posizioniamo ogni parola di codice di c su un nodo dell'albero, seguendo dalla radice il cammino corrispondente ai simboli della parola. Dato che il codice è istantaneo, nessuna parola apparterrà al sotto-albero avente come radice un'altra parola di codice, altrimenti avremmo una parola di codice prefissa di un'altra. Andiamo ora a partizionare le foglie dell'albero in sottoinsiemi disgiunti $A_1, ..., A_n$, dove A_i indica il sottoinsieme di foglie associate alla radice contenente la parola $c(x_i)$.

Nel seguente esempio consideriamo un albero binario di altezza 3, e in rosso sono evidenziate le parole di codice 1, 00, 010, e 011.



Il numero massimo di foglie di un sotto-albero di altezza l_i è $D^{l_{\max}-l_i}$, ma il numero massimo di foglie nell'albero è $D^{l_{\max}}$, quindi

$$\underbrace{\sum_{i=1}^{n}|A_i| = \sum_{i=1}^{n} D^{l_{\max}-l_i}}_{\#\text{foglie coperte}} = \sum_{i=1}^{n} D^{l_{\max}} \cdot D^{-l_i} = D^{l_{\max}} \sum_{i=1}^{n} D^{-l_i} \leq D^{l_{\max}}.$$

Dividendo per $D^{l_{\max}}$ entrambi i membri otteniamo la disuguaglianza di Kraft.

($\Longleftarrow$) Assumiamo di avere n lunghezze positive $l_1, ..., l_n$ che soddisfano la disuguaglianza di Kraft e sia $l_{\max} = \max_{i=1,...,n}(l_i)$ la profondità dell'albero D-ario ordinato e completo usato prima. Associamo ad ogni simbolo $x_i \in \mathrm{X}$ la parola di codice $c(x_1)$, e la inseriamo al primo nodo di altezza l_i che troviamo in ordine lessicografico. Durante l'inserimento delle parole $c(x_i)$ dobbiamo escludere tutti i nodi che appartengono a sotto-alberi con radice una parola di codice già inserita o che includono un sotto-albero con radice una parola di codice già inserita. Il codice così costruito è istantaneo, e visto che rispetta la disuguaglianza di Kraft, la moltiplichiamo da entrambi i membri per $D^{l_{\max}}$ per ottenere

$$\sum_{i=1}^{n} D^{l_{\max}-l_i} \leq D^{l_{\max}},$$

ovvero il numero di foglie necessarie a creare il codice non eccede il numero di foglie disponibili nell'albero. □

2.2.2. Applicazione

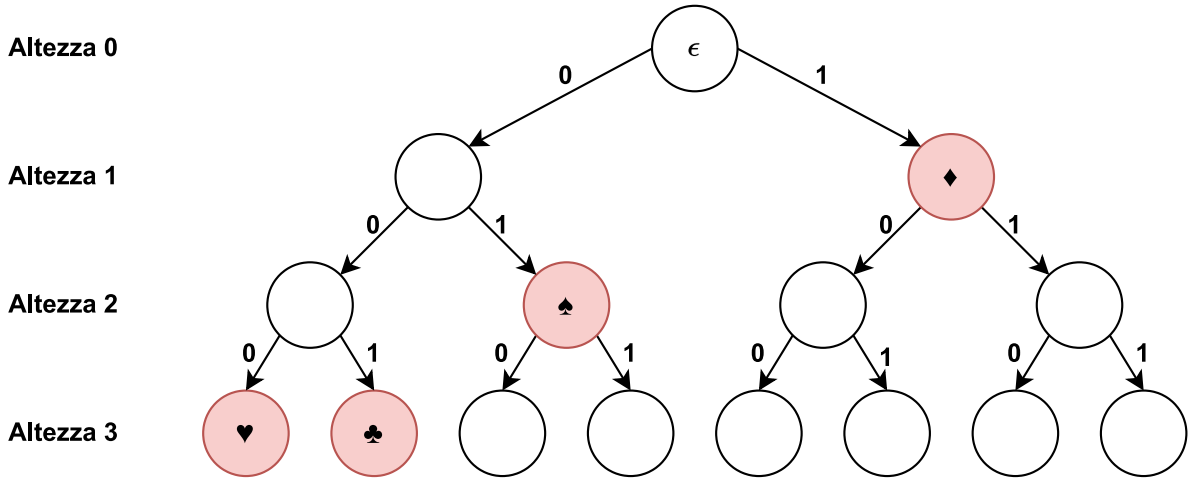
Andiamo a definire:

- $\mathrm{X} = \{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$ insieme dei simboli sorgente;
- $c : \mathrm{X} \longrightarrow \{0,1\}^+$ funzione di codifica;
- $l_c(\heartsuit) = 3$;
- $l_c(\diamondsuit) = 1$;
- $l_c(\clubsuit) = 3$;
- $l_c(\spadesuit) = 2$.

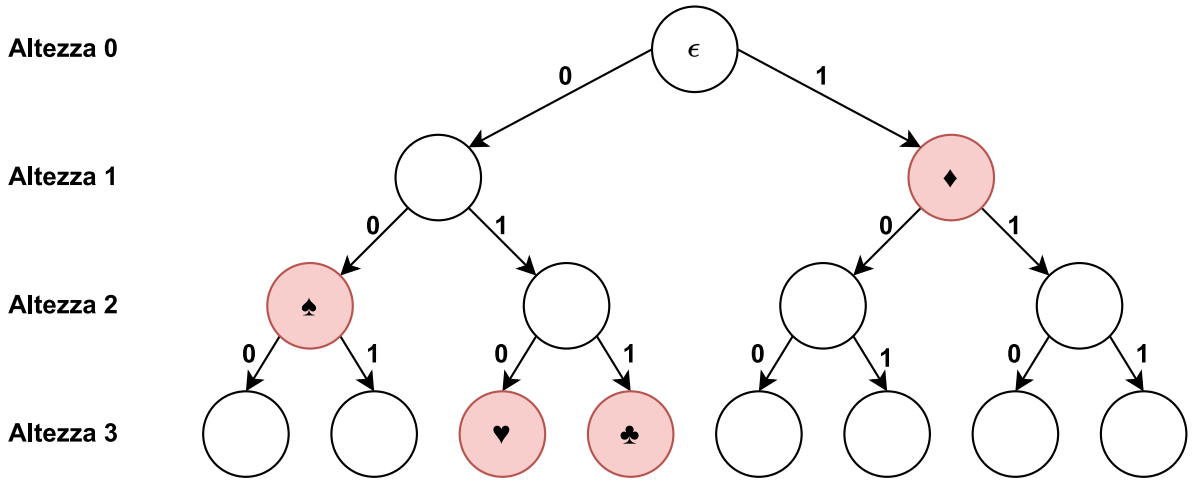
Vogliamo sapere se esiste un codice istantaneo, definito come c, aventi le lunghezze sopra definite. Controlliamo quindi se esse soddisfano la disuguaglianza di Kraft:

$$\sum_{x \in \mathrm{X}} 2^{-l_c(x)} = 2^{-3} + 2^{-1} + 2^{-3} + 2^{-2} = \frac{1}{8} + \frac{1}{2} + \frac{1}{8} + \frac{1}{4} = 1 \leq 1.$$

Andiamo a costruire quindi un codice istantaneo con queste lunghezze costruendo il suo albero.



Il codice ottenuto è l'unico possibile?



Come vediamo, "specchiando" il sotto-albero sinistro con radice ad altezza 1 otteniamo un codice istantaneo diverso dal precedente, ma comunque possibile.

Possiamo concludere quindi che, date n lunghezze positive $l_1, \ldots, l_n$ che soddisfano la disuguaglianza di Kraft, in generale *non è unico* il codice istantaneo che si può costruire.

3. Lezione 03

3.1. Codice di Shannon

3.1.1. Definizione

Abbiamo trovato un modo per verificare se un codice è istantaneo (osservare i prefissi) e un modo per verificare se n lunghezze positive possono essere usate come lunghezze di un codice istantaneo (disuguaglianza di Kraft), ma quale di questi codici istantanei è il migliore possibile?

Andiamo a vedere un modo per **costruire** un codice istantaneo a partire dai simboli sorgente e dalle loro probabilità di essere estratti dalla sorgente.

Dati il modello $\langle \mathrm{X}, p \rangle$ e $D > 1$, vogliamo trovare n lunghezze positive $l_1, ..., l_n$ per costruire un codice istantaneo con le lunghezze appena trovate minimizzando $\mathbb{E}[l_c]$, ovvero:

$$\begin{cases} \underset{l_1,...,l_n}{\text{minimize}} \sum_{i=1}^{n} l_i p_i \\ \text{tale che} \sum_{i=1}^{n} D^{-l_i} \leq 1 \end{cases}.$$

Quello che vogliamo fare è cercare n lunghezze positive $l_1, ..., l_n$ che minimizzino il valore atteso della lunghezza delle parole di codice e che soddisfino la disuguaglianza di Kraft.

Abbiamo a disposizione:
- $\mathrm{X} = \{x_1, ..., x_n\}$ insieme dei simboli sorgente, con $|\mathrm{X}| = n$;
- $P = \{p_i, ..., p_n\}$ insieme delle probabilità, con $p_i = p(x_i)$ e $\sum_{i=1}^{n} p_i = 1$.

Vogliamo trovare $L = \{l_1, ..., l_n\}$ insieme delle lunghezze delle parole di codice.

Andiamo ad unire la disuguaglianza di Kraft con la definizione di probabilità: infatti, sapendo che $\sum_{i=1}^{n} p_i = 1$, andiamo a sostituire questa sommatoria al posto del valore 1 all'interno della disuguaglianza di Kraft, ottenendo

$$\sum_{i=1}^{m} D^{-l_i} \leq \sum_{i=1}^{m} p_i = 1.$$

Sicuramente questa disuguaglianza vale se imponiamo

$$D^{-l_i} \leq p_i \quad \forall i = 1, ..., n$$

ma allora

$$D^{l_i} \cdot D^{-l_i} \leq p_i \cdot D^{l_i} \implies D^{l_i} \geq \frac{1}{p_i} \implies l_i \geq \log_D \frac{1}{p_i}.$$

Non sempre però il logaritmo mi rappresenta delle quantità intere, quindi andiamo ad arrotondare per eccesso questo conto:

$$l_i = \left\lceil \log_D \frac{1}{p_i} \right\rceil.$$

Abbiamo così trovato le lunghezze del mio codice istantaneo, legate in modo stretto alla probabilità di estrarre un simbolo.

Il codice così trovato viene detto **codice di Shannon**, o *codice di Shannon-Fano*, e siamo sicuri che è un codice "che fa bene": infatti, usa tante parole di codice per simboli che vengono estratti raramente, e poche parole di codice per simboli che vengono estratti frequentemente.

Questa proprietà viene dal fatto che il logaritmo, essendo una funzione monotona crescente, produce:
- valori "grandi" quando viene calcolato con numeri "grandi", quindi quando $\frac{1}{p_i}$ è "grande" e quindi p_i è "piccolo";
- valori "piccoli" quando viene calcolato con numeri "piccoli", quindi quando $\frac{1}{p_i}$ è "piccolo" e quindi p_i è "grande".

3.1.2. Esempi

3.1.2.1. Base / migliore

Supponiamo che la sorgente S emetta $n = 4$ simboli con probabilità $P = \left\{\frac{1}{2}, \frac{1}{4}.\frac{1}{8}, \frac{1}{8}\right\}$, vogliamo costruire un codice binario istantaneo.

Calcoliamo le lunghezze con l'algoritmo proposto da Shannon:

- $l_1 = \left\lceil \log_2 \frac{1}{\frac{1}{2}} \right\rceil = \lceil \log_2 2 \rceil = 1$;
- $l_2 = \left\lceil \log_2 \frac{1}{\frac{1}{4}} \right\rceil = \lceil \log_2 4 \rceil = 2$;
- $l_{3,4} = \left\lceil \log_2 \frac{1}{\frac{1}{8}} \right\rceil = \lceil \log_2 8 \rceil = 3$.

Calcoliamo il valore atteso come $\mathbb{E}[l_c] = \sum_{i=1}^{4} l_i p_i = 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + 3 \cdot \frac{1}{8} = \frac{7}{4}$.

Il valore che abbiamo trovato è il migliore possibile?

La risposta è sì, e questo vale perché tutte le probabilità sono *potenze negative* della base D scelta, nel nostro caso $D = 2$.

Possiamo affermare questo perché le lunghezze l_i saranno esattamente uguali a $\log_D \frac{1}{p_i}$ senza eseguire nessuna approssimazione.

3.1.2.2. Degenere

Supponiamo che la sorgente S emetta $n = 4$ simboli con probabilità $P = \{1, 0, 0, 0\}$, vogliamo costruire un codice binario istantaneo.

Calcoliamo le lunghezze con l'algoritmo proposto da Shannon:

- $l_1 = \left\lceil \log_2 \frac{1}{1} \right\rceil = \lceil \log_2 1 \rceil = 0$;
- $l_{2,3,4} = \left\lceil \lim_{t \to 0^+} \log_2 \frac{1}{t} \right\rceil = \lceil \lim_{t \to 0^+} \log_2 +\infty \rceil = +\infty$.

Calcoliamo il valore atteso come $\mathbb{E}[l_c] = \sum_{i=1}^{4} l_i p_i = 0 \cdot 1 + \underbrace{0 \cdot (+\infty) + 0 \cdot (+\infty) + 0 \cdot (+\infty)}_{0 \text{ per } t \to 0^+} = 0$.

3.1.2.3. Equiprobabile

Supponiamo che la sorgente S emetta $n = 4$ simboli con probabilità $P = \left\{\frac{1}{n}, \frac{1}{n}, \frac{1}{n}, \frac{1}{n}\right\}$, vogliamo costruire un codice binario istantaneo.

Calcoliamo le lunghezze con l'algoritmo proposto da Shannon:

- $l_{1,2,3,4} = \left\lceil \log_2 \frac{1}{\frac{1}{4}} \right\rceil = \lceil \log_2 4 \rceil = 2$.

Calcoliamo il valore atteso come $\mathbb{E}[l_c] = \sum_{i=1}^{4} l_i p_i = 2 \cdot \frac{1}{4} + 2 \cdot \frac{1}{4} + 2 \cdot \frac{1}{4} + 2 \cdot \frac{1}{4} = 2$.

3.1.3. Entropia

Nel codice di Shannon andiamo a definire $l_i = \left\lceil \log_D \frac{1}{p_i} \right\rceil$, quindi sostituiamo l_i nel valore atteso che stiamo cercando di minimizzare, ottenendo $\mathbb{E}[l_c] = \sum_{i=1}^{n} p_i \log_D \frac{1}{p_i}$.

La quantità che abbiamo appena scritto si chiama **entropia**, e la usiamo perché è in stretta relazione con la codifica ottimale che possiamo realizzare. In particolare, l'entropia è il *limite inferiore* alla compattezza del codice.

Tutti i valori attesi che abbiamo calcolato negli esempi precedenti sono anche le entropie delle varie sorgenti, ma che valori assume questa entropia?

Sicuramente è una quantità *positiva o nulla*, visto la somma di prodotti di fattori *positivi o nulli*.

Inoltre, raggiunge il proprio massimo quando la distribuzione di probabilità è **uniforme**, e vale

$$\sum_{i=1}^{n} p_i \log_D \frac{1}{p_i} = \sum_{i=1}^{n} \frac{1}{m} \log_D m = \cancel{m} \cdot \frac{1}{\cancel{m}} \cdot \log_D m = \log_D m.$$

In questo caso otteniamo un *albero perfettamente bilanciato* con le codifiche a livello delle ultime foglie.

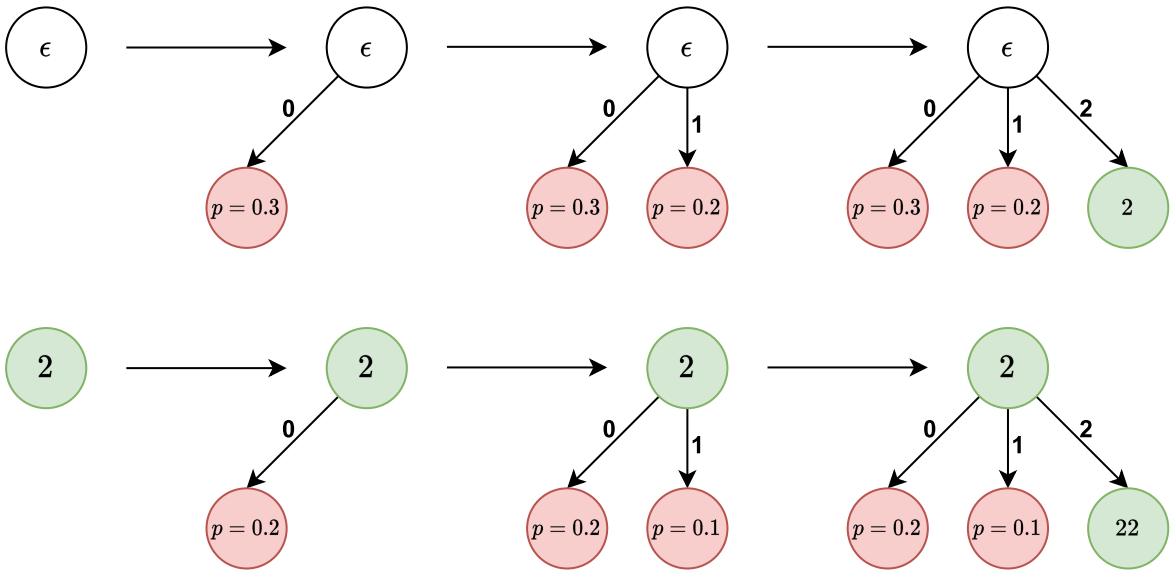
Questo mi va a dire che il codice è *compatto* e bilanciato, non spreco bit, mentre in altre situazioni ho un albero *sbilanciato* e potrei perdere dei bit.

3.2. Codice di Huffman

Supponiamo che la sorgente S emetta $n = 7$ simboli con probabilità $P = \{0.3, 0.2, 0.2, 0.1, 0.1, 0.06, 0.04\}$. Vogliamo costruire un codice ternario istantaneo.

Qui abbiamo due diversi approcci:

- codice di Shannon: otteniamo come lunghezze $2, 2, 2, 3, 3, 3, 3$;
- *grafico*: dato un albero ternario, associamo ai nodi più in alto le parole di codice dei simboli con probabilità maggiore.



Nell'immagine vediamo come prima inseriamo i simboli con probabilità 0.3 e 0.2, poi come terzo nodo mettiamo un "checkpoint" e iniziamo la procedura di nuovo da quest'ultimo nodo.

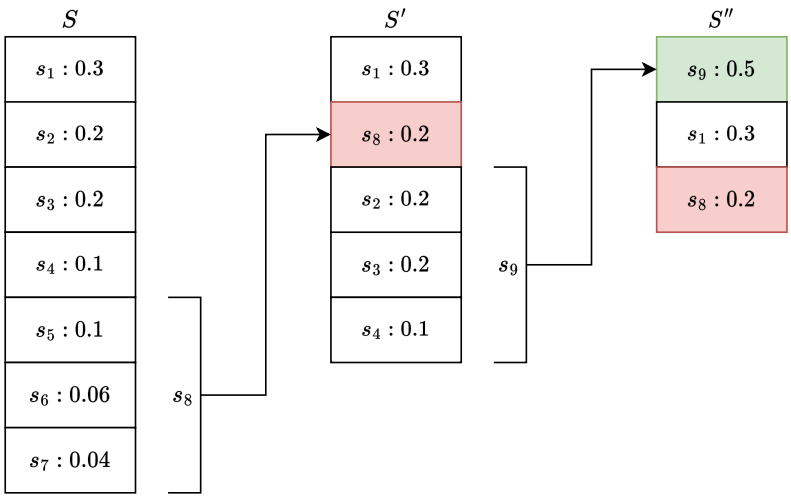
3.2.1. Definizione

Abbiamo in realtà un terzo approccio al problema precedente, ideato da **David Huffman** nel 1953, che lavora nel seguente modo:

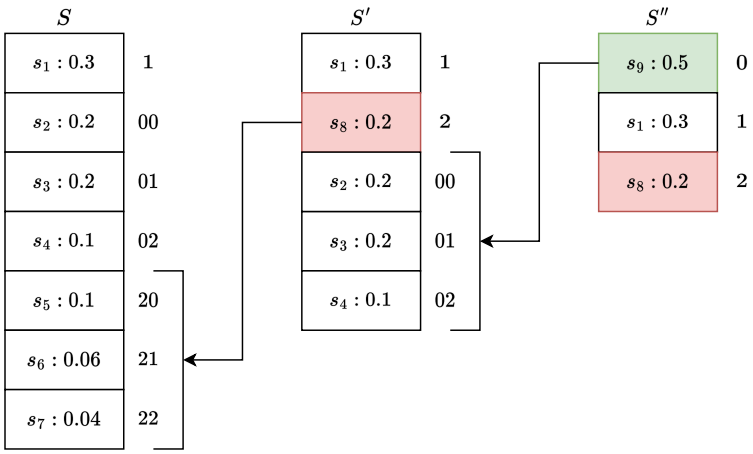
1. ordino le probabilità in ordine decrescente;
2. le ultime D probabilità sono sostituite dalla loro somma, e i simboli corrispondenti sono sostituiti da un simbolo "fantoccio", creando una nuova sorgente "fantoccia";
3. ripeto dal punto 1 fino a quando non si raggiungono t probabilità, con $t \leq D$;
4. scrivo il codice di Huffman facendo un "rollback" alla sorgente iniziale.

Il codice così creato è detto **codice di Huffman**, ed è il *codice istantaneo ottimo* che possiamo costruire.

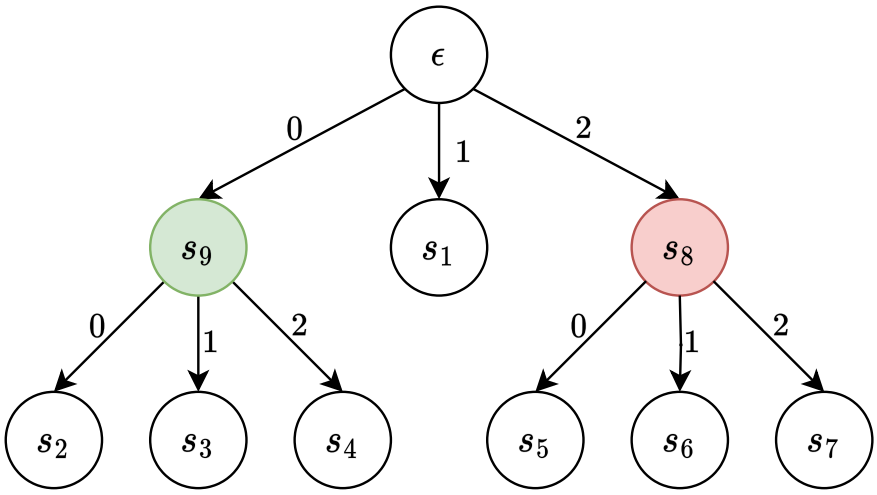
Supponiamo che la sorgente S emetta $n = 7$ simboli $s_1, ..., s_7$ con probabilità $P = \{0.3, 0.2, 0.2, 0.1, 0.1, 0.06, 0.04\}$, vogliamo costruire un codice ternario di Huffman.



In questa prima fase andiamo a "compattare" le probabilità minori fino ad avere una situazione con esattamente $D = 3$ simboli.

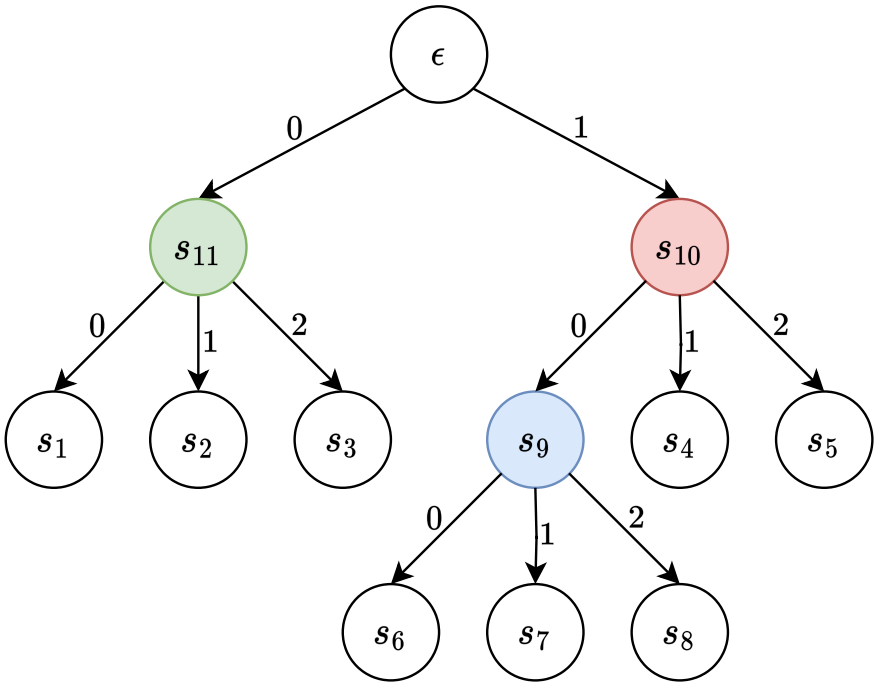


Nella seconda fase invece andiamo ad eseguire un "rollback" delle compressioni, sostituendo ad ogni nodo "compresso" le vecchie probabilità, andando quindi a costruire l'*albero* di codifica.



Supponiamo ora che la sorgente S emetta $n = 8$ simboli $s_1, ..., s_7, s_8$ con probabilità $P = \{0.3, 0.2, 0.2, 0.1, 0.1, 0.06, 0.02, 0.02\}$, vogliamo costruire un codice ternario di Huffman.

Considerando che ad ogni iterazione perdiamo $D = 3$ simboli e ne aggiungiamo uno, in totale perdiamo $D - 1$ simboli. Considerando che l'algoritmo genera il codice istantaneo ottimo quando termina con esattamente D probabilità, in questo esempio alla fine delle iterazioni ci troveremmo con solo due simboli sorgente, e non tre, quindi la codifica non risulta ottimale.



Infatti, notiamo come alla radice perdiamo un ramo, andando ad aumentare di conseguenza l'altezza dell'albero.

La soluzione proposta da Huffman consiste nell'inserire un numero arbitrario di simboli "fantoccio" con probabilità nulla, così da permettere poi un'ottima "compressione" fino ad avere D simboli.

Ma quanti simboli nuovi dobbiamo inserire?

Supponiamo di partire da n simboli e rimuoviamo ogni volta $D-1$ simboli:

$$n \longrightarrow n-(D-1) \longrightarrow n-2\cdot(D-1) \longrightarrow \ldots \longrightarrow n-t\cdot(D-1).$$

Chiamiamo $\blacksquare = n-t\cdot(D-1)$. Siamo arrivati ad avere $\blacksquare$ elementi, con $\blacksquare \leq D$, ma noi vogliamo esattamente D elementi per avere un albero ben bilanciato e senza perdita di rami, quindi aggiungiamo a $\blacksquare$ un numero k di elementi tali per cui $\blacksquare + k = D$. Supponiamo di eseguire ancora un passo dell'algoritmo, quindi da $\blacksquare + k$ andiamo a togliere $D-1$ elementi, lasciando la sorgente con un solo elemento.

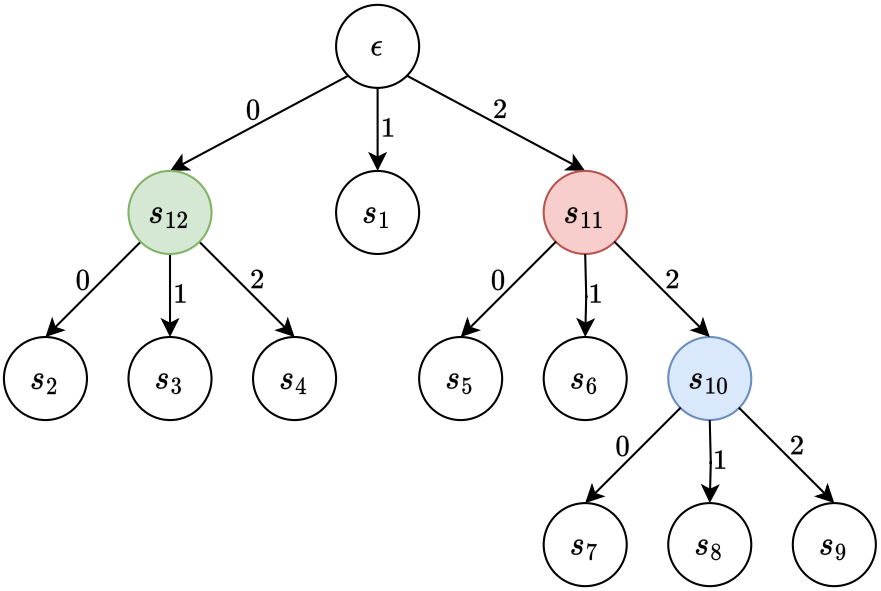
Cosa abbiamo ottenuto? Ricordando che $\blacksquare = n-t\cdot(D-1)$, abbiamo fatto vedere che:

$$\begin{aligned} \blacksquare + k - (D-1) &= 1 \\ n - t\cdot(D-1) + k - (D-1) &= 1 \\ n + k - (t+1)\cdot(D-1) &= 1. \end{aligned}$$

In poche parole, il numero n di simboli sorgente, aggiunto al numero k di simboli "fantoccio", è congruo ad 1 modulo $D-1$, ovvero

$$n + k \equiv 1 \bmod D-1.$$

Ripetiamo l'esempio precedente, aggiungendo il simbolo s_9 ad S con probabilità 0.



Con che ordine vado a inserire i simboli "compressi" dentro la lista delle probabilità? In modo *random*, quindi non è detto che il codice generato sia unico e ottimo.

4. Lezione 04

4.1. Entropia

4.1.1. Introduzione

La scorsa lezione abbiamo introdotto il concetto di **entropia**, ovvero la quantità

$$H_{D(\mathrm{X})} = \sum_{i=1}^{m} p_i \log_D \left(\frac{1}{p_i} \right).$$

L'entropia dipende da D: se questo vale 2 l'entropia calcolata è quella **binaria** e non ha il pedice vicino alla H.

L'entropia fa solo riferimento alla distribuzione di probabilità, non c'entra niente con i simboli: come vediamo, nella definizione compare solo D assieme alle probabilità dei singoli simboli.

4.1.2. Cambio di base

Sappiamo che con i logaritmi possiamo cambiare base. *E se volessi fare un cambio base nell'entropia?* Quello che sto facendo è cambiare l'entropia binaria in una che magari mi risulta più comoda.

Ricordando che

$$\log_b p = \frac{\log_c p}{\log_c b},$$

se fissiamo $c = e$ otteniamo

$$\log_b p = \frac{\ln p}{\ln b} = \frac{\ln p}{\ln b} \cdot \frac{\ln a}{\ln a} = \frac{\ln p}{\ln a} \cdot \frac{\ln a}{\ln b} = \log_a p \cdot \log_b a.$$

In poche parole, *cambiare base* significa *cambiare base e portarsi dietro una costante.*

Modifichiamo quindi l'entropia, ottenendo

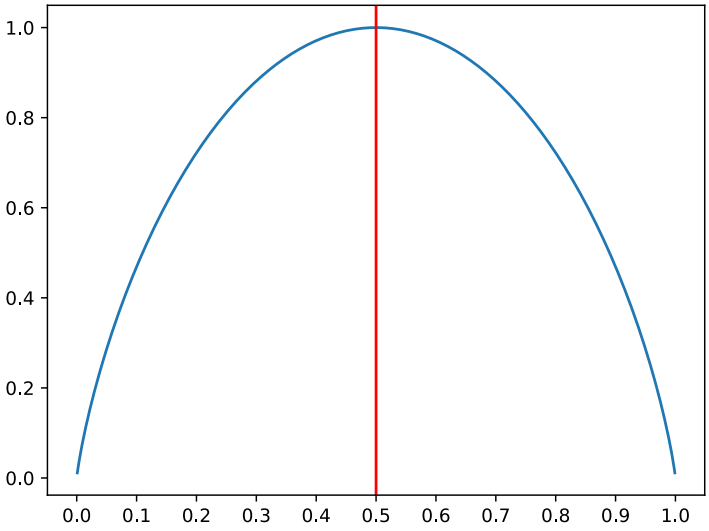
$$H_b(\mathrm{X}) = \sum_{i=1}^{m} p_i \log_b \frac{1}{p_i} \longrightarrow H_a(\mathrm{X}) = \sum_{i=1}^{m} p_i \log_a \frac{1}{p_i} \log_b a = \log_b a \cdot H_a(\mathrm{X}).$$

4.1.3. Esempio

Disegniamo l'entropia binaria usando $\mathbb{X}$ variabile aleatoria bernoulliana.

Sia quindi $P(\mathbb{X} = 1) = p$ e $P(\mathbb{X} = 0) = 1 - p$. Allora vale

$$H(\mathrm{X}) = p \log_2 \frac{1}{p} + (1 - p) \log_2 \frac{1}{1 - p}.$$

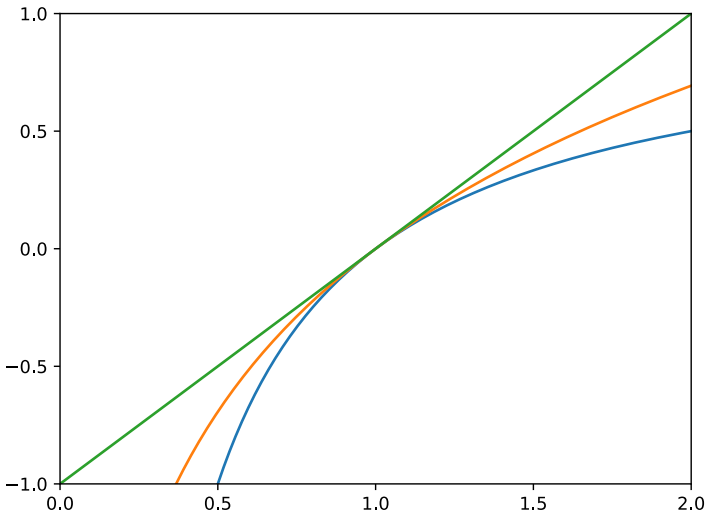


Notiamo come in $p = 0/1$ l'entropia vale $H(0/1) = 0$, mentre è massima in $p = \frac{1}{2}$ e vale $H\left(\frac{1}{2}\right) = 1$.

Infine, introduciamo la relazione

$$1 - \frac{1}{x} \leq \ln x \leq x - 1$$

che ci sarà utile dopo per alcune dimostrazioni.



4.1.4. Proprietà

Diamo prima di tutto un upper bound all'entropia.

Lemma $H_D(\mathrm{X}) \leq \log_D m$.

Dimostrazione

Dimostriamo che $H_D(\mathrm{X}) - \log_D m \leq 0$.

$$H_D(\mathrm{X}) - \log_D m = \sum_{i=1}^{m} p_i \log_D \frac{1}{p_i} - \log_D m \cdot \underbrace{\sum_{i=1}^{m} p_i}_{=1} =$$

$$= \sum_{i=1}^{m} p_i \log_D \frac{1}{p_i} - p_i \log_D m = \sum_{i=1}^{m} p_i \cdot \left(\log_D \frac{1}{p_i} - \log_D m \right) =$$

$$= \sum_{i=1}^{m} p_i \cdot \log_D \frac{1}{p_i \cdot m} \quad .$$

Ricordiamo che $\ln x \leq x - 1$, quindi:

$$\sum_{i=1}^{m} p_i \cdot \ln \underbrace{\frac{1}{p_i \cdot m}}_{x} \cdot \frac{1}{\ln D} \leq \sum_{i=1}^{m} p_i \left(\frac{1}{p_i \cdot m} - 1 \right) \frac{1}{\ln D}$$

$$\leq \frac{1}{\ln D} \sum_{i=1}^{m} \frac{1}{m} - p_i = \frac{1}{\ln D} \underbrace{\left[\underbrace{\sum_{i=1}^{m} \frac{1}{m}}_{1} - \underbrace{\sum_{i=1}^{m} p_i}_{1} \right]}_{0} = 0.$$

Se $H_D(\mathrm{X}) - \log_D m \leq 0$ allora

$$H_D(\mathrm{X}) \leq \log_D m.$$

□

Corollario Se $P(X = a_i) = \frac{1}{m} \quad \forall i = 1, ..., m$ allora

$$H_D(\mathrm{X}) = \log_D m.$$

Dimostrazione
Se $P(X = a_i) = \frac{1}{m} \quad \forall i = 1, ..., m$ allora

$$H_D(\mathrm{X}) = \sum_{i=1}^{m} p_i \log_D \frac{1}{p_i} = \sum_{i=1}^{m} \frac{1}{m} \log_D m = \log_D m \underbrace{\sum_{i=1}^{m} \frac{1}{m}}_{1} = \log_D m.$$

□

Introduciamo ora l'**entropia relativa**: indicata con $\Delta(\mathrm{X} \parallel Y)$ misura la distanza tra X e Y, ovvero la diversità tra X e Y in termini delle due distribuzioni di probabilità.

Definiamo quindi l'entropia relativa come

$$\Delta(\mathrm{X} \parallel Y) = \sum_{s \in S} p_X(s) \log_D \frac{p_{\mathrm{X}}(s)}{p_Y(s)}.$$

L'insieme S indica il **dominio** sul quale X e Y lavorano.

Teorema Per ogni coppia di variabili casuali X, Y definite sullo stesso dominio S vale la disuguaglianza

$$\Delta(\mathrm{X} \parallel Y) \geq 0.$$

Dimostrazione

$$\begin{aligned}
\Delta(\mathrm{X} \parallel Y) &= \sum_{s \in S} p_{\mathrm{X}}(s) \log_D \frac{p_{\mathrm{X}}(s)}{p_Y(s)} = \sum_{s \in S} p_{\mathrm{X}}(s) \ln \frac{p_{\mathrm{X}}(s)}{p_Y(s)} \frac{1}{\ln D} = \\
&= \frac{1}{\ln D} \sum_{s \in S} p_{\mathrm{X}}(s) \ln \underbrace{\frac{p_{\mathrm{X}}(s)}{p_Y(s)}}_{x} = \text{uso } 1 - \frac{1}{x} \leq \ln x = \\
&\geq \frac{1}{\ln D} \sum_{s \in S} p_{\mathrm{X}}(s) \left(1 - \frac{p_Y(s)}{p_{\mathrm{X}}(s)}\right) = \frac{1}{\ln D} \sum_{s \in S} p_{\mathrm{X}}(s) - p_Y(s) \\
&\geq \frac{1}{\ln D} \left(\underbrace{\sum_{s \in S} p_{\mathrm{X}}(s)}_{1} - \underbrace{\sum_{s \in S} p_Y(S)}_{1} \right) = 0.
\end{aligned}$$

□

Infine, vediamo la relazione tra *valore atteso delle lunghezze del codice* e *entropia.*

Teorema Sia $c : \mathrm{X} \to \mathbb{D}^+$ un codice istantaneo D-ario per una sorgente $\langle \mathrm{X}, p \rangle$, allora

$$\mathbb{E}[l_c] \geq H_D(\mathrm{X}).$$

Dimostrazione
Definisco $\mathbb{Z} : \mathrm{X} \longrightarrow \mathbb{R}$ variabile casuale alla quale associamo una distribuzione di probabilità

$$q(x) = \frac{D^{-l_c(x)}}{\sum_{x' \in \mathrm{X}} D^{-l_c(x')}}.$$

Dimostriamo che $\mathbb{E}[l_c] - H_D(\mathrm{X}) \geq 0$.

$$
\begin{aligned}
\mathbb{E}[l_c] - H_D(\mathrm{X}) &= \sum_{x\in\mathrm{X}} p(x) l_c(x) - \sum_{x\in\mathrm{X}} p(x)\log_D \frac{1}{p(x)} = \sum_{x\in\mathrm{X}} p(x)\cdot\left(l_c(x) - \log_D \frac{1}{p(x)}\right) = \\
&= \sum_{x\in\mathrm{X}} p(x)\cdot\left(\log_D D^{l_c(x)} - \log_D \frac{1}{p(x)}\right) = \\
&= \sum_{x\in\mathrm{X}} p(x)\cdot\left(\log_D D^{l_c(x)} + \log_D p(x)\right) = \sum_{x\in\mathrm{X}} p(x)\log_D\left(D^{l_c(x)}\cdot p(x)\right) = \\
&= \sum_{x\in\mathrm{X}} p(x)\cdot\left(\log_D\left(\frac{p(x)}{D^{-l_c(x)}}\cdot 1\right)\right) = \\
&= \sum_{x\in X} p(x)\cdot\left(\log_D\left(\frac{p(x)}{D^{-l_c(x)}}\cdot\frac{\sum_{x'\in\mathrm{X}} D^{-l_c(x')}}{\sum_{x'\in\mathrm{X}} D^{-l_c(x')}}\right)\right) = \\
&= \sum_{x\in\mathrm{X}} p(x)\cdot\left(\log_D\left(p(x)\frac{\sum_{x'\in\mathrm{X}} D^{-l_c(x')}}{D^{-l_c(x)}}\right) - \log_D\left(\sum_{x'\in\mathrm{X}} D^{-l_c(x')}\right)\right) = \\
&= \sum_{x\in\mathrm{X}} p(x)\cdot\left(\log_D\frac{p(x)}{q(x)} - \log_D\left(\sum_{x'\in\mathrm{X}} D^{-l_c(x')}\right)\right) = \\
&= \sum_{x\in\mathrm{X}} p(x)\log_D\frac{p(x)}{q(x)} - p(x)\log_D\left(\sum_{x'\in\mathrm{X}} D^{-l_c(x')}\right) = \\
&= \sum_{x\in\mathrm{X}} p(x)\log_D\frac{p(x)}{q(x)} - \sum_{x\in\mathrm{X}} p(x)\log_D\left(\sum_{x'\in\mathrm{X}} D^{-l_c(x')}\right) = \\
&= \underbrace{\Delta(\mathrm{X}\parallel\mathbb{Z})}_{\geq 0} \underbrace{\underbrace{-\log_D\left(\sum_{x'\in\mathrm{X}} D^{-l_c(x')}\right)}_{c \text{ istantaneo} \longrightarrow \log_D(t\leq 1)\leq 0}\cdot\underbrace{\sum_{x\in\mathrm{X}} p(x)}_{=1}}_{\geq 0} \geq 0.
\end{aligned}
$$

□

4.2. Sardinas-Patterson

L'**algoritmo di Sardinas-Patterson** è un algoritmo che permette di dimostrare se un codice è univocamente decodificabile.

4.2.1. Definizione

Dato $A = \{x_1, \ldots, x_n\}$ insieme dei simboli usati da un codice c, costruiamo l'insieme $S_1 = A$. Per stabilire se c è univocamente decodificabile generiamo una serie di insiemi S_i applicando le due regole seguenti:

- $\forall x \in S_1 \quad (\exists y \mid xy \in S_i \implies y \in S_{i+1})$;
- $\forall z \in S_i \quad (\exists y \mid zy \in S_1 \implies y \in S_{i+1})$.

Quello che viene fatto è un controllo "incrociato":

1. parto da una parola $x \in S_1$;
2. cerco se esiste una parola $w \in S_i$ che abbia x come prefisso;
3. se esiste aggiungo $y = w - x$ all'insieme S_{i+1};
4. faccio lo stesso invertendo S_1 con S_i.

Fermiamo l'applicazione di queste regole quando:

- S_i contiene un elemento di A: il codice c non è univocamente decodificabile;
- $S_i = \varnothing$: il codice c è univocamente decodificabile.

4.2.2. Esempi

Sia $A = \{A, E, C, ABB, CED, BBEC\}$, e sia $S_1 = A$.

Generiamo gli insiemi S_i con le due regole presentate.

- $S_2 = \{BB, ED\}$;
  - BB: $A \in S_1 \land ABB \in S_1$;
  - ED: $C \in S_1 \land CED \in S_1$;
- $S_3 = \{D, EC\}$;
  - D: $E \in S_1 \land ED \in S_2$;
  - EC: $BB \in S_2 \land BBEC \in S_1$;
- $S_4 = \{C\}$;
  - C: $E \in S_1 \land EC \in S_3$.

Ci fermiamo perché $C \in S_4$ e $C \in S_1$, quindi il codice non è univocamente decodificabile.

5. Lezione 05

5.1. Upper bound valore atteso

La scorsa lezione abbiamo mostrato che $H_D(\mathrm{X}) \leq \mathbb{E}[l_c]$, ma *quanto sono vicini questi due valori?*

Se la distanza è alta vuol dire che il nostro codice è poco efficiente.

Teorema Dato c codice istantaneo di Shannon con lunghezze $l_1, ..., l_m$ tali che

$$l_i = \left\lceil \log_D \frac{1}{p_i} \right\rceil \quad \forall i \in \{1, ..., m\}$$

allora

$$\mathbb{E}[l_c] < H_D(\mathrm{X}) + 1.$$

Dimostrazione

$$\begin{aligned}\mathbb{E}[l_c] &= \sum_{i=1}^{m} p_i l_i = \sum_{i=1}^{m} p_i \left\lceil \log_D \frac{1}{p_i} \right\rceil \\ &< \sum_{i=1}^{m} p_i \left(\log_D \frac{1}{p_i} + 1 \right) = \sum_{i=1}^{m} p_i \log_D \frac{1}{p_i} + \sum_{i=1}^{m} p_i = H_D(\mathrm{X}) + 1 \quad .\end{aligned}$$

□

Questo teorema ci dice che il massimo spreco per simbolo rispetto all'ottimo è di un bit: sembra un ottimo risultato, ma se ho a disposizione tanti simboli si perdono molti bit.

5.2. Estrazione a blocchi

Dati $c : \mathrm{X} \longrightarrow \mathbb{D}^+$ e $C : \mathrm{X}^+ \longrightarrow D^+$ cerchiamo di dare una definizione dell'**estrazione a blocchi** di n con la funzione $C_n : \mathrm{X}^n \longrightarrow D^+$.

Lemma $l_c(x_1, ..., x_n) \geq l_{C_n}(x_1, ..., x_n)$.

Dimostrazione

$$\begin{aligned}l_C(x_1, ..., x_n) &= \sum_{i=1}^{n} \left\lceil \log_D \frac{1}{p_i} \right\rceil \\ &\geq \left\lceil \sum_{i=1}^{m} \log_D \frac{1}{p_i} \right\rceil = \left\lceil \log_D \frac{1}{\prod_i p_i} \right\rceil = \left\lceil \log_D \frac{1}{p(x_1, ..., x_n)} \right\rceil = l_{C_n}(x_1, ..., x_n).\end{aligned}$$

□

In poche parole, la lunghezza di un messaggio trattato con i singoli simboli è non minore della lunghezza del messaggio stesso trattato a blocchi.

Infatti, definiamo

$$C_n : \mathrm{X}^n \longrightarrow D^+$$

come la funzione che estrae messaggi di n simboli. Prima avevamo l'estrazione di n simboli, ora l'estrazione di *un messaggio* di n simboli.

In questo modo paghiamo 1 bit ogni n simboli, ma abbiamo un modello sorgente più complesso.

Esiste una relazione tra $H(x_1, ..., x_n)$ e $H(x)$?

Definiamo prima di tutto $H(x_1, ..., x_n)$ come

$$H(x_1, ..., x_n) = \sum_{x_1, ..., x_n} p_n(x_1, ..., x_n) \log_2 \frac{1}{p_n(x_1, ..., x_n)}.$$

Inoltre, sapendo che

$$p_n(x_1, ..., x_n) = \prod_i p(x_i)$$

e che

$$\log_2 \frac{1}{\prod_{i=1}^n p(x_i)} = \log_2 \prod_{i=1}^n p(x_i)^{-1} = \sum_{i=1}^n \log_2 p(x_1)^{-1} = \sum_{i=1}^n \log_2 \frac{1}{p(x_i)},$$

possiamo affermare che

$$H(x_1, ..., x_n) = \sum_{x_1} \cdots \sum_{x_n} \prod_{i=1}^n p(x_i) \sum_{i=1}^n \log_2 \frac{1}{p(x_i)}.$$

Vediamo il caso $n = 2$ per semplicità, poi diamo una versione generale.

$$\begin{aligned}
H(x_1, ..., x_n) &= \sum_{x_1} \sum_{x_2} \prod_{i=1}^2 p(x_i) \left(\log_2 \frac{1}{p(x_1)} + \log_2 \frac{1}{p(x_2)} \right) = \\
&= \sum_{x_1} \sum_{x_2} p(x_1)p(x_2) \log_2 \frac{1}{p(x_1)} + p(x_1)p(x_2) \log_2 \frac{1}{p(x_2)} = \\
&= \sum_{x_1} \sum_{x_2} p(x_1)p(x_2) \log_2 \frac{1}{p(x_1)} + \sum_{x_1} \sum_{x_2} p(x_1)p(x_2) \log_2 \frac{1}{p(x_2)} = \\
&= \sum_{x_1} p(x_1) \log_2 \frac{1}{p(x_1)} \sum_{x_2} p(x_2) + \sum_{x_1} p(x_1) \sum_{x_2} p(x_2) \log_2 \frac{1}{p(x_2)} = \\
&= H(x_1) + H(x_2).
\end{aligned}$$

In generale possiamo affermare che

$$H(x_1, ..., x_n) = nH(\mathrm{X}).$$

5.3. Primo teorema di Shannon

Teorema *(Primo teorema di Shannon)* Sia $C_n : \mathrm{X}_n \longrightarrow D^+$ un codice di Shannon D-ario a blocchi per la sorgente $\langle \mathrm{X}, p \rangle$, ovvero

$$l_{C_n}(x_1, ..., x_n) = \left\lceil \log_D \frac{1}{p_n(x_1, ..., x_n)} \right\rceil.$$

Allora

$$\lim_{n\to\infty} \frac{1}{n}\mathbb{E}\left[l_{C_n}\right] = H_D(\mathrm{X}).$$

Dimostrazione
Sappiamo che

$$nH_D(\mathrm{X}) = H_D(X_1, ..., X_n) \leq \mathbb{E}\left[l_{C_n}\right] < H_D(X_1, ..., X_n) + 1 = nH_D(\mathrm{X}) + 1.$$

Dividiamo entrambi i membri per n e otteniamo

$$H_D(\mathrm{X}) \leq \frac{1}{n}\mathbb{E}\left[l_{C_n}\right] < H_D(\mathrm{X}) + \frac{1}{n}.$$

Se $n \to \infty$ allora $H_D(\mathrm{X}) = H_D(\mathrm{X}) + \frac{1}{n}$ e quindi

$$\frac{1}{n}\mathbb{E}\left[l_{C_n}\right] = H_D(\mathrm{X})$$

per il teorema degli sbirri. □

Il primo teorema di Shannon ci indica che il valore atteso si "schiaccia" sull'entropia col crescere della dimensioni dei blocchi: quindi, se facciamo crescere la dimensione del blocco paghiamo poco in termini di bit.

5.4. Approssimare i modelli sorgente

A volte non conosciamo a priori il modello $\langle \mathrm{X}, p\rangle$ che abbiamo a disposizione, quindi dobbiamo stimare quest'ultimo con un altro modello $\langle Y, q\rangle$.

Abbiamo a disposizione l'entropia relativa $\Delta(\mathrm{X} \parallel Y)$: infatti, quest'ultima ci dice quanto sono distanti i due modelli.

Teorema Dato il modello sorgente $\langle \mathrm{X}, p\rangle$, se $c : \mathrm{X} \longrightarrow D^+$ è un codice di Shannon con $l_c(x) = \left\lceil \log_D \frac{1}{q(x)} \right\rceil$, dove q è una distribuzione di probabilità su X, allora

$$\mathbb{E}[l_c] < \Delta(\mathrm{X} \parallel Y) + H_D(\mathrm{X}) + 1.$$

Dimostrazione

$$\begin{aligned}
\mathbb{E}[l_c] &= \sum_{x\in\mathrm{X}} p(x)\left\lceil \log_D \frac{1}{q(x)} \right\rceil \\
&< \sum_{x\in\mathrm{X}} p(x)\left(\log_D \frac{1}{q(x)} + 1\right) = \sum_{x\in\mathrm{X}} p(x)\log_D \frac{1}{q(x)} + \sum_{x\in\mathrm{X}} p(x) \\
&= \sum_{x\in\mathrm{X}} p(x)\log_D\left(\frac{1}{q(x)}\frac{p(x)}{p(x)}\right) + 1 = \sum_{x\in\mathrm{X}} p(x)\log_D \frac{p(x)}{q(x)} + \sum_{x\in\mathrm{X}} p(x)\log_D \frac{1}{p(x)} + 1 \\
&= \Delta(\mathrm{X} \parallel Y) + H_D(\mathrm{X}) + 1.
\end{aligned}$$

□

6. Lezione 06

6.1. Algoritmo di Huffman

Avevamo già visto l'algoritmo di Huffman per la costruzione di un codice di Huffman, ma visto che non ce lo ricordiamo lo rivediamo.

Per costruire un codice di Huffman bisogna:
1. ordinare i simboli sorgente in base alla probabilità decrescente;
2. crea un modello fittizio in cui i D simboli meno probabili vengono raggruppati e sostituiti con un nuovo simbolo la cui probabilità è la somma delle probabilità dei simboli sostituiti;
3. ripeti dal punto 1 se la sorgente contiene più di D simboli.

Avevamo inoltre aggiunto un numero k di simboli a probabilità 0 per verificare che

$$m + k \bmod D - 1 \equiv 1,$$

con m numero di simboli sorgente.

6.2. Codici di Huffman

Presentiamo adesso un teorema permette di capire se il codice di Huffman, generato con l'algoritmo appena ripreso, sia buono o meno.

Teorema Dati una sorgente $\langle \mathrm{X}, p \rangle$ e $D > 1$, il codice D-ario c di Huffman minimizza $\mathbb{E}[l_c]$ tra tutti i codici istantanei per la medesima sorgente.

Per dimostrare questo teorema dobbiamo prima enunciare un fatto.

Lemma Sia c' un codice D-ario di Huffman per la sorgente $\mathrm{X}' = \{x_1, ..., x_{m-D+1}\}$ con probabilità $p_1 \geq ... \geq x_{m-D+1}$. Sia X la sorgente ottenuta togliendo da X′ il simbolo x_k e aggiungendo D nuovi simboli $x_{m-D+2}, ..., x_{m+1}$ con probabilità $p_{m-D+2}, ..., p_{m+1} < p_{m-D+1}$. Inoltre, deve valere $p_{m-D+2} + ... + p_{m+1} = p_k$. Allora vale

$$c(x) = \begin{cases} c'(x) & \text{se } x \neq x_k \\ ic'(x_k) & \text{se } k \in \{m - D + 2, m + 1\} \quad \forall i \in \{0, ..., D - 1\} \end{cases}$$

è un codice di Huffman per la sorgente.

Grazie a questo fatto dimostriamo il teorema precedente.

Dimostrazione
Dimostriamo per induzione su m.

Passo base: $m = 2$

Huffman produce il codice $c(x_1) = 0$ e $c(x_2) = 1$ che p ottimo, qualunque sia la distribuzione di probabilità.

Passo induttivo: $m > 2$

Supponiamo ora Huffman ottimo per $k \leq m - 1$ e dimostriamolo per m.

Fissiamo $\langle \mathrm{X}, p \rangle$ sorgente di m simboli. Sia $\mathrm{X} = \{..., u, ..., v\}$, con $p(u), p(v)$ minime. Definiamo $\langle \mathrm{X}', p' \rangle$ con $u, v \in \mathrm{X}$ rimpiazzate da $z \in \mathrm{X}'$. La funzione p' è tale che

$$p'(x) = \begin{cases} p(x) & \text{se } x = z \\ p(u) + p(v) & \text{se } x = z \end{cases} .$$

Sia c' il codice di Huffman per $\langle \mathrm{X}', p \rangle$. Dato che $|\mathrm{X}'| = m - 1$ allora c' è ottimale per ipotesi induttiva.

Il codice c per X è definito come

$$c(x) = \begin{cases} c'(x) & \text{se } x \notin \{u, v\} \\ c'(z) \cdot 0 & \text{se } x = u \\ c'(z) \cdot 1 & \text{se } x = v \end{cases} .$$

Dimostriamo che il codice c è ottimo.

Esprimiamo il valore atteso

$$\mathbb{E}[l_c] = \sum_{x \in \mathrm{X}} l_c(x) p(x)$$

in termini di X' come

$$\begin{aligned} \mathbb{E}[l_c] &= \sum_{x \in \mathrm{X}'} l_{c'}(x) p'(x) - l_{c'}(z) p'(z) + l_c(u) p(u) + l_c(v) p(v) = \\ &= \mathbb{E}[l_{c'}] - l_{c'}(z) p'(z) + (l_{c'}(z) + 1) p(u) + (l_{c'}(z) + 1) p(v) = \\ &= \mathbb{E}[l_{c'}] - l_{c'}(z) p(z) + (l_{c'}(z) + 1)(p(u) + p(v)) = \\ &= \mathbb{E}[l_{c'}] - l_{c'}(z) p(z) + (l_{c'}(z) + 1) p'(z) = \mathbb{E}[l_{c'}] - l_{c'}(z) p(z) + l_{c'}(z) p'(z) + p'(z) = \\ &= \mathbb{E}[l_{c'}] + p'(z) \quad . \end{aligned}$$

Per dimostrare l'ottimalità di c consideriamo un codice c_2 per $\langle \mathrm{X}, p \rangle$ e verifichiamo che $\mathbb{E}[l_c] \leq \mathbb{E}\left[l_{c_2}\right]$.

Sia c_2 istantaneo per $\langle \mathrm{X}, p \rangle$ e siano $r, s \in \mathrm{X}$ tali che $l_{c_2}(r)$ e $l_{c_2}(s)$ sono massimi. Senza perdita di generalità assumiamo che r, s siano fratelli nell'albero di codifica di c_2. Infatti:

- se non fossero fratelli e avessero un altro fratello (chiamato f fratello di s) scegliamo s e f al posto di s e r;
- se non avessero fratelli possiamo sostituire le loro codifiche con quelle del padre fino a riportarci in una situazione in cui abbiano un fratello.

Definiamo il codice $\overline{c}_2$ tale che

$$\overline{c}_2 = \begin{cases} c_2(x) \text{ se } x \notin \{u, v, r, s\} \\ c_2(u) \text{ se } x = r \\ c_2(r) \text{ se } x = u \\ c_2(v) \text{ se } x = s \\ c_2(s) \text{ se } x = v \end{cases} .$$

Abbiamo scambiato la codifica di r con quella di u e quella di s con quella di v.

Analizziamo $\mathbb{E}\left[l_{\bar{c}_2}\right] - \mathbb{E}\left[l_{c_2}\right]$ per dimostrare che il primo valore è minore o uguale del secondo.

$$\begin{aligned}
\mathbb{E}\left[l_{c_2}\right] - \mathbb{E}\left[l_{c_2}\right] &= p(r)l_{c_2}(u) + p(u)l_{c_2}(r) + p(s)l_{c_2}(v) + p(v)l_{c_2}(s) \\
&\quad -p(r)l_{c_2}(r) - p(u)l_{c_2}(u) - p(s)l_{c_2}(s) - p(v)l_{c_2}(v) = \\
&= p(r)\Big(l_{c_2}(u) - l_{c_2}(r)\Big) + p(u)\Big(l_{c_2}(r) - l_{c_2}(u)\Big) + \\
&\quad +p(s)\Big(l_{v_2}(v) - l_{c_2}(s)\Big) + p(v)\Big(l_{c_2}(s) - l_{c_2}(v)\Big) = \\
&= (p(r) - p(u))\Big(l_{c_2}(u) - l_{c_2}(r)\Big) + (p(s) - p(v))\Big(l_{c_2}(v) - l_{c_2}(s)\Big).
\end{aligned}$$

Sapendo che:

- $p(r) - p(u) \geq 0$ dato che u è minimo;
- $l_{c_2}(u) - l_{c_2}(r) \leq 0$;
- $p(s) - p(v) \geq 0$ dato che v è minimo;
- $l_{c_2}(v) - l_{c_2}(s) \leq 0$.

Stiamo sommando due quantità negative, quindi

$$\mathbb{E}\left[l_{\bar{c}_2}\right] - \mathbb{E}\left[l_{c_2}\right] \leq 0 \Longrightarrow \mathbb{E}\left[l_{\bar{c}_2}\right] \leq \mathbb{E}\left[l_{c_2}\right].$$

□

7. Lezione 07

7.1. Codici di Huffman

Non abbiamo finito la dimostrazione pookie!

Dimostrazione

Introduciamo ora il codice $\overline{c}_{2'}$ fatto come segue:

$$\overline{c}_{2'} = \begin{cases} \overline{c}_2(x) & \text{se } x \neq z \\ \omega & \text{se } x = z \end{cases} \quad .$$

Questo codice è definito sulla sorgente $\langle \mathrm{X}', p' \rangle$. Inoltre, dopo aver scambiato r e s con u e v, questi ultimi sono fratelli, quindi:

- $\overline{c}_2(u) = \omega \cdot 0$;
- $\overline{c}_2(v) = \omega \cdot 1$.

Ma allora

$$\begin{aligned} \mathbb{E}\left[l_{\overline{c}_2}\right] &= \sum_{x \in \mathrm{X}' \mid x \neq z} p'(x) l_{\overline{c}_2}(x) + p(u)\left(l_{\overline{c}_{2'}} + 1\right) + p(v)\left(l_{\overline{c}_{2'}}(z) + 1\right) = \\ &= \sum_{x \in \mathrm{X}' \mid x \neq z} p'(x) l_{\overline{c}_2}(x) + p'(z) l_{\overline{c}_{2'}}(z) + p'(z) = \mathbb{E}\left[l_{\overline{c}_{2'}}\right] + p'(z) \\ &\geq \mathbb{E}[l_{c'}] + p'(z) \quad . \end{aligned}$$

Mettendo insiemi i due risultati otteniamo

$$\mathbb{E}[l_c] = \mathbb{E}[l_{c'}] + p'(z) \leq \mathbb{E}\left[l_{c_{2'}}\right] + p'(z) = E\left[l_{\overline{c}_2}\right] \leq \mathbb{E}\left[l_{c_2}\right] \quad .$$

Quindi

$$\mathbb{E}[l_c] \leq \mathbb{E}\left[l_{c_2}\right] \quad .$$

□

7.2. Disuguaglianza di Kraft-McMillan

Per cercare il codice ottimo ci siamo ristretti ai codici istantanei, ma così facendo rischiamo di lasciare fuori dei codici che potrebbero essere ottimi nonostante non siano istantanei.

Vedremo però che questi codici non esistono, dato che anche i codici univocamente decodificabili seguono la disuguaglianza di Kraft.

Teorema *(Disuguaglianza di Kraft-McMillan)* Siano $l_1, \ldots, l_m$ lunghezze positive. Allora queste ultime sono lunghezze delle parole di codice di un codice D-ario univocamente decodificabile per una sorgente di m simboli se e solo se

$$\sum_{i=1}^{m} D^{-l_i} \leq 1.$$

Per dimostrare questo teorema ci servirà l'estensione k-esima di un codice c definita come

$$C_k : \mathrm{X}^k \longrightarrow D^+.$$

Dimostrazione

[$\Longleftarrow$] Questa implicazione è banale: infatti, se vale la disuguaglianza di Kraft, le lunghezze $l_1, ..., l_m$ sono le lunghezze di un codice istantaneo, che è anche univocamente decodificabile.

[$\Longrightarrow$] Vale $\forall k \geq 1$ l'uguaglianza

$$\left(\sum_{x \in \mathrm{X}} D^{-l_c(x)}\right)^k = \sum_{x_1} \cdots \sum_{x_k} D^{-l_c(x_1)} \cdot \ldots \cdot D^{-l_c(x_k)} \quad .$$

Andiamo avanti dicendo che

$$\sum_{x_1} \cdots \sum_{x_k} D^{-l_c(x_1)} \cdot \ldots \cdot D^{-l_c(x_k)} = \sum_{x_k \in \mathrm{X}^k} D^{-(l_c(x_1)+\ldots+\ldots l_c(x_k))} = \sum_{x_k \in \mathrm{X}^k} D^{-l_{C_k}(x_k)} \quad .$$

In questo caso

$$l_{C_k}(x_k) = l_c(x_1) + \ldots + l_c(x_k).$$

Introduciamo l'insieme X_n^k come

$$\left\{x_k \in \mathrm{X}^k \mid l_{C_k}(x_k) = n\right\}.$$

Andiamo ancora avanti con l'uguaglianza:

$$\sum_{x_k \in \mathrm{X}^k} D^{-l_{C_k}(x_k)} = \sum_{n=1}^{kl_{\max}} \sum_{x_k \in \mathrm{X}^k} D^{-l_{C_k}(x_k)} = \sum_{n=1}^{kl_{\max}} |X_n^k| \; D^{-n} \quad .$$

Visto che c è univocamente decodificabile C è iniettiva, quindi $|X_n^k| \leq |D^n|$, e quindi

$$\sum_{n=1}^{kl_{\max}} |X_n^k| \; D^{-n} \leq \sum_{n=1}^{kl_{\max}} D^n D^{-n} \leq kl_{\max},$$

ma allora

$$\left(\sum_{x \in \mathrm{X}} D^{-l_c(x)}\right)^k \leq kl_{\max}.$$

Poniamo $\sum_{x \in \mathrm{X}} D^{-l_c(x)} = M$ e vediamo quanto vale questa quantità.

Visto che vogliamo M^x sotto $xl_{\max}$, allora M deve stare tra 0 e 1, quindi

$$\left(\sum_{x \in \mathrm{X}} D^{-l_c(x)}\right)^k \leq 1.$$

□

8. Lezione 08

8.1. Esercizi di probabilità

8.1.1. Esercizio 01

Sapendo che la probabilità di ottenere un messaggio corrotto è $\frac{1}{8}$, quanti bit servono per rappresentarlo?

RISPOSTA:

8.1.2. Esercizio 02

Qual è la probabilità di ottenere 4 messaggi dove il primo è corretto e gli altri 3 no, sapendo che la probabilità di ottenere un messaggio corretto è uguale a $\frac{1}{8}$?

RISPOSTA:

8.1.3. Esercizio 03

Preso l'esercizio precedente, quanti bit ci servono?

RISPOSTA:

8.1.4. Esercizio 04

Abbiamo un dato a 6 facce lanciato 20 volte. Qual è la probabilità di:

- fare 20 lanci senza mai ottenere 5?
  - RISPOSTA:
- fare 20 lanci ottenendo 5 solo una volta?
  - RISPOSTA:
- fare 20 lanci ottenendo 5 almeno una volta?
  - RISPOSTA:

8.2. Numero di bit necessari

Quanti bit ci servono per comunicare il risultato di un certo evento?

Se usiamo un codice istantaneo sappiamo che

$$H(\mathrm{X}) < \#\text{bit} \ .$$

Se avessimo due variabili? Ovvero, *se avessimo due risultati da comunicare?*

Introduciamo l'**entropia congiunta** come

$$H(\mathrm{X}, Y) = \sum_{x \in \mathrm{X}} \sum_{y \in Y} p(x, y) \log_2 \frac{1}{p(x, y)}.$$

Se invece avessimo un evento condizionante? Ovvero, *se il ricevente conosce* X. *quanti bit ci servono per comunicare Y?*

Introduciamo allora l'**entropia condizionata** come

$$\begin{aligned} H(Y \mid \mathrm{X}) &= \sum_{x \in \mathrm{X}} p(x) H(Y \mid \mathrm{X} = x) = \\ &= \sum_{x \in \mathrm{X}} p(x) \left(\sum_{y \in Y} p(y \mid x) 2 p(y \mid x) \right) = \\ &= \sum_{x \in X} \sum_{y \in Y} p(x, y) 2 p(y \mid x). \end{aligned}$$

In questo caso:
- $p(x, y)$ è detta **probabilità congiunta** ed è la probabilità che avvengano x e y;
- $p(x) = \sum_{y \in Y} p(x, y)$ è detta **probabilità marginale**;
- $p(y \mid x) = \frac{p(x,y)}{p(x)}$ è detta **probabilità condizionale**.

Lemma *(Chain Rule per l'entropia)* Vale la seguente uguaglianza:

$$H(\mathrm{X}, Y) = H(\mathrm{X}) + H(Y \mid \mathrm{X}) = H(Y) + H(\mathrm{X} \mid Y).$$

Vale inoltre, negli spazi condizionati, l'uguaglianza:

$$H(\mathrm{X}, Y \mid Z) = H(\mathrm{X}, Z) + H(Y \mid X, Z).$$

8.3. Esercizi su entropia

8.3.1. Esercizio 01

Sia $x \in \mathrm{X}$ una variabile rappresentante l'estrazione di un numero tra 0 e 9, e sia Y definita come $Y = \mathrm{X} + 2 \bmod 10$. Quanto vale $H(Y|\mathrm{X})$?

RISPOSTA:

8.3.2. Esercizio 02

Siano $\mathrm{X} = \{-1, 0, 1\}$ e $Y = X^2$. Quanto vale $H(Y \mid \mathrm{X})$? E invece $H(\mathrm{X} \mid Y)$?

RISPOSTA:

8.3.3. Esercizio 03

Dato un sistema SCR (sorgente-canale-ricevente), sia M una matrice che rappresenta il canale C.

$$M = \begin{pmatrix} & b_1 & b_2 & b_3 & b_4 & b_5 \\ a_1 & 0.3 & 0.1 & 0.3 & 0.1 & 0.1 \\ a_2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ a_3 & 0.3 & 0.3 & 0.1 & 0.1 & 0.2 \\ a_4 & 0.3 & 0.3 & 0.3 & 0.05 & 0.05 \end{pmatrix}.$$

Siano inoltre $\mathrm{X} = \{a_1, a_2, a_3, a_4\}$ e $p = [0.2, 0.2, 0.2, 0.4]$ per S(?).

Quanto vale $H(R \mid S)$?

RISPOSTA:

8.3.4. Esercizio 04

Calcolare $H(R \mid S)$ come nell'esercizio precedente usando però

$$M = \begin{pmatrix} 0.2 & 0.2 & 0.3 & 0.2 & 0.1 \\ 0.2 & 0.5 & 0.1 & 0.1 & 0.1 \\ 0.6 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.3 & 0.1 & 0.1 & 0.1 & 0.4 \end{pmatrix}$$

e $p = [0.2, 0.3, 0.1, 0.4]$.

RISPOSTA:

8.3.5. Esercizio 05

Posso ottenere lo stesso risultato dell'esercizio in un altro modo?

Ad esempio:

- usando la chain rule?
  - RISPOSTA:
- usando la chain rule, regola 1 modificata?
  - RISPOSTA:

9. Lezione 09

9.1. Informazione mutua

L'**informazione mutua** è un parametro che fa riferimento a due variabili casuali e indica quanta informazione viene rilasciata da una rispetto all'altra. È definita come

$$I(\mathrm{X}, Y) = \sum_{x \in \mathrm{X}} \sum_{y \in Y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}.$$

Lemma L'informazione mutua è non negativa, ovvero

$$I(\mathrm{X}, Y) \geq 0.$$

Dimostrazione
Applicando la definizione di probabilità congiunta otteniamo

$$\begin{aligned} I(\mathrm{X}, Y) &= \sum_{x \in \mathrm{X}} \sum_{y \in Y} p(x, y) \log \frac{p(y)p(x \mid y)}{p(x)p(y)} = \\ &= \sum_{x \in \mathrm{X}} \sum_{y \in Y} p(x, y) \log \frac{1}{p(x)} + \sum_{x \in \mathrm{X}} \sum_{y \in Y} p(x, y) \log p(x \mid y) = \\ &= H(\mathrm{X}) - H(\mathrm{X} \mid Y) \geq 0. \end{aligned}$$

□

Se X e Y sono **indipendenti** quanto vale l'informazione mutua? Ovviamente zero: infatti,

$$H(\mathrm{X} \mid y) = H(\mathrm{X}) \implies I(\mathrm{X}, Y) = H(\mathrm{X}) - H(\mathrm{X}) = 0.$$

E se invece $\mathrm{X} = g(Y)$? In questo caso $H(\mathrm{X} \mid Y) = 0$, quindi

$$I(\mathrm{X}, Y) = H(X) - \underbrace{H(\mathrm{X} \mid Y)}_{0} = H(\mathrm{X}).$$

Infine, vale anche

$$I(\mathrm{X}, Y \mid Z) = \sum_{x \in \mathrm{X}} \sum_{y \in Y} p(x, y \mid z) \log \frac{p(x, y \mid z)}{p(y \mid z)p(x \mid z)}.$$

9.2. Data processing inequality

Introduciamo un teorema molto importante, che però non dimostreremo.

Teorema *(Data processing inequality)* Siano X, Y, Z variabili casuali a dominio finito tali che $p(x, y, z)$ soddisfa

$$p(x, y \mid z) = p(x \mid y)p(z \mid y) \quad \forall x, y, z,$$

cioè x, z sono indipendenti dato y. Allora l'informazione mutua tra X e Y è non minore dell'informazione mutua tra X e Z. Formalmente,

$$I(\mathrm{X}, Y) \geq I(\mathrm{X}, Z).$$

Corollario Vale la disequazione

$$I(\mathrm{X}, Y) \geq I(\mathrm{X}, Y \mid Z).$$

9.3. Disuguaglianza di Fano

Un altro teorema importante che enunciamo ma che non dimostriamo è la **disuguaglianza di Fano**.

Teorema *(Disuguaglianza di Fano)* Siano X, Y variabili casuali su domini $\mathbb{X}, \mathbb{Y}$ finiti. Sia $g : \mathbb{Y} \longrightarrow \mathbb{X}$ la funzione di decodifica e p_e la probabilità di errore $p_e = p(g(Y) \neq \mathrm{X})$, allora

$$p_e \geq \frac{H(\mathrm{X} \mid Y) - 1}{\log_2 |\mathbb{X}|}.$$

10. Lezione 10

10.1. Canale

10.1.1. Introduzione

Dobbiamo codificare un messaggio sul **canale**. Definiamo quest'ultimo come una tripla

$$C \equiv \langle \mathbb{X}, \mathbb{Y}, p(y \mid x)$$

dove:

- $\mathbb{X}$ **insieme dei simboli di input**;
- $\mathbb{Y}$ **insieme dei simboli di output**;
- $p(y \mid x)$ **probabilità** di ottenere y dato x. Formalmente sarebbe più corretto scrivere

$$p(Y = y \mid X = x)$$

  visto che x e y sono due *realizzazioni* delle variabili aleatorie X e Y.

Non è detto che $\mathbb{X} = \mathbb{Y}$: potremmo avere due diversi insiemi di simboli.

Useremo **canali discreti e senza memoria**, ovvero canali dove il bit ricevuto dipende solamente dal bit appena inviato.

Se un canale viene usato n volte, qual è la probabilità che, inviato il messaggio $x^n = \{x_1, ..., x_n\}$, riceviamo il messaggio $y^n = \{y_1, ..., y_n\}$?

Scriviamo questa probabilità come $p(y^n \mid x^n)$, ma visto che i simboli sono indipendenti tra loro vale

$$p(y_n \mid y^{n-1}, x^n) \cdot p(y_{n-1} \mid y^{n-2}, x^n) \cdot \dots \cdot p(y_1 \mid y^0, x^n).$$

Visto che il canale è senza memoria allora

$$p(y_n \mid x_n) \cdot \dots \cdot p(y_1 \mid x_1) = \prod_{i=1}^{n} p(y_i \mid x_i).$$

In poche parole, consideriamo solo l'ultimo simbolo inviato.

10.1.2. Canale binario senza rumore

Il **canale binario senza rumore** è il più facile e ha due rappresentazioni possibili.

10.1.2.1. Rappresentazione grafica

$$0 \rightsquigarrow 0$$
$$1 \rightsquigarrow 1$$

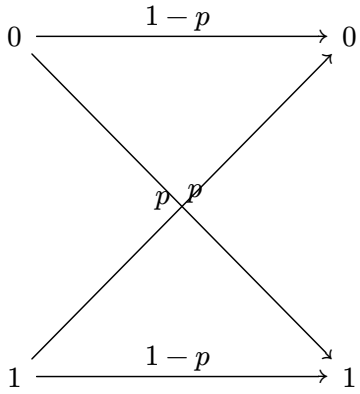
10.1.2.2. Rappresentazione matriciale

$$\begin{pmatrix} X \setminus Y & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

10.1.3. Canale binario simmetrico

Anche il **canale binario simmetrico** ha due rappresentazioni possibili.

10.1.3.1. Rappresentazione grafica



10.1.3.2. Rappresentazione matriciale

$$\begin{pmatrix} X \setminus Y & 0 & 1 \\ 0 & 1-p & p \\ 1 & p & 1-p \end{pmatrix}$$

10.1.4. Capacità del canale

La **capacità del canale** indica quanta informazione possiamo mandare sul canale. Formalmente è definita come

$$C = \max_{p(\mathrm{X})} I(\mathrm{X}, Y),$$

dove $\max_{p(x)}$ prende in considerazione tutte le possibili distribuzioni di $p(x)$, ovvero la probabilità di generare un simbolo sorgente.

10.1.4.1. Canale binario senza rumore

Ricaviamo la capacità riscrivendo l'informazione mutua come

$$C = \max_{p(\mathrm{X})}(H(\mathrm{X}) - H(\mathrm{X} \mid Y)).$$

Visto che Y non ci dà incertezza su X, allora $H(X \mid Y) = 0$ e quindi

$$C = \max_{p(\mathrm{X})} H(X).$$

Scegliendo $p(0) = p(1) = \frac{1}{2}$ massimizziamo l'entropia e quindi anche la capacità del canale, che raggiunge il valore $C = 1$.

10.1.4.2. Canale binario simmetrico

Osserviamo prima di tutto che

$$I(\mathrm{X}, Y) = H(Y) - H(Y \mid \mathrm{X}) = H(Y)H(Y \mid \mathrm{X} = 0)p(\mathrm{X} = 0) - H(Y \mid \mathrm{X} = 1)p(\mathrm{X} = 1).$$

Notiamo poi che

$$\begin{aligned} H(Y \mid \mathrm{X} = 0) &= -p(y = 0 \mid x = 0) \log_2 p(y = 0 \mid x = 0) - p(y = 1 \mid x = 0) \log_2 p(y = 1 \mid x = 0) = \\ &= -(1-p) \log_2 (1-p) - p \log_2 p = H(p), \end{aligned}$$

dove $H(p)$ rappresenta l'entropia di una variabile aleatoria bernoulliana di parametro p. Analogamente, possiamo dimostrare che $H(Y \mid X = 1) = H(p)$.

La capacità del canale si riduce quindi a

$$C = \max_{p(\mathrm{X})} H(Y) - H(p).$$

Cerchiamo il massimo valore di $H(Y)$: analizziamo per ora $P(Y = 1)$.

$$\begin{aligned} P(Y = 1) &= P(Y = 1 \mid \mathrm{X} = 0)P(\mathrm{X} = 0) + P(Y = 1 \mid \mathrm{X} = 1)P(\mathrm{X} = 1) = \\ &= pP(\mathrm{X} = 0) + (1 - p)P(\mathrm{X} = 1). \end{aligned}$$

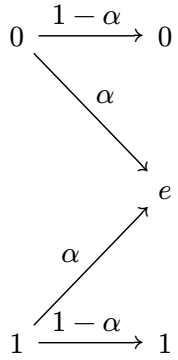
Notiamo che quando $P(X = 1) = \frac{1}{2}$ abbiamo $P(Y = 1) = \frac{1}{2}$: grazie a questa scelta massimizziamo $H(Y) = 1$, quindi concludiamo che

$$C = 1 - H(p).$$

11. Lezione 11

11.1. Canale binario a cancellazione

L'altro canale che introduciamo è il **canale binario a cancellazione**, detto anche **BEC** (*binary erased channel*).



Il termine e indica l'errore: con probabilità $1-\alpha$ riceviamo il simbolo giusto, mentre con probabilità α si verifica un errore.

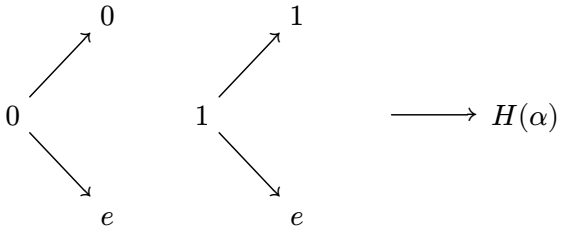
Calcoliamo la capacità di questo canale osservando che

$$H(Y \mid \mathrm{X}=0) = H(Y \mid X=1) = H(\alpha)$$

e quindi che

$$H(\alpha) = H(Y \mid \mathrm{X}).$$

Lo possiamo osservare graficamente questo risultato.



Questo implica che

$$I(\mathrm{X}, Y) = H(Y) - H(\alpha).$$

Dobbiamo quindi calcolare il valore massimo di $H(Y)$. Introduciamo la variabile aleatoria Z tale che

$$Z(\mathrm{bho}) = \begin{cases} 1 & \text{se } \mathbb{Y} = e \\ 0 & \text{altrimenti} \end{cases}.$$

Questa variabile vale 1 se c'è errore, altrimenti vale 0. Notiamo che

$$H(Y, Z) = H(Y) + H(Z \mid Y),$$

ma $H(Z \mid Y) = 0$ quindi

$$H(Y, Z) = H(Y).$$

Vale anche che

$$H(Y,Z) = H(Z) + H(Y,Z),$$

ma allora

$$H(Y) = H(Z) + H(Y \mid Z).$$

Dopo questa bella catena di uguaglianze osserviamo che

$$\begin{aligned} p(Z=1) &= p(Z=1 \mid \mathrm{X}=0)p(\mathrm{X}=0) + p(Z=1 \mid \mathrm{X}=1)p(\mathrm{X}=1) = \\ &= \alpha p(\mathrm{X}=0) + \alpha p(\mathrm{X}=1) = \alpha. \end{aligned}$$

Ne consegue che $p(Z=0) = 1-\alpha$ e quindi che $H(\alpha) = H(Z)$. Manca da calcolare $H(Y \mid Z)$. Sappiamo che $p(Y=1 \mid Z=0) = p(X=1)$ e quindi che $H(Y \mid Z=0) = H(\mathrm{X})$. Possiamo allora scrivere che

$$H(Y \mid Z) = H(Y \mid Z=0)\underbrace{p(Z=0)}_{1-\alpha} + \underbrace{H(Y \mid Z=1)}_{0}p(Z=1) = H(\mathrm{X})(1-\alpha).$$

Concludiamo quindi dicendo che

$$C = \max_{p(\mathrm{X})} H(Y) - H(\alpha) = \max_{p(\mathrm{X})}(H(\alpha) + H(\mathrm{X})(1-\alpha)) - H(\alpha) = (1-\alpha)\max_{p(\mathrm{X})} H(\mathrm{X}) = 1-\alpha.$$

11.2. Codice di Fano

11.2.1. Definizione

Presentiamo l'algoritmo per costruire un codice di Fano.

Per costruire un **codice di Fano** seguiamo i seguenti passi:

1. ordiniamo le probabilità in maniera decrescente;
2. dividiamo le probabilità in due gruppi G_1, G_2 tali che

$$\sum_{g_1 \in G_1} p_{g_1} \cong \sum_{g_2 \in G_2} p_{g_2},$$

   ovvero due gruppi che hanno più o meno la stessa somma di probabilità;
3. assegniamo valore 0 agli eventi del primo gruppo G_1 e valore 1 agli eventi del secondo gruppo G_2;
4. applichiamo ricorsivamente i punti 1,2,3 finché ci sono ancora simboli da assegnare.

11.2.2. Esempio

Dati i simboli $\{A, B, C, D, E\}$ con probabilità $\{0.35, 0.25, 0.15, 0.15, 0.1\}$.

Applichiamo l'algoritmo passo passo:

1. dividiamo in due gruppi con probabilità più o meno uguali, ad esempio $\{A, B\}$ e $\{C, D, E\}$;
2. assegniamo 0 a $\{A, B\}$;
   1. dividiamo in due gruppi con probabilità più o meno uguali, ad esempio $\{A\}$ e $\{B\}$;
   2. assegniamo 0 a $\{A\}$;
   3. assegniamo 1 a $\{B\}$;
3. assegniamo 1 a $\{C, D, E\}$;
   1. dividiamo in due gruppi con probabilità più o meno uguali, ad esempio $\{C\}$ e $\{D, E\}$;
   2. assegniamo 0 a $\{C\}$;
   3. assegniamo 1 a $\{D, E\}$;
      1. dividiamo in due gruppi con probabilità più o meno uguali, ad esempio $\{D\}$ e $\{E\}$;
      2. assegniamo 0 a $\{D\}$;
      3. assegniamo 1 a $\{E\}$.

Abbiamo ottenuto il seguente codice:

- $A \longrightarrow 00$;
- $B \longrightarrow 01$;
- $C \longrightarrow 10$;
- $D \longrightarrow 110$;
- $E \longrightarrow 111$.

Notiamo come questo codice non sia ottimale: il problema è che partiamo dalla radice e già assegniamo i prefissi, quindi il contrario di quello che fa Huffman.

12. Lezione 12

12.1. Introduzione

Il problema iniziale che ci eravamo posti era suddiviso in due parti:
- compressione del messaggio;
- ridondanza del messaggio compresso.

$$\text{MSG} \rightsquigarrow \text{compressione} \rightsquigarrow \text{ridondanza}\ .$$

La parte di **ridondanza** è suddivisa in due sottoparti:
- **rilevazione** dell'errore;
- **correzione** dell'errore.

Suddividiamo gli errori in due categorie:
- **errori singoli**: colpiscono un certo numero di bit *singoli* nella parola, ad esempio $1010x01010x$;
- **errori burst**: colpiscono una serie di bit *consecutivi*, ad esempio $1xxx01xxxx01$.

Definiamo infine il **rumore bianco** come il rumore che ha effetto su tutti i bit allo stesso modo.

12.2. Esercizi di probabilità

Siano:
- n numero di bit;
- p probabilità di errore;
- $(1-p)^n$ probabilità di avere n bit giusti.

12.2.1. Esercizio 01

Qual'è la probabilità di avere esattamente un errore?

RISPOSTA:

12.2.2. Esercizio 02

Qual è la probabilità di avere esattamente l errori?

RISPOSTA:

12.2.3. Esercizio 03

Qual è la probabilità di avere al massimo l errori?

RISPOSTA:

12.2.4. Esercizio 04

Qual è la probabilità di avere un numero pari di errori?

RISPOSTA:

12.3. Codici di correzione

Consideriamo alcuni codici noti che permettono di rilevare errori in fase di trasmissioni e, in alcuni casi, di correggerli.

12.3.1. Single parity check code

Il **single parity check code** permette di identificare *un solo errore* all'interno della stringa.

Questo codice calcola la **parità** della stringa di bit come

$$p = \sum_{i=1}^{n} x_i \bmod 2$$

e la manda insieme alla stringa per permettere al ricevente di fare un check.

Non possiamo però correggere: infatti, se cambiamo due 1 in due 0 otteniamo la stessa parità e il ricevente avrà un check positivo.

Per avere dei codici di rilevazione+correzione bisogna aggiungere dei bit. Il numero di bit aggiunti è chiamata **ridondanza** ed è definita come

$$\frac{\text{BIT SPEDITI}}{\text{BIT DI INFORMAZIONE}},$$

dove il denominatore indica il numero di bit che avremmo inizialmente spedito sul canale senza ridondanza.

In questo caso, la ridondanza è

$$\frac{n}{n-1} = 1 + \frac{1}{n-1}.$$

Il valore $\frac{1}{n-1}$ è detto **ridondanza aggiunta**.

Dobbiamo trovare un compromesso tra esattezza (affidabilità) e lunghezza del messaggio (efficienza), ovvero dobbiamo sì aggiungere bit per essere affidabili ma dobbiamo anche essere efficienti.

12.3.2. Codice ASCII

12.3.2.1. Definizione

Il **codice ASCII** fu inizialmente pensato per 7 bit. In questa versione viene aggiunto un ottavo bit a quelli di informazione, chiamato bit di parità, come in precedenza.

Il codice ASCII viene usato per errori burst ma il side effect è che errori multipli possono auto-cancellarsi.

Il codice ASCII si costruisce nel seguente modo:
1. prendere il messaggio e convertirlo in binario;
2. per ogni parola aggiungiamo un bit di parità;
3. per ogni colonna calcoliamo la checksum e aggiungiamo la parola ottenuta al messaggio.

12.3.2.2. Esempio

Data la stringa “Hello NCTU” calcoliamo bit di parità e conversione in binario.

LETTERA	PAROLA IN BINARIO	BIT DI PARITÀ
H	1001000	0
e	1100101	0
l	1101100	0
l	1101100	0
o	1101111	0
	0100000	1

N	1001110	0
C	1000011	1
T	1010100	1
U	1010101	0

La checksum delle parole in binario è 1101110.

12.3.3. Codici pesati

12.3.3.1. Definizione

I **codici pesati** aggiungono una checksum al messaggio calcolata considerando la posizione dei simboli all'interno del messaggio. Proprio per questa particolarità sono detti pesati.

Procediamo nel seguente modo per calcolare la checksum.

MESSAGGIO	$\sum$	$\sum\sum$
w	w	w
x	$w+y$	$2w+x$
y	$w+x+y$	$2w+2x+y$
z	$w+x+y+z$	$2w+2x+2y+z$
checksum?	$w+x+y+z$	$2w+2x+2y+z+(w+x+y+z)$

Una volta ottenuto $t = 2w + 2x + 2y + z + (w + x + y + z)$ possiamo ricavare la checksum: infatti, siano n il numero di simboli dell'alfabeto e r il resto tra t e n, la checksum è quel numero tale che

$$r + \text{checksum} \underset{n}{\equiv} 0.$$

12.3.3.2. Esempio

Data la stringa $3B\beta 8$, trovare la sua checksum.

Prima di tutto notiamo che $n = |\{0, ..., 9, A, ..., Z, \beta\}| = 37$.

Calcoliamo le somme pesate come definito poco fa:
- 3: ha indice 3 nell'insieme, ha $\sum = 3$ e $\sum\sum = 3$;
- B: ha indice 11 nell'insieme, ha $\sum = 14$ e $\sum\sum = 17$;
- β: ha indice 36 nell'insieme, ha $\sum = 50$ e $\sum\sum = 67$;
- 8: ha indice 8 nell'insieme, ha $\sum = 58$ e $\sum\sum = 125$;
- checksum: ha $\sum = 58$ e $\sum\sum = 183$.

Facciamo $\frac{183}{37} = 4$ con resto 35: devo trovare quel numero che, sommato a 35, è uguale a 0 modulo 37.

In caso caso, la checksum vale 2.

Come controlliamo che sia giusto? Pesiamo l'indice della lettera nell'insieme con la posizione decrescente, quindi

$$3 \cdot 5 + 11 \cdot 4 + 36 \cdot 3 + 8 \cdot 2 + 2 \cdot 1 = 185$$

e controlliamo se questo valore è congruo a 0 modulo n.

12.3.4. Codici (M,N)

Abbiamo dato la definizione di canale come la tripla

$$\langle \mathbb{X}, \mathbb{Y}, p(y \mid x).$$

Diamo ora la definizione di canale su cui vengono spediti n messaggi come la tripla

$$\langle \mathbb{X}^n, \mathbb{Y}^n, p(y^n \mid x^n).$$

Siccome siamo su canali senza memoria sappiamo che

$$p(y^n \mid x^n) = \prod_{i=1}^{n} p(y_i \mid x_i).$$

Un codice (M, N) è tale che:

- M è la lunghezza del messaggio spedito sul canale. Il messaggio è formato da simboli numerati da 1 a M;
- N è il numero di volte che viene utilizzato il canale. Ad ogni utilizzo posso scrivere 0 o 1, quindi utilizzandolo N volte scriviamo N bit, e quindi un messaggio di dimensione massima di 2^N.

Introduciamo le funzioni di **codifica**

$$x^q : \{1, ..., M\} \longrightarrow \mathbb{X}^q$$

e **decodifica**

$$g : \mathbb{Y}^q \longrightarrow \{1, ..., M\}.$$

Sia

$$x_i = p(q(y^q) \neq i \mid \mathbb{X}^q = x^q(i))$$

la **probabilità di errore** sull'i-esimo simbolo. Allora la **probabilità massima** di errore è

$$x^{(n)} = \max_i x_i.$$

La **probabilità media** invece è

$$p_e^{(n)} = \frac{1}{m} \sum_{i=1}^{m} x_i.$$

Introdotte queste grandezze, vale la seguente disequazioni:

$$p_e^{(n)} \leq x^{(n)}.$$

Il **tasso di trasmissione** di un codice (M, N) è dato da

$$R = \frac{\log_2 M}{N}.$$

Nel caso di canale senza rumore abbiamo $R = 1$, negli altri casi invece non vale.

12.4. Secondo teorema di Shannon

Teorema *(Secondo teorema di Shannon)* Sia $\langle \mathbb{X}, \mathbb{Y}, p(y \mid x) \rangle$ un canale con capacità c. Allora $\forall R < c \quad \exists k_1, k_2, \ldots$ sequenza di codici dove k_n è di tipo $\left(2^{nR_n}, n\right)$ tale che

$$\lim_{n \to \infty} R_n = R$$

e

$$\lim_{n \to \infty} x^{(n)} k_n = 0.$$

R_n indica il rumore: se uguale a 1 non c'è rumore, mentre più ci avviciniamo a 0 più ne abbiamo.

Questo teorema ci dice che:

1. più il messaggio è grande, più il rumore (sempre minore di c) diventa trascurabile;
2. la probabilità massima di errore tende a 0 con il crescere della lunghezza del messaggio.

13. Lezione 13

13.1. Codici di rilevazione errori

13.1.1. ISBN-10

13.1.1.1. Definizione

Il codice **ISBN-10** è un codice univoco di identificazione dei libri (esiste anche *ISBN-13*). L'alfabeto usato è

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, x\},$$

con $x = 10$. Questo codice è un *codice pesato*.

13.1.1.2. Esempio

Il codice 0-471-24195-4 è un codice ISBN-10, ed è tale che:

- 0 è la **nazione**, in questo caso paese anglofono;
- 471 è l'**id-publisher**;
- 24195 è il **book-number**;
- 4 è l'**error detection**.

Il codice 0-52-18-4868-7 è anch'esso un codice ISBN-10, con 18-4868 uguale al publisher id. Per controllare se è corretto usiamo la procedura dei codici pesati.

MESSAGGIO	$\sum$	$\sum\sum$
5	5	5
2	7	12
1	8	20
8	16	36
4	20	56
8	28	84
6	34	118
8	42	160
7	49	209

Il codice è corretto se e solo se

$$209 \underset{11}{\equiv} 0,$$

e questo vale, quindi il codice è corretto.

13.1.2. UPC

13.1.2.1. Definizione

Il codice **UPC** (*Universal Product Code*) è il comune codice a barre. È un codice di parità a 12 cifre.

13.1.2.2. Esempio

Un codice UPC è nella forma

$$\underbrace{036000}_{\text{ID produttore}} \quad \underbrace{29145}_{\text{ID prodotto}} \quad \underbrace{2}_{\text{checksum}}.$$

Il codice è corretto se:
- sommiamo le cifre dispari e le moltiplichiamo per 3;
- sommiamo le cifre pari;
- la somma di queste due quantità deve essere 0 modulo 12.

Si conta da sinistra a partire da 1.

Controlliamo l'esempio:

$$3(0+6+0+2+1+5)+(3+0+0+9+4+2) = 42+18 = 60 \underset{11}{\equiv} 0.$$

13.2. Codici di rilevazione e correzione errori

Fin'ora abbiamo visto solo codici che rilevano gli errori ma non li correggono. Proviamo a creare un codice che corregga gli errori di trasmissione.

Sia (x_1, x_2, x_3) messaggio da spedire. Aggiungiamo (x_4, x_5, x_6) **bit di controllo** definiti come

$$\begin{cases} x_4 = x_1 + x_2 \\ x_5 = x_1 + x_3 \\ x_6 = x_2 + x_3 \end{cases}.$$

Spediamo quindi $(x_1, x_2, x_3, x_4, x_5, x_6)$ nel canale.

Lato ricevente deve valere

$$\begin{cases} y_4 = y_1 + y_2 \\ y_5 = y_1 + y_3 \\ y_6 = y_2 + y_3 \end{cases}.$$

Cosa succede:
- se sbaglio x_1 vengono errati y_4 e y_5;
- se sbaglio x_2 vengono errati y_4 e y_6;
- se sbaglio x_3 vengono errati y_5 e y_6;
- se sbaglio x_4 vengono errati y_4;
- se sbaglio x_5 vengono errati y_5;
- se sbaglio x_6 vengono errati y_6.

Questo codice riesce a correggere un errore solo, non di più. Riesce a rilevare il doppio errore ma non lo riesce a correggere.

Infatti, cosa succede:
- se sbaglio x_1 e x_2 vengono errati y_5 e y_6;
- se sbaglio x_1 e x_3 vengono errati y_4 e y_6;
- se sbaglio x_1 e x_4 vengono errati y_5;

- se sbaglio x_1 e x_5 vengono errati y_4;
- se sbaglio x_1 e x_6 vengono errati y_4, y_5 e y_6.

13.2.1. Codice a ripetizione tripla

Facciamo una copia del bit iniziale, mentre il ricevitore mantiene il bit che compare più volte.

13.2.2. Codici di tipo (n,k)

Sia C un codice di tipo (n, k), ovvero un codice che mappa una stringa s di k bit in una stringa binaria di n bit.

C è lineare se esistono le matrici $G_{k\times n}$ e $H_{(n-k)\times n}$ tali che

$$x = (x_1, ..., x_n) = (s_1, ..., s_k)G$$

e

$$Hx^T = 0^T,$$

dove x è la parola del codice e s il messaggio binario associato.

Il **rate**

$$R = \frac{n}{k}$$

indica il numero di bit inviati per bit di informazione.

13.2.2.1. Codice di Hamming

Il **codice di Hamming** è un codice di tipo $(7, 4)$ con rate $R = 1.75$. Ha le stesse caratteristiche del codice a rilevazione tripla, ma usa meno bit.

Abbiamo quindi $\left(\underbrace{p_1, p_2, p_3}_{\text{controllo}}, \underbrace{s_1, s_2, s_3, s_4}_{\text{informazione}} \right)$.

I *bit di controllo* sono calcolati come

$$\begin{cases} p_1 = s_1 + s_3 + s_4 \\ p_2 = s_1 + s_2 + s_3 \\ p_3 = s_2 + s_3 + s_4 \end{cases}.$$

In forma matriciale abbiamo

$$H = \begin{pmatrix} p_1 & p_2 & p_3 & s_1 & s_2 & s_3 & s_4 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

e la matrice generatrice

$$G = \begin{pmatrix} & p_1 & p_2 & p_3 & & & & \\ s_1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ s_2 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ s_3 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ s_4 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Ricevuto un messaggio $p_1p_2p_3$, il vettore risultante (detto **sindrome**) deve essere composto da soli 0. Se nella i-esima posizione c'è un 1 allora p_i è errato.

Per capire il bit errato usiamo il digramma di **McEliece**.

14. Lezione 14

14.1. Campi di Galois

14.1.1. Introduzione

Solitamente lavoriamo su 1 byte, quindi 8 bit: con questo numero di bit a disposizione abbiamo 256 numeri diversi, da 0 a 255. Abbiamo un problema: questi valori formano l'anello $\mathbb{Z}_{256}$, che però non garantisce la presenza di un inverso per ogni elemento, visto che 256 non è primo.

L'**inverso** di un numero a è quel valore b tale per cui $a \cdot b \underset{256}{\equiv} 1$.

Perché serve l'inverso? Serve perché la divisione, definita come $b/a = b \cdot a^{-1}$, necessita della presenza dell'inverso a^{-1}.

Lavorare su un anello è problematico, esiste almeno un elemento che non ha l'inverso.

Su un campo si dice **generatore** (o **radice primitiva**) un numero che, elevato a tutti gli elementi del campo diversi da 0, mi genera gli elementi stessi.

Se è un generatore devo ottenere 1 all'inizio (elevamento alla zero) e alla fine, come dice il **piccolo teorema di Fermat**.

Teorema *(Piccolo teorema di Fermat)* Non ricordo la definizione, ma so che vale

$$a^{p-1} \underset{p}{\equiv} 1.$$

Ci possono essere più generatori all'interno di un campo: li possiamo mappare tra loro, e quando succede quando il campo è **isomorfo**.

14.1.2. Definizione

I **campi di Galois** ci permettono di avere un campo anche se il numero non è un numero primo. Formalmente, un campo di Galois è un campo

$$\mathrm{GF}(p^n),$$

dove p è un numero primo e n un numero naturale. Nel nostro caso, usiamo il campo

$$\mathbb{Z}_{256} = \mathrm{GF}(2^8).$$

Gli elementi del campo di Galois sono i polinomi di grado $n-1$ con coefficienti in $\mathbb{Z}_p$. Nel nostro caso, i polinomi hanno grado massimo 7 e coefficienti in $\{0, 1\}$.

Presi due polinomi $a(x), b(x) \in \mathrm{GF}(2^8)$ sicuramente la loro somma (intesa come somma binaria, o meglio, come XOR binario) sarà sempre un elemento del campo, ma questo invece non vale per la moltiplicazione: infatti, potremmo uscire dal grado massimo consentito dal campo. Per "rientrare" nel campo dobbiamo ridurli facendo il modulo con un polinomio $m(x)$ irriducibile: in poche parole,

$$a(x) \cdot b(x) \bmod m(x) = r(x).$$

Di solito viene usato $m(x) = x^8 + x^4 + x^3 + x + 1$ (anche in AES), ma ce ne sono circa 30.

14.2. Codici ciclici

14.2.1. Definizione

Abbiamo la seguente gerarchia:

$$\text{BCH} \subset \text{ciclici} \subset \text{lineari}\ .$$

I **codici ciclici** sono codici (n, k) in cui il polinomio che rappresenta il messaggio è del tipo

$$m(x) = m_0 x^0 + m_1 x^1 + ... + m_{k-1} x^{k-1}$$

e il polinomio generatore è

$$g(x) = g_0 x^0 + g_1 x^1 + ... + g_{k-1} x^{k-1}.$$

I codici ciclici possono essere:

- **sistematici**: le posizioni dei bit di controllo e di informazione sono prestabiliti. La parola di codice $v(x)$ è rappresentata come

  $$v(x) = m(x)g(x),$$

  con $m(x)$ messaggio da spedire;
- **non sistematici**: la posizione dei bit di controllo non è prefissata. La parole di codice $v(x)$ è rappresentata come

  $$v(x) = x^{n-k} m(x) + r(x).$$

14.2.2. Esempio

Dato il codice $(7, 4)$ con $g(x) = 1 + x + x^3$ e $m(x) = 1 + x^2 + x^3$ trovare la parola di codice corrispondente.

Se il codice è *sistematico* allora

$$v(x) = m(x)g(x) = (1 + x^2 + x^3)(1 + x + x^3) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1.$$

Se il codice è *non sistematico* allora

$$v(x) = x^{n-k} m(x) + r(x) = x^3(1 + x^2 + x^3) + r(x) = \text{conti non fatti} = x^6 + x^5 + x^3 + 1.$$

14.2.3. Matrice generatrice

Per trovare le parole di codice possiamo usare anche la **matrice generatrice**.

In ordine:

1. scriviamo la matrice generatrice

   $$G = \begin{pmatrix} x^{k-1} g(x) \\ x^{k-2} g(x) \\ \vdots\vdots\vdots \\ x^{k-k} g(x) \end{pmatrix} \quad ;$$

2. se vogliamo ottenere la parole nella forma sistematica applichiamo una combinazione lineare delle righe per ottenere

   $$G = [I],$$

   con I matrice identità;
3. trovo le parole di codice con

   $$[c] = [d] \cdot [G],$$

   con c parola di codice e d messaggio.

Definiamo il **polinomio di parità** come

$$h(x) = \frac{x^n + 1}{g(x)}.$$

Per essere un polinomio generatore valido $g(x)$ deve dividere propriamente $x^n + 1$.