Teoria dell'informazione e della trasmissione

Indice

1. Introduzione

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.

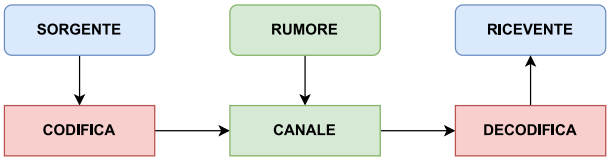
Nel luglio 1948 infatti viene pubblicato l'articolo "*A Mathematical Theory of Communication*" da parte di Bell Labs, dove Shannon pone le basi della teoria dell'informazione.

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.

1.2. 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".

1.3. 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.

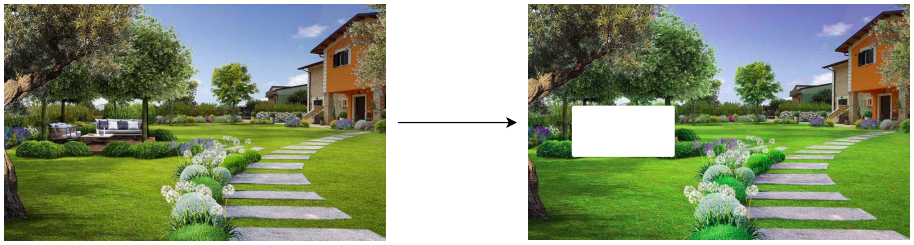
In un caso generale del *divide-et-impera* si ricade in una soluzione sub-ottimale.

1.4. 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".



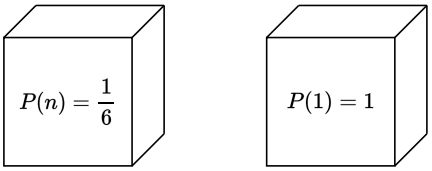
In questo secondo esempio notiamo invece che, nonostante l'informazione persa sia molto meno rispetto a quella precedente, "risalire" al contenuto perso è molto più difficile.

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.

1.5. 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.

TODO	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.

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

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.

2.1. Introduzione matematica

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

- insieme $\mathrm{X} \implies$ insieme finito di simboli che compongono i messaggi generati dalla sorgente;
- messaggio $\overline{x} \implies$ 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 $\{0, ..., D-1\} \implies$ insieme finito dei simboli che compongono il codice scelto;
- insieme $\{0, ..., D-1\}^+ \implies$ 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 $c \implies$ 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.

2.2. Prima applicazione

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

Andiamo a definire:

- $\mathrm{X} = \{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$ insieme dei simboli sorgente;
- $c : \mathrm{X} \longrightarrow \{0, 1\}^+$ funzione di codifica;
- $c(\heartsuit) = 0$;
- $c(\diamondsuit) = 01$;
- $c(\clubsuit) = 010$;
- $c(\spadesuit) = 10$.

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

- ambiguità: se ricevo "010" ho come possibili traduzioni $\clubsuit$, $\heartsuit\spadesuit$ e $\diamondsuit\heartsuit$;
- pessima compressione: usiamo tre simboli di codice per codificare $\clubsuit$ e solo uno per codificare $\heartsuit$.

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 $x \in \mathrm{X}$.

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 $P_n \implies$ 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 $\mathbb{X} \implies$ variabile aleatoria $\mathbb{X} : \mathrm{X} \longrightarrow \mathbb{R}$ che rappresenta un'estrazione di un simbolo sorgente;
- insieme $\mathbb{D} \implies$ già definito in precedenza come $\{0, ..., D-1\}$;
- funzione $c \implies$ 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 $c(x) = 0$ oppure $c(x) = 1 \ \forall x \in \mathrm{X}$ avrei sì una codifica di lunghezza minima ma sarebbe impossibile da decodificare.

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

3. 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.

3.1. 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:
- $\mathrm{X} = \{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$ insieme dei simboli sorgente;
- $c : \mathrm{X} \longrightarrow \{0,1\}^+$ funzione di codifica;
- $c(\heartsuit) = 0$;
- $c(\diamondsuit) = 01$;
- $c(\clubsuit) = 010$;
- $c(\spadesuit) = 10$.

La codifica c scelta è sicuramente non singolare, però abbiamo delle problematiche a livello di decodifica: infatti, possiamo scrivere 01001 come $c(\clubsuit, \diamondsuit)$, $c(\diamondsuit, \heartsuit, \diamondsuit)$ o $c(\heartsuit, \spadesuit, \diamondsuit)$, e lato ricevente questo è un problema, perché "unendo" tutte 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 $L = \sum_{x \in \mathrm{X}} l_c(x)$

3.2. 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:
- $\mathrm{X} = \{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$ insieme dei simboli sorgente;
- $c : \mathrm{X} \longrightarrow \{0,1\}^+$ funzione di codifica;
- $c(\heartsuit) = 10$;
- $c(\diamondsuit) = 00$;
- $c(\clubsuit) = 11$;
- $c(\spadesuit) = 110$.

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 $\clubsuit\diamondsuit...\diamondsuit$;
- #0 dispari: decodifichiamo con $\spadesuit\diamondsuit...\diamondsuit$.

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.

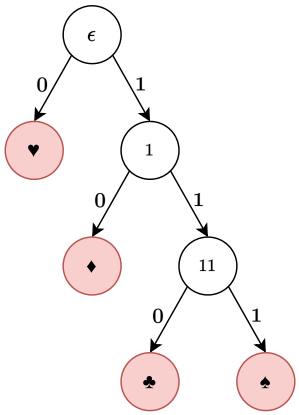
Un altro problema lo riscontriamo a livello di memoria: supponiamo che lato sorgente vengano codificati dei dati dell'ordine dei terabyte, per il ricevente sarà impossibile tenere in memoria una quantità simile di dati per fare la decodifica alla fine della ricezione.

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:
- $\mathrm{X} = \{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$ insieme dei simboli sorgente;
- $c : \mathrm{X} \longrightarrow \{0,1\}^+$ funzione di codifica;
- $c(\heartsuit) = 0$;
- $c(\diamondsuit) = 10$;
- $c(\clubsuit) = 110$;
- $c(\spadesuit) = 111$.

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.

3.3. 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 è non 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:

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

Una cosa che possiamo notare è come, aumentando di volta in volta le proprietà di un codice, e quindi passando di "classe" in "classe", 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.

4. Disuguaglianza di Kraft

4.1. Definizione

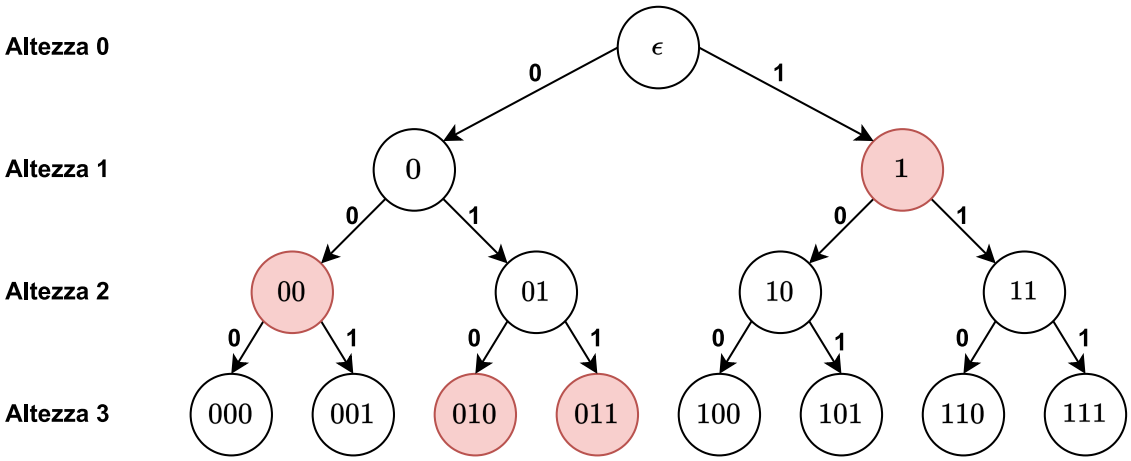
I codici istantanei soddisfano la **disuguaglianza di Kraft**, che ci permette, solo osservando le lunghezze delle parole di codice $l_c(x)$, di dire *se esiste* un codice istantaneo con quelle lunghezze.

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

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

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

4.2. Applicazione

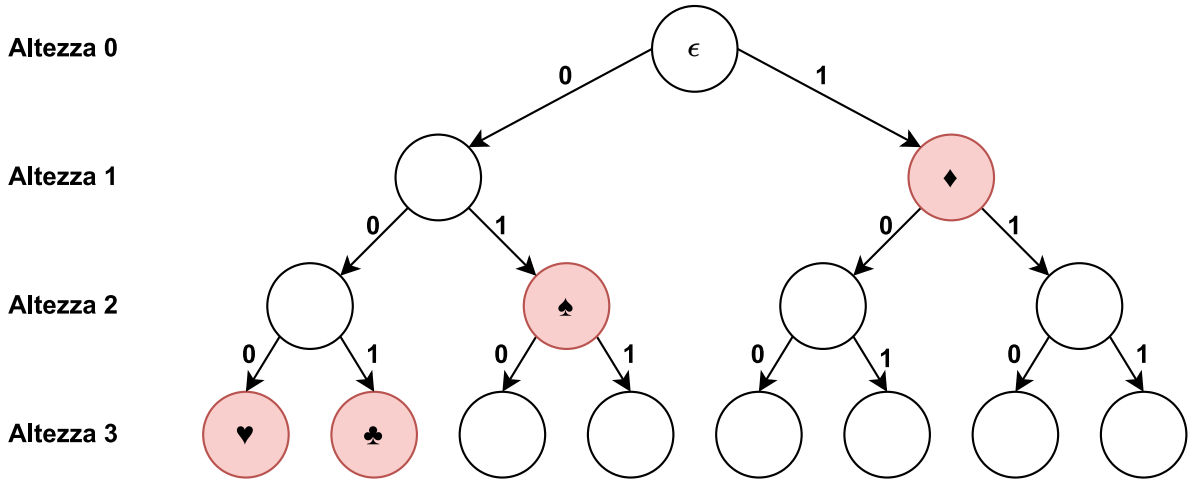
Andiamo a definire:

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

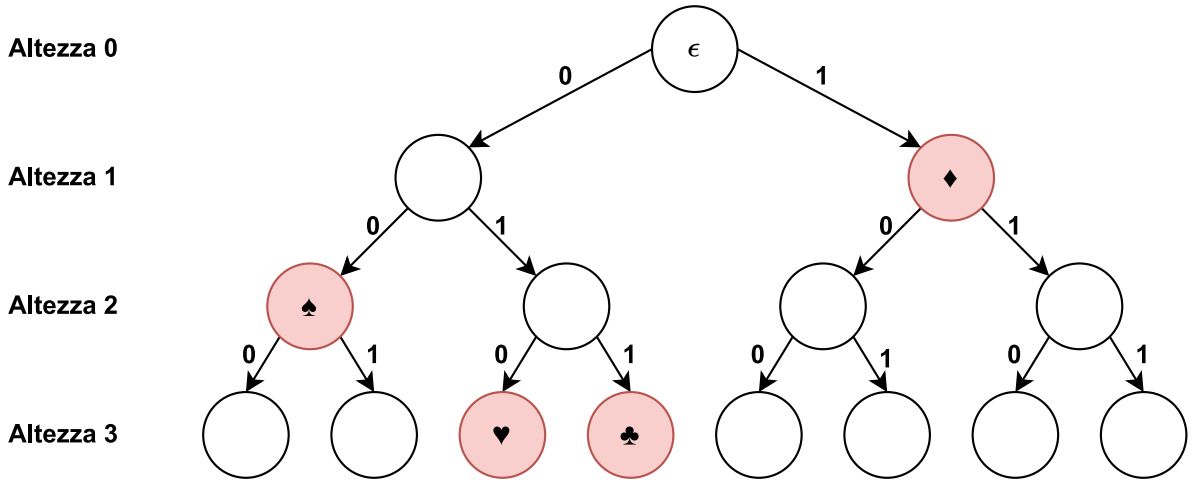
Vogliamo sapere se esiste un codice istantaneo, definito come c, aventi le lunghezze sopra definite: controlliamo quindi se esse soddisfano la disuguaglianza di Kraft.

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

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.

5. Codice di Shannon

5.1. Definizione

Abbiamo trovato un modo per verificare se un codice è istantaneo (osservare i prefissi) e un modo per verificare se una serie di 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 $D^{-l_i} \leq p_i \ \forall i = 1, ..., n$, 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”.

5.2. Esempi

5.2.1. 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.

5.2.2. 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 $\mathbb{E}[l_c] = \sum_{i=1}^{4} l_i p_i = 0 \cdot 1 + \underbrace{3 \cdot 0 \cdot +\infty}_{0 \text{ per } t \to 0^+} = 0$.

5.2.3. 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 $\mathbb{E}[l_c] = \sum_{i=1}^{4} l_i p_i = 2 \cdot \frac{1}{4} + 2 \cdot \frac{1}{4} + 2 \cdot \frac{1}{4} + 2 \cdot \frac{1}{4} = 2$.

5.3. 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.

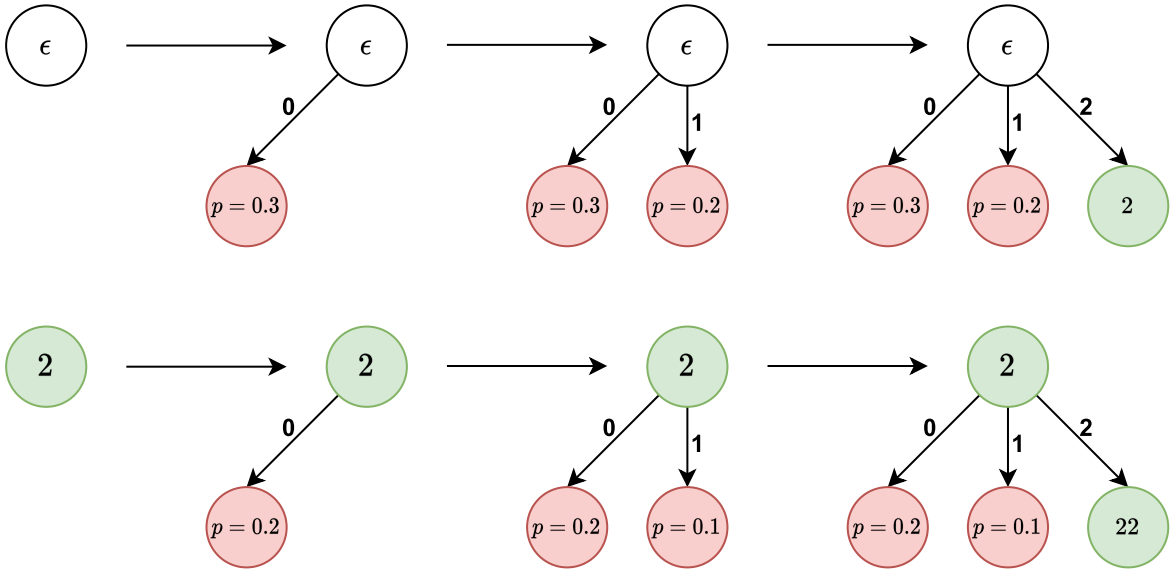
Questo mi va a dire che il codice è *compatto* e bilanciato, non spreco bit, mentre in altre situazioni ho un albero *sbilanciato* e potrei perdere dei bit.

6. 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.

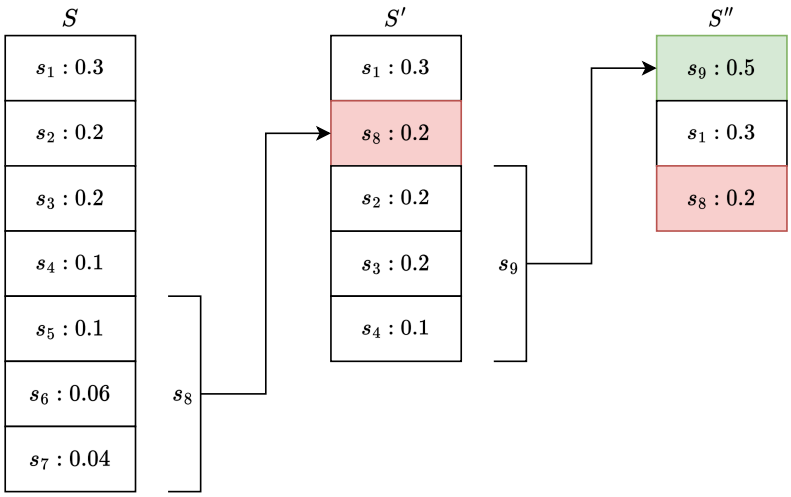
6.1. Definizione

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

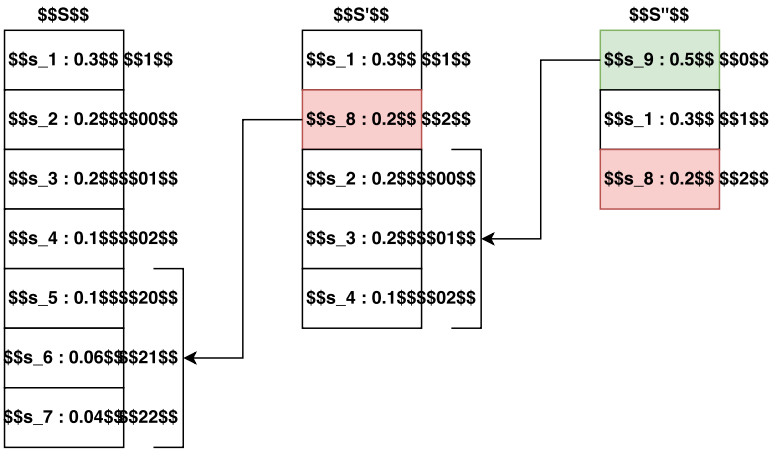
1. ordino le probabilità in ordine decrescente;
2. le ultime D probabilità sono sostituite dalla loro somma, e i simboli corrispondenti sono sostituiti da un simbolo "fantoccio", creando una nuova sorgente "fantoccia";
3. ripeto dal punto 1 fino a quando non si raggiungono t probabilità, con $t \leq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

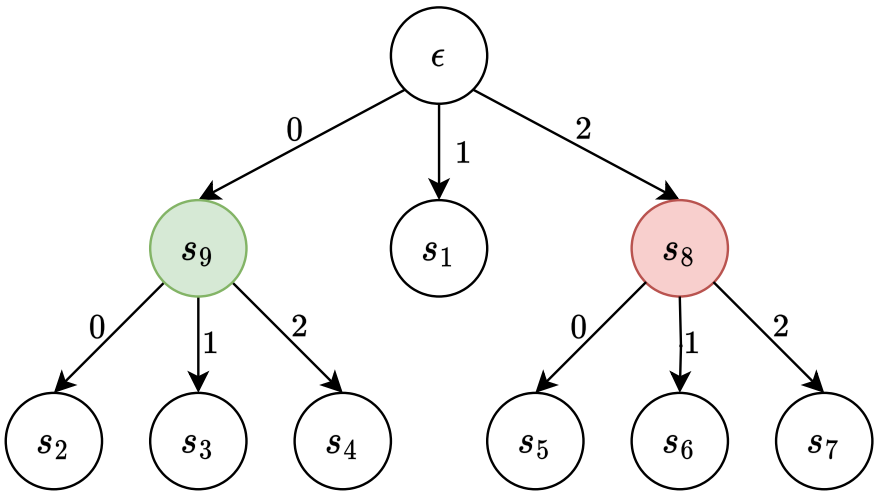
Supponiamo che la sorgente S emetta $n = 7$ simboli $s_1, ..., s_7$ con probabilità $P = \{0.3, 0.2, 0.2, 0.1, 0.1, 0.06, 0.04\}$, vogliamo costruire un codice ternario di Huffman.



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

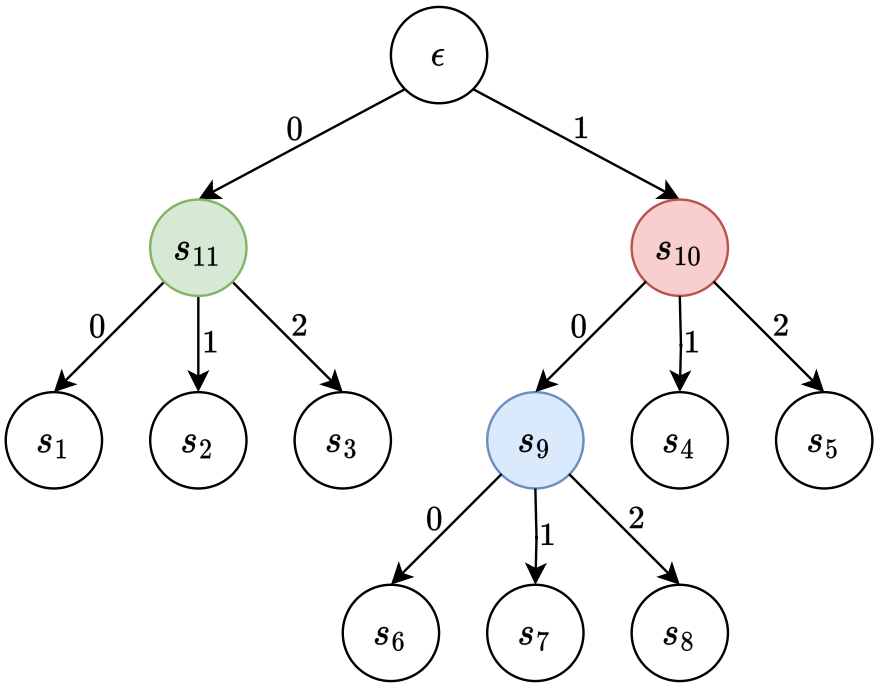


Nella seconda fase invece andiamo ad eseguire un "rollback" delle compressioni, sostituendo ad ogni nodo "compresso" le vecchie probabilità, andando quindi a costruire l'*albero* di codifica.



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:

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

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

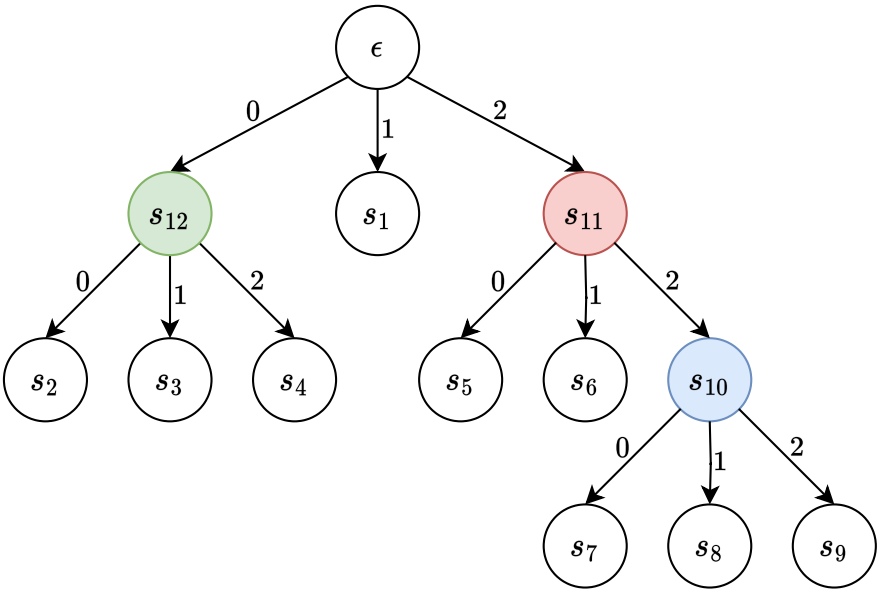
Cosa abbiamo ottenuto? Ricordando che ■ $= n - t \cdot (D-1)$, abbiamo fatto vedere che:

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

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.

7. Entropia

Dato il modello sorgente X, p, con $\mathrm{X} = \{x_1, ..., x_m\}$ e $P = \{p_1, ..., p_m\}$, anche $\mathbb{X} : \mathrm{X} \longrightarrow \mathbb{R} = \{a_1, ..., a_m\}$(sottoinsieme), $P(\mathbb{X} = a_i) = p_i$

Definisco

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

Se non metto pedici è entropia binaria $D = 2$

Entropia fa solo riferimento alla distribuzione di probabilità, non c'entra niente con i simboli

Se volessi fare cambio base (quindi cambiare l'entropia binaria) devo ricordarmi che

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

Ovvero cambiare base significa cambiare base e portarsi dietro una costante

Quindi

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

Proviamo a graficare l'entropia binaria con una bernoulliana

$p(\mathbb{X} = 1) = p$ e $p(\mathbb{X} = 0) = 1 - p$, sviluppa entropia e vedi che in 0 e 1 l'entropia vale 0

In $\frac{1}{2}$ ho la massima entropia e vale 1

Aggiungi grafico

So inoltre che

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

Questi bound ci serviranno dopo (aggiungi grafico)

Che proprietà ha l'entropia?

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

Sia $\mathbb{X}$ variabile casuale che assume i valori distinti $a_1, ..., a_m$; allora $H_D(\mathrm{X}) \leq \log_D m \ \forall D > 1$; inoltre, vale uguaglianza se e solo se $\mathbb{X}$ ha una distribuzione uniforme su $a_1, ..., a_m$

DIMOSTRAZIONE (voglio far vedere ≤ 0)

$$\begin{aligned} H_D(\mathrm{X}) - \log_D m &= \sum_{i=1}^{m} p_i \log_D \frac{1}{p_i} - \log_D m \cdot \underbrace{\sum_{i=1}^{m} p_i}_{=1} \\ &= \sum_{i=1}^{m} p_i \log_D \frac{1}{p_i} - p_i \log_D m = \sum_{i=1}^{m} p_i \cdot \left(\log_D \frac{1}{p_i} - \log_D m \right) \\ &= \sum_{i=1}^{m} p_i \cdot \left(\log_D \frac{1}{p_i} + \log_D m^{-1} \right) = \sum_{i=1}^{m} p_i \cdot \log_D \frac{m^{-1}}{p_i} (\text{sistema}) \end{aligned}$$

Io so la maggiorazione del logaritmo

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

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

Quindi $H_D(\mathrm{X}) - \log_D m \leq 0 \longrightarrow H_D(\mathrm{X}) \leq \log_D m$

Ora, se $P(X = a_i) = \frac{1}{m} \ \forall i = 1, ..., m$ allora

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

FINE DIMOSTRAZIONE

Introduciamo l'entropia relativa $\Delta(\mathrm{X} \parallel Y)$ misura la distanza tra X e Y, la diversità tra X e Y, due distribuzioni di probabilità

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

S è 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

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

FINE DIMOSTRAZIONE

Ora vediamo relazione tra il valore atteso delle lunghezze del codice e l'entropia

Teorema: sia $c : \mathrm{X} \to \mathbb{D}^+$ 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

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

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

FINE DIMOSTRAZIONE

8. Sardinas-Patterson

Algoritmo che permette di dimostrare se un codice è univocamente decodificabile

Inserisci due esempi risolti a mano

Sia $S_1 = \mathrm{X}$, allora proseguo in modo iterativo applicando le due regole seguenti:

- per ogni $x \in S_1$, se esiste $y \in S_i$ tale che $xy \in S_i$ allora $y \in S_{i+1}$
- per ogni $z \in S_i$, se esiste $y \in S_1$ tale che $xy \in S_1$ allora $y \in S_{i+1}$

Inserisci secondo esempio risolto con Sardinas-Patterson

9. Primo teorema di Shannon

Il prossimo risultato rivela che i codici di Shannon forniscono una descrizione delle realizzazioni di X di lunghezza media quasi ottimale rispetto a tutti i codici istantanei.

9.1. Introduzione

Lemma Per ogni sorgente $\langle \mathrm{X}, p \rangle$ con $\mathrm{X} = \{x_1, ..., x_m\}$ e $p = \{p_1, ..., p_n\}$. Dato il codice istantaneo di Shannon c con lunghezze $l_i = l_c(x_i)$ tali che $l_i = \left\lceil \log_D \frac{1}{p_i} \right\rceil$ per $i = 1, ..., m$, vale

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

Quindi con questo teorema, Shannon capisce che il suo codice paga al più 1 bit in più di informazione aggiuntivo.

Dimostrazione

$$\mathbb{E}[l_c] = \sum_{i=1}^{n} p_i \left\lceil \log_D \frac{1}{p_i} \right\rceil < \sum_{i=1}^{n} p_i \left(\log_D \frac{1}{p_i} + 1 \right)$$

$$= H_{D(\mathrm{X})} + 1$$

□

Quindi, combinando questo risultato con $\mathbb{E}[l_i] \geq H_D(\mathrm{X})$ *che vale per ogni codice istantaneo* c, otteniamo che il codice di Shannon utilizza un numero di bit che è compreso tra l'entropia e una variabile additiva di 1 bit, in altre parole si avvicina al codice teorico migliore (entropia) sprecando nel caso peggiore un simbolo in più del necessario.

Tuttavia sorge un problema, perché l'inefficienza dei codici di Shannon cresce linearmente con la lunghezza del messaggio da codificare. Infatti, la lunghezza della codifica di Shannon di un messaggio $(x_1, ..., x_n)$, generato con n estrazioni da una sorgente $\langle \mathrm{X}, p \rangle$, è pari a

$$\sum_{i=1}^{n} \left\lceil \log_D \frac{1}{p(x_i)} \right\rceil$$

Per risovere questa situazione possiamo usare una tecnica nota come **codifica a blocchi**

9.2. Codifica a blocchi

9.2.1. Definizione

La codifica a blocchi suddivide ogni messaggio in blocchi di simboli sorgente dove ogni blocco ha la stessa lunghezza. I blocchi vengono quindi codificati con un codice per la sorgente i cui simboli sono tutti i possibili blocchi di lunghezza data. Quindi a partire dalla nostra sorgente $\langle \mathrm{X}, p \rangle$ avendo $C : \mathrm{X} \to D^+$ *estensione del codice*, vale che $l_c(x_1, ..., x_n) = \sum_{i=1}^{n} \left\lceil \log_D \frac{1}{p_i} \right\rceil = \log_D \left\lceil \frac{1}{\prod_{i=1}^{n} p(x_i)} \right\rceil$ quindi in $l_c(x_1, ..., x_n) = \log_D \left\lceil \frac{1}{P(x_i, ..., x_n)} \right\rceil$, Questo definisce il modello $\langle \mathrm{X}^n, P_n \rangle$ con codice $C_n : \mathrm{X}^n \to D^n$, dove X^n è l'insieme di n-uple $(x_1, ..., x_n)$ di simboli di X e P_n è la distribuzione su X^n associata a n estrazioni identicamente distribuite secondo p.

Quindi Shannon sta spostando il problema alla sorgente, da $\langle \mathrm{X}, p \rangle$ a $\langle \mathrm{X}^n, P_n \rangle$.

9.2.2. Entropia su P_n

Sia X una variabile casuale con distribuzione p,lavorando con la codifica a blocchi abbiamo $\mathrm{X}_1, ..., \mathrm{X}_n$ variabili casuali anch'esse con distribuzione p.

Abbiamo che $P_n(x_1, ..., x_n) = \prod_{i=1}^{n} p(x_i)$ definiamo l'entropia

$$H(\mathrm{X}_1, ..., \mathrm{X}_n) = \sum_{x_1,...,x_n} P_n(x_1, ..., x_n) \log_2 \frac{1}{P_n(x_1, ..., x_n)}$$

notiamo che il primo termine della moltiplicazione è un produttoria e il secondo possiamo ridurlo a $\sum_{i=1}^{n} log_2 \frac{1}{p(x_i)}$, quindi ricavo che

$$H(\mathrm{X}_1, ..., \mathrm{X}_n) = \sum_{x_1}, ..., \sum_{x_n} \left(\prod_{i=1}^{n} p(x_i) \right) \sum_{i=1}^{n} \log_2 \frac{1}{p(x_i)}$$

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

$$= nH(\mathrm{X})$$

Quindi sto sommando la stessa entropia, cioè l'entropia della sorgente a blocchi di n simboli è n volte l'entropia della sorgente base.

Siamo ora pronti per enunciare e dimostrare il vero primo teoreama di Shannon o *source coding*.

9.3. Teorema (Primo teorema di Shannon)

Teorema *(Primo teorema di Shannon)* Sia $C_n : \mathrm{X}^n \to D^+$ un codice di Shannon D-ario a blocchi per la sorgente $\langle \mathrm{X}, p \rangle$, ossia $l_c(x_1, ..., x_n) = \left\lceil \log \frac{1}{P(x_1,...,x_n)} \right\rceil$ allora

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

Questo risultato dimostra che se codifichiamo con Shannon a blocchi, allora la lunghezza media della parola di codice per simbolo sorgente è asintotica all'entropia quando la lunghezza del blocco cresce all'infinito, *in altre parole il codice di Shannon diventa asintoticamente ottimo.*

Dimostrazione Osserviamo che vale

$$H_D(x_1, ..., x_n) \leq \mathbb{E}[l_c] \leq H_D(x_1, ..., x_n) + 1$$

$$= nH_D(x) \leq \mathbb{E}[l_c] \leq nH_D(x) + 1$$

Dividendo entrambi i mebri per n otteniamo

$$H_D(x) \leq \mathbb{E}[l_c] \leq H_D(x) + \frac{1}{n}$$

Da quest'ultima relazione ricaviamo l'enunciato del teorema. □

Quindi Shannon dimostra che in assenza di rumore, codificando a blocchi $\mathbb{E}$ si "schiaccia" verso l'entropia, *quindi più grande è il blocco più grande è il vantaggio.*

9.4. Significato operativo all'entropia relativa

Un fatto che per ora non abbiamo mai considerato è che in realtà il modello teorico di Shannon $\langle \mathrm{X}, p \rangle$ è un modello (troppo) ideale, nella realtà p non la conosciamo (nella maggior parte dei casi), quindi è necessario fare una stima usando un altro modello $\langle Y, q \rangle$ che simula il modello X. Quindi ora che abbiamo associato l'entropia alla lunghezza minima media di codici istantanei, diamo un analogo significato operativo all'entropia relativa, mostrando che essa corrisponde alla differenza fra

la lunghezza media di un codice di Shannon costruito sul modello di sorgente X e quello di un codice di Shannon costruito su un diverso modello di sorgente Y.

Teorema 9.4.1 Dato un modello sorgente $\langle \mathrm{X}, p \rangle$, se $c : \mathrm{X} \to D^+$ è un codice di Shannon con lunghezze $l_c(x) = \left\lceil \frac{1}{q(x)} \right\rceil$, dove q è una distribuzione arbitraria su X, allora

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

dove X ha distribuzione p e Y ha distribuzione q.

Dimostrazione Osserviamo che

$$\mathbb{E}[l_c] = \sum_{x \in \mathrm{X}} p(x) \left\lceil \log_D \frac{1}{q(x)} \right\rceil < \sum_{x \in \mathrm{X}} p(x) \log_D \frac{1}{q(x)} + 1$$

$$= \sum_{x \in \mathrm{X}} p(x) \log_D \left(\frac{1}{q(x)} \frac{p(x)}{p(x)} \right) + 1 = \sum_{x \in \mathrm{X}} p(x) \log_D \left(\frac{p(x)}{p(x)} \frac{1}{p(x)} \right) + 1$$

$$= \sum_{x \in \mathrm{X}} p(x) \log_D \frac{p(x)}{q(x)} + \sum_{x \in \mathrm{X}} p(x) \log_D \frac{1}{p(x)} + 1$$

$$= D(\mathrm{X} \parallel Y) + H_D(\mathrm{X}) + 1$$

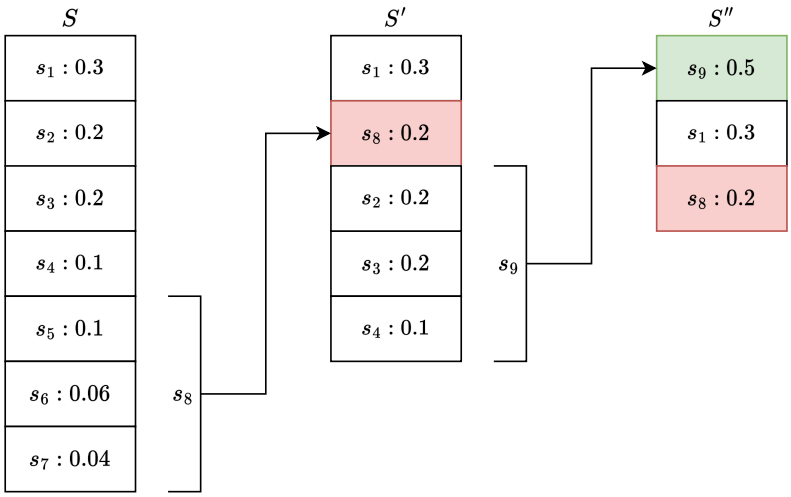
□

10. Ottimalità del codice di Huffman

10.1. Introduzione

Ricordiamo che un codice di Huffman è costruito con il seguente sistema (Algoritmo di Huffman):

1. i simboli sorgente vengono ordinati in base alle probabilità;
2. si crea un nuovo modello di sorgente in cui i D simboli meno frequenti sono rimpiazzati da un nuovo simbolo con probabilità pari alla somma delle loro probabilità;
3. se la nuova sorgente contiene più di D simboli si ricomincia dal passo 1.



$|\mathrm{X}| = m$

$m = (D-1)K + 1$, quindi è divisibile per $D-1$ con resto 1.

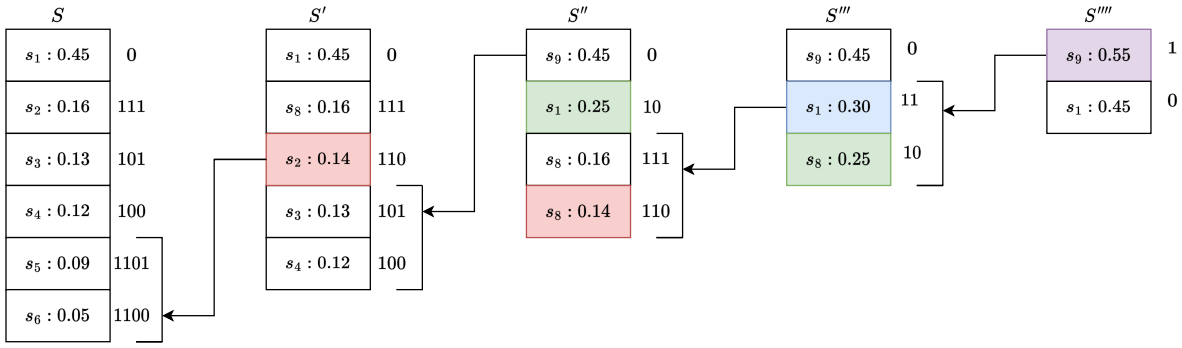
Procediamo ora a dimostrare l'ottimalità del codice di Huffman nell'ambito dei codici sorgente istantanei. Prima di dimostrare il teorema, dobbiamo però fare una semplice osservazione preliminare. Ovvero: da un codice di Huffman *D-ario* per una sorgente di $m - D + 1$ simboli possiamo ricavare un codice di Huffman D-ario per una sorgente di m simboli semplicemente sostituendo un simbolo sorgente con D nuovi simboli cosicché le probabilità assegnate ad essi siano tutte più piccole di quelle dei rimanenti m – D vecchi simboli.

10.2. Lemma sulla generazione con giustapposizione

Lemma Sia c' un codice *D-ario* di Huffman per la sorgente $\mathrm{X}' = \{x_1, ..., x_{m-D+1}\}$ con probabilità $p_1 \geq ... \geq p_{m-D+1}$. Sia X la sorgente di m simboli $\{x_1, ..., x_{k-1}, x_{k+1}, ..., x_{m+1}\}$ ottenuta da X′ togliendo x_k e aggiungendo D nuovi simboli $x_{m-D+2}, ..., x_{m+1}$ con probabilità $p_{m-D+2}, ..., p_{m+1}$ tali che $0 < \underbrace{p_{m-D+2}, ..., p_{m+1}}_{\text{nuove prob}} < p_{m-D+1}$ e $p_{m-D+2} + ... + p_{m+1} = p_k$. Allora il codice

$$c(x) = \begin{cases} c'(x) & \text{se } x \in \{x_1, ..., x_{k-1}, x_{k+1}, ..., x_{m-D+1}\} \\ c'(x_k)i & \text{se } x = x_{m-D+i+2} \text{ per } i = 0, ..., D-1. \end{cases}$$

è un codice di Huffman per la sorgente X.



In sostanza sto dicendo che se $c(x)$ è uguale a $c'(x)$ fino a x_k escluso e dopo aggiunge i nuovi simboli giustapponendoli a x_k (nota che $p_k = p_{\text{simbolo1}} + p_{\text{simbolo2}} + \dots$ in base a D). In pratica è come fare Huffman al contrario, quindi srotolando i simboli "fantoccio" che costituivano la somma delle D probabilità più basse.

Dimostrazione La dimostrazione è ovvia considerando che dopo il primo passo nella costruzione del codice di Huffman per X otteniamo X′ come nuova sorgente. Quindi i due codici differiscono solo per le codifiche ai D simboli $x_{m-D+2}, \dots, x_{m+1}$ che sono quelli meno probabili in X. Per definizione dell'algoritmo di Huffman, le codifiche dei simboli meno probabili di X sono definite in termini del codice di Huffman per X′ esattamente come descritto nel sistema precedente. □

10.3. Teorema sull'ottimalità del codice di Huffman

Teorema *(Ottimalità del codice di Huffman)* Data una sorgente $\langle \mathrm{X}, p \rangle$ con $D > 1$, il codice *D-ario* c di Huffman minimizza $\mathbb{E}[l_c]$ fra tutti i codici *D-ari* istantanei per la medesima sorgente.

Quindi: $\mathbb{E}[l_c] \leq \mathbb{E}\left[l_{c',c'',\dots}\right]$

Dimostrazione La dimostrazione procede per induzione.

Passo base: (per facilità nei conti consideriamo D = 2 e si ricorda che $|\mathrm{X}| = m$). Nel caso base $m = 2$ Huffman è ottimo. Infatti, intuitivamente mi basta osservare che l'algoritmo di Huffman produce il codice $c(x_1) = 0$ e $c(x_2) = 1$ che è ottimale per ogni distribuzione di probabilità su x_1, x_2. Passo induttivo: Assumendo quindi $m > 2$, grazie all'ipotesi induttiva abbiamo che Huffman è ottimo per $k \leq m - 1$. Fissiamo una sorgente $\langle \mathrm{X}, p \rangle$ e siano $u, v \in \mathrm{X}$ tale che $p(u)$ e $p(v)$ sono minime (quindi le ultime che ordineremmo nell'algoritmo di Huffman). Definiamo la sorgente $\langle \mathrm{X}', p' \rangle$ dove $u, v \in \mathrm{X}$ sono rimpiazzati da $z \in \mathrm{X}'$ e dove

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

Sia c' il codice di Huffman per la sorgente $\langle \mathrm{X}', p' \rangle$. Dato che $|\mathrm{X}'| = m - 1$, c' è ottimo per ipotesi induttiva (Huffman è ottimo per $k \leq m - 1$).

Definiamo ora il codice $c(x)$ per X.

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

nota che c'(x)0 e c'(x)1 sono costruiti con una giustapposizione.

Per il lemma precedente sappiamo che c è un codice istantaneo di Huffman. Ora vogliamo dimostrare che c è ottimale (quindi che $\mathbb{E}$ <= di qualsiasi codice istantaneo).

Per fare ciò abbiamo bisogno di dimostrare tre relazioni, che in seguito ci permetteranno di trarre le dovute conclusioni.

$$\mathbb{E}[l_c] = \sum_{x \in \mathrm{X}} l_c(x)p(x) = \sum_{x \in \mathrm{X}} \underbrace{l_{c'}(x)p'(x)}_{\text{identico a c per ora}} \underbrace{- l_{c'}(z)p'(z)}_{\text{tolgo z}} + \underbrace{l_c(u)p(u) + l_{c(v)}p(v)}_{\text{aggiungo u e v}}$$

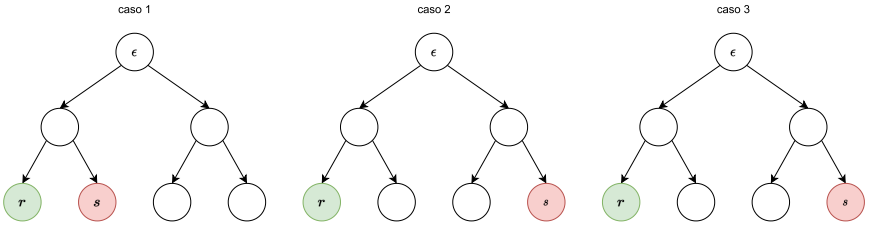
$$= \mathbb{E}[l_{c'}] - l_{c'}(z)p'(z) + (l_{c'}(z) + 1)p(u) + (l_{c'}(z) + 1)p(v)$$

$$= \mathbb{E}[l_{c'}] - l_{c'}(z)p'(z) + l_{c'}(z)p'(z) + p'(z)$$

$$= \mathbb{E}[l_{c'}] + p'(z)$$

$$\underline{\text{Quindi } \mathbb{E}[l_c] = \mathbb{E}[l_{c'}] + p'(z)}$$

Ora consideriamo un altro codice istantaneo c_2 per la medesima sorgente $\langle \mathrm{X}, p \rangle$ e verifichiamo sempre che $\mathbb{E}[l_c] \leq \mathbb{E}\left[l_{c_2}\right]$. Fissato c_2, siano $r, s \in \mathrm{X}$ tali che $l_{c_2}(r)$ e $l_{c_2}(s)$ sono massime.



Esaminando le foglie r e s nell'albero di codifica di c_2, osserviamo che che senza perdita di generalità, possiamo assumere che c_2 sia tale che r e s sono fratelli. Infatti se r e s sono fratelli, non facciamo nulla. Se r o s hanno un fatello (ad esempio il fratello di r è f), allora possiamo scegliere r e f tali che $l_{c_2}(r)$ e $l_{c_2}(f)$ sono massime invece di r e s (tanto se f è fratello di s allora hanno la stessa l_{c_2}). Se invece né r ne s hanno un fratello nell'albero, allora possiamo sostituire alla codifica di ciascun la codifica del padre finché ci riportiamo nella situazione in cui r e s hanno entrambi un fratello. Ora trasformiamo c_2 in un codice

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

In pratica quello che fa $\tilde{c}_2$ è sostituire la codifica dei simboli di lunghezza massima (r e s) con quella dei simboli di lunghezza minima (u e v). Ora dobbiamo capire quele codice tra c_2 e $\tilde{c}_2$ "sfida" meglio c (sull'ottimalità) esaminando la differenza fra la lunghezza media di questi ultimi.

$$\mathbb{E}\left[l_{\tilde{c}_2}\right] - \mathbb{E}\left[l_{c_2}\right] = \sum_{x \in \mathrm{X}} p(x)\left(l_{\tilde{c}_2}(x) - l_{c_2}(x)\right)$$

$$= p(r)l_{c_2}(u) + p(u)l_{c_2}(r) + p(s)l_{c_2}(v) + p(v)l_{c_2}(s) -$$

$$-p(u)l_{c_2}(u) - p(r)l_{c_2}(r) - p(v)l_{c_2}(v) - p(s)l_{c_2}(s)$$

$$= \underbrace{\underbrace{(p(r) - p(u))}_{\geq 0}\underbrace{\left(l_{c_2}(u) - l_{c_2}(r)\right)}_{\leq 0} + \underbrace{(p(s) - p(v))}_{\geq 0}\underbrace{\left(l_{c_2}(v) - l_{c_2}(s)\right)}_{\leq 0}}_{\leq 0}$$

I segni delle differenze sono determinati dalla scelta di u, v, r, s in quanto

$$\max\{p(u), p(v)\} \leq \min\{p(r), p(s)\}, \quad \min\left\{l_{c_2}(r), l_{c_2}(s)\right\} \geq \max\left\{L_{c_2}(u), l_{c_2}(v)\right\}$$

Quindi abbiamo dimostrato che $\underline{\mathbb{E}\left[l_{\tilde{c}_2}\right] \leq \mathbb{E}\left[l_{c_2}\right]}$ (lo "sfidante" migliore è $\tilde{c}_2$). □