Capitolo 2

L'ENTROPIA NELLA COMPRESSIONE DEI DATI

I. I CODICI DI HUFFMAN

Che la frequenza caratteristica delle lettere fosse diversa venne ben presto scoperto dai tipografi, cui non poteva sfuggire che i tipi della E si logoravono molto prima dei tipi della Z, per cui bisognava tenerne di più nella provvista: quanti di più? Del resto, nella decifrazione delle scritture segrete già gli Arabi del IX secolo erano giunti a un'accurata descrizione statistica della lingua del Corano. Le leggi statistiche furono poste a buon uso da Samuel Morse, che nel 1836 propose un codice basato sul principio: tanto più frequente la lettera, tanto più breve la sua parola di codice. Ciò costringe a usare parole di codice lunghe per le lettere rare, ma i tempi di trasmissione e/o lo spazio di memoria occupato vengono comunque ridotti: oggi diremmo che il *codice Morse* mira alla *compressione dei dati*. Questo codice ormai storico è *ternario*: i simboli di codifica sono il *tratto*, il *punto* e la *virgola*; quest'ultima viene scritta alla fine di ciascuna parola di codice per consentire la corretta *segmentazione* del flusso di simboli ricevuti dal destinatario. Il codice di Huffman, se si vuole, è il perfezionamento "scientifico" del codice di Morse.

1. Trascrizione dei dati

E' dato un testo scritto sull'alfabeto *primario* $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$; dobbiamo *trascriverlo* utilizzando le β lettere (i β simboli) dell'alfabeto *secondario* $\mathcal{B} = \{b_1, b_2, \ldots, b_\beta\}$; in pratica spesso $\mathcal{B} = \{0, 1\}$ è binario. La trascrizione dev'essere *attendibile* (*reliable*): a partire dal flusso di simboli secondari, che va *decodificato*, il destinatario deve poter univocamente ricostruire l'informazione primaria; sceglieremo un criterio di attendibilità severo, che proibisce gli errori di decodifica, anche quelli poco probabili. Per ragioni di complessità (da evitare) considereremo solo codici *giustappositivi*, che trascrivono un testo giustapponendo le trascrizioni (le *parole di codice*) delle singole lettere che lo compongono.

Attenzione: fra le "lettere" in questione ci può ben essere la spaziatura, il ritorno carrello, ecc.

Definizioni. Un *codice* è un'applicazione di $\mathcal{A}^*$ in $\mathcal{B}^*$. Se l'applicazione è iniettiva il codice si dice *univocamente decodificabile*, o UD.

Rammentiamo che $\mathcal{A}^*$ è l'insieme formato da tutte le stringhe costruite su $\mathcal{A}$, inclusa quella di lunghezza zero. In termini solenni un codice giustappositivo è un omomorfismo fra due *monoidi liberi*, quello primario e quello secondario, strutturati con l'operazione algebrica di giustapposizione. I nostri codici, salvo esplicita specifica contraria, saranno tutti giustappositivi, e dunque si possono assegnare specificando le K parole di codice associate alle lettere primarie. In un codice UD sono escluse parole di codice ripetute ed è esclusa la parola di codice di lunghezza 0.

E' utile disegnare gli *alberi di codice* (radice a sinistra, 0 = in alto, 1 = in basso) associati ai due codici binari $\mathcal{C}_1 = \{0, 10, 11\}$ e $\mathcal{C}_2 = \{1, 10, 100\}$ che codificano le tre lettere primarie $\{a, b, c\}$; $K = 3$, $\beta = 2$. Gli alberi di codice possono essere usati in fase di decodifica: il decodificatore, "istruito" dai β-it della codifica, percorre l'albero di codice dalla radice verso l'esterno. Nel caso di $\mathcal{C}_1$ le lettere di $\mathcal{A}$ etichettano soltanto foglie: ciò consente al decodificatore di riconoscere una parola di codice non appena essa è terminata. Nel caso di $\mathcal{C}_2$ l'univoca decodificabilità è assicurata dal ruolo che ha il bit 1 di "delimitatore iniziale" delle parole di codice; nella sua ricerca del delimitatore, purtroppo, il decodificatore può incorrere in *ritardi di decodifica*; ad esempio, per segmentare 1010100 in 10|10|100 (che poi decodifica a *bbc*), deve per due volte superare il nodo giusto, quello di b, controllare che non ci sia lo 0, e poi tornare indietro. E' un fatto generale: se nell'albero di codifica c'è una lettera di $\mathcal{A}$ che etichetta un nodo non terminale, i ritardi di decodifica sono inevitabili ($\Box$! giunto al nodo interno, il decodificatore *non* non ha ancora modo di sapere se ciò che ha appena letto è la parola di codice dell'etichetta di quel nodo, oppure il prefisso di una parola di codice più lunga). La definizione seguente identifica, all'interno della classe dei codici univocamente decodificabili, l'importante sottoclasse dei codici *istantaneamente decodificabili*:

Definizione. Un codice *a prefisso* è un codice in cui due parole di codice non possono essere l'una il prefisso dell'altra.

La proprietà del prefisso, a dire il vero, si dovrebbe chiamare proprietà del non prefisso, poichè ne predica l'assenza. Pensando agli alberi, un nome carino sarebbe "codice a foglie".

Esempio sgradevole. Il codice $\mathcal{C}_3 = \{1, 00, 10\}$, che per inciso ha le stesse lunghezze di $\mathcal{C}_1$, è UD; purtroppo il suo ritardo di decodifica è *infinito*! (Così dicendo, intendiamo che i ritardi di decodifica che si possono via via presentare sono superiormente illimitati; $\Box$! in decodifica è la parità della tratta di zeri che permette di scegliere fra le segmentazioni $1|00|00|00|\ldots$ e $10|00|00|00\ldots$, bisogna dunque contare tutti gli zeri fino al prossimo 1 e poi tornare indietro.)

Esempio storico. Il codice $a \to 110$, $b \to 10$, $c \to 0$ è UD (lo 0 funziona da "delimitatore" a destra delle parole di codice). Se i tre simboli primari sono interpretati come $a =$ *tratto*, $b =$ *punto* e $c =$ *virgola*, e i due simboli secondari come $1 =$ *impulso di corrente* di durata unitaria, o *pieno*, e $0 =$ *vuoto di corrente* di durata unitaria, si ottiene la realizzazione circuitale del codice Morse (l'unità di tempo è dell'ordine di grandezza dei centesimi di secondo). Insomma il "vero" codice Morse, quello che vede la macchina, è binario e non ternario. A proposito: la composizione "in serie" di due codici UD è ancora UD ($\Box$!).

2. La disuguaglianza di Kraft

Nella costruzione dei codici dobbiamo contemperare le esigenze conflittuali dell'attendibilità e dell'economicità. E' troppo facile assegnare le parole di codice se non c'è il vincolo dell'economicità, ma solo quello dell'attendibilità. Nel seguito identificheremo economicità e brevità , ciò che è di solito molto naturale. La disuguaglianza di Kraft risponde in maniera del tutto esauriente alla domanda: quanto brevi possono essere le parole di un codice UD? Partiamo dal teorema *inverso*, "normativo" (ossia limitante, che esclude lunghezze troppo brevi); poi passeremo al teorema *diretto*, propositivo.

Teorema inverso. In un codice UD β-ario che codifica K lettere vale la disuguaglianza di Kraft:

$$\sum_{1\le i\le K} \beta^{-l_i} \le 1$$

Dimostrazione. Indichiamo con l_x la lunghezza in β-it del "parolone di codice" (della l_x-upla secondaria) che codifica la n-upla primaria x; se λ è la massima lunghezza delle K parole di codice che codificano le lettere primarie ($\lambda = \max l_i$), $n\lambda$ è la massima lunghezza dei paroloni di codice che codificano n-uple ($n\lambda = \max_{x\in\mathcal{A}^n} l_x$). Come mostra una breve riflessione:

$$(\sum_{1\le i\le K} \beta^{-l_i})^n \equiv (\sum_{a\in\mathcal{A}} \beta^{-l_a})^n = \sum_{x\in\mathcal{A}^n} \beta^{-l_x}$$

Nella somma a destra raggrupperemo gli addendi uguali: sotto $N(n,l)$ indica appunto il numero ≥ 0 di n-uple primarie il cui parolone di codice è lungo l; $N(n,l)=0$ per $l > n\lambda$. Se il codice è UD $N(n,l) \le \beta^l$, poichè β^l è il numero totale delle l-uple secondarie, e non ci possono essere due n-uple primarie codificate con la stessa l-upla. Allora:

$$(\sum_{1\le i\le K} \beta^{-l_i})^n = \sum_{1\le l\le n\lambda} N(n,l)\beta^{-l} \le \sum_{1\le l\le n\lambda} 1 = n\lambda$$

Ciò vale per n "arbitrariamente grande", ma un'andamento esponenziale viene asintoticamente battuta da una crescita lineare sse la base dell'esponente è ≤ 1. $\square$

La dimostrazione è ingegnosa: è asintotica, ma di asintotico, in ciò che abbiamo dimostrato, non c'è un bel niente.

Assumiamo ora un punto di vista propositivo. E' dato un albero β-ario *zeppo*, o *saturo*, di profondità, o altezza, l, con β^l foglie. Il problema è inserire a partire dalla radice un sottoalbero con K foglie $a_1, a_2, \dots, a_K$ di profondità $l_1 \le l_2 \le \dots \le l_K = l$. Pensando alla figura, le foglie dell'albero zeppo sono ordinate dall'alto verso il basso; l'algoritmo punta sempre alla foglia libera più alta; all'inizio tutte le foglie sono libere.

Algoritmo di inserimento.

Passo j, $1 \le j < K$. Inserisci a_j nel nodo di profondità l_j. Cancella il sottoalbero radicato in a_j; ciò comporta la perdita di β^{l-l_j} foglie a profondità l. Se ci sono ancora foglie libere prosegui, altrimenti esci e dichiara fallimento.
...
Passo K. Inserisci a_K come nel passo generico. Dichiara successo.

Com'è evidente, l'algoritmo di inserimento ha successo sse

$$\sum_{1\le j\le K} \beta^{l-l_j} \le \beta^l$$

ossia (dividi per β^l) sse le lunghezze $l_1, l_2, \ldots, l_K$ rispettano la disuguaglianza di Kraft. E dunque:

Teorema diretto. Se le lunghezze $l_1, l_2, \ldots, l_K$ verificano la disuguaglianza di Kraft esiste un codice a prefisso con parole di codice di quelle lunghezze.

Corollario. Un codice UD con ritardi di decodifica può venir sostituto da un codice istantaneamente decodificabile che ha le stesse lunghezze lettera per lettera.

E dunque lo stesso costo: i ritardi di decodifica non "pagano", non c'è nessuna contropartita, perlomeno nel modello di codifica che stiamo approfondendo, in cui costi e lunghezze vengono identificati. Si rifletta alla portata pratica del risultato.

Esercizio. Da' l'esempio di un codice inattendibile le cui lunghezze verificano la disuguaglianza di Kraft (e dunque sono usate male).

Di solito si dice che la disuguaglianza di Kraft è la condizione necessaria e sufficiente perchè valga l'univoca decodificabilità: è sostanzialmente vero, ma non alla lettera, come mostra l'esercizio appena proposto.

3. La divergenza informazionale e l'entropia mista.

Questa sezione, in cui introdurremo un'importante disuguaglianza della statistica e della teoria dell'informazione, è una prima divagazione dedicata alle misure d'informazione e alle loro proprietà; ce ne saranno altre, un po' noiose (è per questo che le abbiamo "disperse" nel testo), ma inevitabili. A tranquillità del lettore: le formule della teoria dell'informazione sono semplici da ricordare appena si pensa al loro significato euristico, e non causano traumi come le formule di trigonometria.

Se P e Q sono due vettori di probabilità si definisce la loro *divergenza informazionale* (o *discriminazione*) come:

$$D(P\|Q) = \sum_{1\le i\le K} p_i \log \frac{p_i}{q_i}$$

In questa definizione conveniamo che sia $0\log 0 = 0$, $0\log\frac{0}{0} \doteq 0$ e $-a\log 0 = +\infty$ per $a > 0$; conveniamo inoltre che sia infinita una somma di addendi non-negativi in cui almeno un addendo è infinito. Si tratta di convenzioni che saranno sempre sottintese nel seguito. Si noti tuttavia che la seconda *non* è una convenzione di continuità ($\Box$! Non esiste il limite $\lim x\log\frac{x}{y}$ quando gli argomenti tendono a 0); si tratta piuttosto di una convenzione "misuristica", secondo cui gli eventi di misura zero sono irrilevanti; qui è nulla sia la P-misura che la Q-misura della lettera (dell'evento elementare") a_i.

Converrà subito pensare euristicamente alla divergenza come a una “distanza”, sia pur non metrica e neanche simmetrica, fra vettori di probabilità. Vedremo in un successivo capitolo che la divergenza è un indice di “distinguibilità statistica” fra due leggi di probabilità, una vera e una falsa, di cui a priori non si sa quale sia la vera: tanto più grande è la divergenza, tanto più facile è scoprire la verità mediante sperimentazione statistica.

La divergenza può essere infinita; più precisamente lo è quando esiste una lettera a_i di Q-probabilità nulla ma di P-probabilità positiva. Nel linguaggio forbito dei misuristi: $D(P\|Q)$ è finita sse la “misura” P è *assolutamente continua* rispetto alla “misura” Q: $D(P\|Q) < +\infty$ sse $P \ll Q$. Più interessante è la limitazione inferiore della divergenza:

Non-negatività della divergenza:

$$D(P\|Q) \geq 0 \text{ , con uguaglianza sse } Q = P$$

Dimostrazione. Supponiamo pure che i logaritmi siano naturali, per cui le entropie sono misurate in *nat* (ciò non è restrittivo: □ !). Supponiamo dapprima che P e Q non abbiano zeri; basterà applicare la disuguaglianza $\ln x \geq 1 - \frac{1}{x}$, con uguaglianza sse $x = 1$, a ciascuno dei K addendi della differenza divergenza. Gli zeri eventuali non creano problemi, come il lettore verificherà. □

Esercizio. Più in generale si ha la *disuguaglianza della somma logaritmica (log-sum inequality)*. Dati i due K-vettori non negativi $\{a_1, a_2, \ldots, a_K\}$ e $\{b_1, b_2, \ldots, b_K\}$ a somma $\sum a_i \doteq A$ e $\sum b_i \doteq B$ si ha:

$$\sum_{1 \leq i \leq K} a_i \log \frac{a_i}{b_i} \geq A \log \frac{A}{B}$$

con uguaglianza sse i due vettori sono proporzionali.

Dimostrazione. “Forza” le due K-uple a diventare vettori di probabilità dividendo le loro componenti per A e per B. Controlla a parte i casi $A = 0$, $B = 0$. □

Esercizio: La divergenza è *inferiormente continua.* La continuità infatti si perde ai bordi, dove la divergenza può “saltare in aria” (pensa ai singoli addendi $x \log \frac{x}{y}$ quando x e y tendono a 0). E' colpa di una delle nostre convenzioni, la $0 \log \frac{0}{0} = 0$, quella appunto che *non* è una convenzione di continuità.

La somma $D(P\|Q) + H(P)$ fra la divergenza e l'entropia di Shannon si chiama talvolta *entropia mista* (o *inaccuratezza*, *inaccuracy*):

$$H(P\|Q) = \sum_{1 \leq i \leq K} p_i \log q_i$$

La non-negatività della divergenza si traduce subito nella cosiddetta *disuguaglianza di Gibbs*, che mostra come l'entropia mista minima $\min_Q H(P\|Q)$ sia proprio l'entropia di Shannon $H(P) \equiv H(P\|P)$, con uguaglinza sse $P = Q$: $-\sum p_i \log q_i \geq -\sum p_i \log p_i$.

Esercizio: $\min_P H(P\|Q) = -\log \max_i q_i$; l'$=$ vale sse P è deterministica.

4. L'inevitabile prezzo dell'attendibilità

Per poter confrontare l'economicità di due codici converrà arricchire il nostro modello, e assegnare delle probabilità alle lettere primarie:

$$p_i = \mathrm{Prob}(a_i) > 0$$

tramite un vettore di probabilità $P = (p_1, p_2, \ldots, p_K)$; per il momento possiamo escludere le probabilità nulle, depennando le eventuali lettere di probabilità 0 (questo giochetto non riesce se c'è più di una probabilità in ballo).

Siano $l_1, l_2, \ldots, l_K$ le lunghezze delle parole di un codice β-rio. Rimane allora definita la *lunghezza media* delle parole di codice:

$$EL = \sum_{1 \leq i \leq K} p_i l_i$$

E' proprio questo il parametro "unidimensionale" che ci consentirà di confrontare il costo di codici diversi.

Introduciamo un secondo vettore di probabilità $Q \propto (\beta^{-l_1}, \beta^{-l_2}, \ldots, \beta^{-l_K})$, stavolta "formale" (non descrive nessun fenomeno aleatorio); le sue componenti vanno divise per $\alpha \doteq \sum_i \beta^{-l_i}$. Per la non negatività della divergenza (per la disuguaglianza di Gibbs):

$$D_\beta(P\|Q) = -H_\beta(P) + EL + \log_\beta \alpha \geq 0$$

In un codice UD vale la disuguaglianza di Kraft, per cui $\log_\beta \alpha \leq 0$. Ma allora abbiamo provato il seguente risultato inverso, "normativo":

Teorema. La lunghezza media EL di un codice UD è maggiore o uguale all'entropia $H_\beta(P)$.

Esercizio. Nella disuguaglianza precedente vale l'uguale sse $l_i = -\log_\beta p_i$, una condizione "difficile" da rispettare, perchè i logaritmi di solito non hanno la gentilezza di essere numeri interi. (Cenno: ripercorrendo i conti, per l'uguale dev'essere $\frac{\beta^{-l_i}}{\alpha} = p_i$ e $\alpha = 1$, ossia $\beta^{-l_i} = p_i$.)

Esercizio. Dobbiamo codificare le lettere a, b, c di probabilità $\frac{1}{2}, \frac{1}{4}, \frac{1}{4}$; i due codici $\{0, 10, 11\}$ e $\{1, 00, 10\}$, il primo a prefisso, il secondo a ritardo di decodifica infinito, sono "ottimi", ossia hanno lungezza media minima (infatti essa è uguale all'entropia: $EL = H(P) = 1,5$ bit).

5. I codici di Shannon-Fano

Assumiamo un punto di vista propositivo: esibiremo un codice che garantisce "buone prestazioni"; è attendibile ma non costa troppo. Le lunghezze $l_i = \lceil -\log_\beta p_i \rceil$ verificano la disuguaglianza di Kraft (□ !) e dunque sono adeguate a costruire un codice a prefisso, che si chiama il *codice di Shannon-Fano*. Per la sua lunghezza media:

$$EL = \sum_{1 \le i \le K} p_i \lceil -\log_\beta p_i \rceil < H_\beta(P) + 1$$

A maggior ragione è minore dell'entropia più un β-it la lunghezza media di un codice ottimo:

$$H_\beta(P) \le \min EL < H_\beta(P) + 1$$

Esercizio: prima delusione. Nel caso sciocco in cui l'alfabeto primario e quello secondario siano entrambi binari, il codice ottimo è $\{0,1\}$ (o $\{1,0\}$) di lunghezza media 1. Se $p_a \approx 0$ il codice di Shannon-Fano assegna ad a una parola di codice assurdamente lunga (che per fortuna quasi non si usa): $l_a = \lceil -\log_\beta p_a \rceil \gg 1$. Dunque: il codice di Shannon-Fano non è in generale ottimo.

> L'algoritmo di Huffman risolve in modo brillante il problema dei codici ottimi. L'interesse dei codici di Shannon-Fano è solo teorico, sono più comodi per fare i conti.

Esercizio: seconda delusione. Riprendiamo il caso sciocco dell'esercizio precedente. Se $p_a \approx 0$, anche $H(P) = h(p_a) \approx 0$, e dunque $\min EL \approx H(P) + 1$. Ciò dimostra che l'intervallo di variazione $[H(P), H(P)+1]$ per $\min EL$ non si può restringere.

Esercizio: miscodifica. Costruiamo un codice di Shannon-Fano β-ario per il vettore di probabilità sbagliato Q e usiamolo per il vettore di probabilità giusto P. La lunghezza media del codice è $\sum_i p_i \lceil -\log_\beta q_i \rceil$, per cui:

$$H_\beta(P) + D_\beta(P\|Q) \le EL < H_\beta(P) + D_\beta(P\|Q) + 1$$

6. I codici di Huffman

Stabilito che i codici affetti da ritardi di decodifica sono inutili, nella ricerca dei codici UD ottimi potremo limitarci ai soli codici a prefisso: l'algoritmo di Huffman costruisce appunto un codice a prefisso ottimo, ossia di lunghezza media minima. Per ora costruiremo codici *binari*, in cui le parole di codice sono sequenze di 0 e 1; a fine capitolo accenneremo anche ai codici di Huffman β-ari. Come vedremo il problema di costruire un codice ottimo per K lettere viene riportato a quello di costruire un codice ottimo per $K-1$ lettere: procedendo iterativamente, il problema si riduce a quello, banalissimo, di costruire un codice binario ottimo per due sole lettere primarie.

Senza effettiva restrizione supponiamo che sia $p_1 \geq p_2 \geq \ldots \geq p_K > 0$. Partiremo da due osservazioni. Prima osservazione: sia c_i di lunghezza l_i la parola di codice più lunga. Fissa l'attenzione al nodo che precede la foglia e alle parole di codice che per esso "transitano": per ragioni di lunghezza massima può esserci al più una parola di codice identica a c_i salvo nell'ultimo bit; sì che c'è ! se non ci fosse potremmo "tagliare" l'ultimo bit di c_i, senza violare la proprietà del prefisso ma riducendo la lunghezza media del codice, che dunque non sarebbe ottimo. Seconda osservazione: sia $i < j$ ma $l_i \geq lj$: se scambiamo le parole di codice c_i e c_j la lunghezza media o non cambia o diminuisce. Infatti, se EL' è la nuova lunghezza media, $EL - EL' = (p_i - p_j)(l_i - l_j) \geq 0$. Dunque:

Lemma del codice ben formato. Esiste un codice ottimo a prefisso in cui $l_{K-1} = l_K$ è la lunghezza massima delle parole di codice, e in cui le parole di codice c_{K-1} e c_K sono identiche salvo nell'ultimo bit.

Poniamoci dal punto di vista di un osservatore miope che non distingue fra di loro le due lettere più improbabili, a_{K-1} e a_K. Invece dell'alfabeto primario *completo*, $\mathcal{A} = \{a_1, a_2, \ldots, a_{K-1}, a_K\}$, retto dalla probabilità $P = (p_1, p_2, \ldots, p_{K-1}, p_K)$, egli vede un alfabeto *ridotto*, o *ingrossato*, $\mathcal{A}^* = \{a_1, a_2, \ldots, a_{K-2}, a^*_{K-1}\}$, retto dalla probabilità $P^* = (p_1, p_2, \ldots, p_{K-1}, p^*_{K-1})$, con $a*_{K-1} = \{a_{K-1}, a_K\}$, $p^*_{K-1} = p_{K-1} + p_K$. Anche i codici si possono ridurre o completare:

Taglio (potatura). Dato un codice $\mathcal{C}$ per l'alfabeto completo $\mathcal{A}$ in cui le parole di codice di a_{K-1} e a_K differiscono solo nell'ultimo bit, si può ottenere un codice $\mathcal{C}^*$ per l'alfabeto ridotto $\mathcal{A}^*$ "tagliando" quel bit. La lunghezza media scende di $p^*_{K-1} = p_{K-1} + p_K$ (□ !).

Geminazione. Dato un codice $\mathcal{C}^*$ per l'alfabeto ridotto $\mathcal{A}^*$, si può ottenere un codice $\mathcal{C}$ per l'alfabeto completo $\mathcal{A}$ "aggiungendo" alla parola di codice di a^*_{K-1} una volta il bit 0 e una volta il bit 1. La lunghezza media sale di $p^*_{K-1} = p_{K-1} + p_K$ (□ !).

L'algoritmo di Huffman è basato sul seguente:

Lemma di Huffman. La geminazione di un codice ottimo per l'alfabeto ridotto produce un codice ottimo per l'alfabeto completo. Viceversa, il taglio di un codice ottimo ben formato per l'alfabeto completo produce un codice ottimo per l'alfabeto ridotto.

Dimostrazione. Sia C* il codice ottimo per l'alfabeto ridotto, e sia EL^* la sua lunghezza media, che è la minima possibile. Geminiamolo: si trova un codice per l'alfabeto completo, la cui lunghezza media è $EL^* + p^*_{K-1}$. Passiamo a un codice ottimo per l'alfabeto completo: la lunghezza media scende a $EL^* + p^*_{K-1} - \Delta$; $\Delta = 0$ sse già il codice geminato era ottimo. Per il lemma del codice ben formato possiamo supporre che al codice ottimo completo possa essere praticata l'operazione di taglio: si trova un codice per l'alfabeto ridotto di lunghezza media $(EL^* + p^*_{K-1} - \Delta) - p^*_{K-1} = EL^* - \Delta$. Dunque $\Delta = 0$. Il viceversa si dimostra allo stesso modo. □

Esempio. Si tratta di "meccanizzare" l'algoritmo di Huffman; pensando all'albero, il taglio di una fronda porta alla "fusione" di due foglie in un'unica foglia dell'albero ridotto. Sia

$K = 4$, $\mathcal{A} = \{a, b, c, d\}$, $P \propto (4, 3, 2, 1)$ (le vere probabilità vanno espresse in decimi). Nella prima tabella indichiamo le successive fusioni, riga dopo riga (iterazione dopo iterazione), specificando le probabilità $\times 10$ delle "macrolettere" ottenute nelle fusioni:
o (a;4) (b;3) (c;2) (d;1)
i (a;4) (b;3) (cd;3)
ii (a;4) (bcd;6)
iii (abcd;10)

Nella seconda tabella gli asterischi indicano spezzoni di parola di codice non ancora costruiti. Ogni fusione comporta la produzione di due bit, uno per ciascuno dei due oggetti fusi. Se l'oggetto è una macrolettera, il bit va replicato per ognuna delle "microlettere" che la costituiscono; nella riga *ii*, ad esempio, i due bit finali delle parole di codice di c e di d, scritti più in piccolo, erano già stati costruiti all'iterazione precedente, alla riga *i*. Rammentate che le parole di codice vengono costruite "a gambero", a partire dal bit di coda! La lunghezza media delle parole di codice (non accorciabile se si vuole conservare l'univoca decodificabilità) è $EL = 4/10 \times 1 + 3/10 \times 2 + (2/10 + 1/10) \times 3 = 1,9$ bit. Il valore dell'entropia di Shannon, come sappiamo trattandosi di un codice ottimo e come il lettore potrà verificare calcolando $H(P)$, è allora compreso fra 0.9 e $1,9$. Al lettore il compito di disegnere l'albero di codice:
i (c;$*$0) $\|$ (d;$*$1)
ii (b;$*$0) $\|$ (c;$*1_0$) (d;$*1_1$)
iii (a;0) $\|$ (b;1_0) (c;1_{10}) (d;1_{11})

Il fatto curioso che l'algoritmo proceda "a gambero" è poco naturale, tant'è che i padri fondatori della teoria dell'informazione per anni dovettero accontentarsi di codici subottimi. L'algoritmo di Huffman, se ben attuato, ha una complessità molto bassa, dell'ordine di $K \log K$, il che spiega la sua popolarit d'uso nella compressione dei dati.

Nota: l'algoritmo di Huffman determina la lunghezza media, e non le lunghezze individuali. Proviamolo con un esempio. Le probabilità delle lettere primarie a, b, c, d siano proporzionali a $3, 3, 2, 1$. Puoi costruire ($\square$!) *due* codici di Huffman con lunghezze individuali diverse (nel secondo caso il codice è *a blocco*, come si usa dire quando le parole di codice hanno tutte la stessa lunghezza). Primo codice: $\{0, 10, 110, 111\}$; secondo codice: $\{00, 01, 10, 11\}$. Naturalmente la lunghezza media è la stessa nei due casi: $EL = 2$.

Esercizio. Il taglio di un codice ottimo per l'alfabeto completo produce un codice ottimo per l'alfabeto ridotto.

Complementi: Codici di Huffman β-ari. E' naturale congetturare che nel caso dei codici ottimi β-ari ad ogni passo si debbano fondere β lettere primarie anzichè due. La congettura è sostanzialmente corretta; c'è però un problema reso evidente dalla figura: per banali ragioni aritmetiche siamo costretti a "fondere" da qualche parte meno di β lettere. L'evento è spiacevole e, come si può dimostrare, è meglio che avvenga all'inizio, quando le probabilità in gioco sono basse. Quante lettere vanno fuse? vediamolo. Costruiamo in $j + 1$ passi un albero β-ario con una "fronda" forse incompleta di Δ foglie. Si ha: $2 \leq \Delta \leq \beta$ (una fronda

di un solo ramo pendulo non serve, perchè potremmo potarla senza violare la proprietà del prefisso).
Passo 1. A partire dalla radice, crea β foglie
Passo intermedio. Incolla a una foglia una fronda completa: il numero delle foglie cresce di $\beta - 1$
Passo $j + 1$. Incolla a una foglia una fronda incompleta: il numero delle foglie cresce di $\Delta - 1$

Alla fine ci devono essere K foglie, e dunque: $K = j(\beta - 1) + \Delta$
Se togliamo 2 ai due membri: $K - 2 = j(\beta - 1) + (\Delta - 2)$, con $\Delta - 2$ compreso fra 0 e $\beta - 2$ e dunque possibile resto in una divisione per $\beta - 1$. Ma allora: $\Delta - 2$ è il resto che si ottiene quando si divide $K - 2$ per $\beta - 1$:

$$\Delta = 2 + \text{resto}(K - 2 : \beta - 1)$$

Esempio Sia $K = 9$, $\beta = 4$. Si ha $\Delta = 2 + \text{resto}(7 : 3) = 3$. Nella prima iterazione vengono fuse le tre lettere più improbabili; nelle due iterazioni successive vengono fuse $\beta = 4$ (macro)lettere.

Esercizio. Gli alberi di Huffman binari sono sempre saturi (zeppi): ne consegue che la disuguaglianza di Kraft è verificata con l'uguale.

7. Risultati asintotici: sorgenti estese

Un codice Morse non viene usato per codificare una singola lettera, bensì un *testo*, ossia una successione di lettere. Supporremo che le lettere del testo vengano emesse da una *sorgente d'informazione*, la quale, in linea di principio (*in potenza*, come dicevano gli antichi) opera senza limiti di tempo, e genera un testo infinito a destra ("arbitrariamente lungo"). Per le sorgenti si adotta spesso un un modello piuttosto primitivo, quello bernoulliano (stazionario e senza memoria), che non dà ragione dei comportamenti aleatori molto complessi delle sorgenti d'informazione "naturali". Ciò nonostante, il modello bernoulliano si rivelerà assai istruttivo e utile in pratica, oltre che ben trattabile dal punto di vista matematico. Del resto anche il codice Morse ipotizza che il processo di generazione del testo linguistico sia stazionario e senza memoria: la parola di codice della lettera U è lunga perchè la U è improbabile; se però è stata appena codificata una Q la probabilità *condizionata* della U "salta" a un valore prossimo a 1 (che non è 1 solo per colpa del soqquadro, delle linee aree australiane Qantas, dell'Iraq, ecc); nel codice Morse, tuttavia, non c'è nessuna "parola di codice condizionata" per la U, molto breve, da utilizzare in questi casi.

Per i formalisti che sottovalutano la forza suggestiva delle parole: la voce *sorgente d'informazione* è un doppione, i probabilisti parlano di *processi stocastici*.

Esempio. Sia $\mathcal{A} = \{a, b\}$, $P = (\frac{4}{5}, \frac{1}{5})$; il banalissimo codice di Huffman per le lettere singole ha lunghezza media $EL = 1$. Codifichiamo piuttosto le coppie primarie, ossia la sorgente bernoulliana di "alfabeto" $\mathcal{A}^2 = \{aa, ab, ba, bb\}$ retta dalla probabilità

$P^2 = (\frac{16}{25}, \frac{4}{25}, \frac{4}{25}, \frac{1}{25})$; il lettore disegni l'albero di decodifica con le sue quattro foglie contrassegnate aa, ab, ba, bb. La lunghezza media è $\frac{39}{25} \approx 1,6$. Il numero medio di bit per lettera primaria di $\mathcal{A}$ è $\frac{EL}{2} \approx 0,8$. C'è stato un risparmio di un quinto di bit in media (anche se "localmente" si possono presentare sprechi, basta pensare alla coppia improbabile bb che è codificata con tre bit invece di due).

L'insegnamento che si trae da questo esempio è: se siamo disposti ad aumentare la complessità dello schema, possiamo migliorare la compressione dei dati.

Data una sorgente che genera lettere, nulla vieta di "dilatare" l'unità di tempo, e di vederla come una sorgente che genera n-uple (nell'esempio era $n = 2$). La nuova *sorgente estesa* Ξ^n, le cui "pseudolettere" sono i K^n "blocchi" primari di lunghezza n, è anch'essa bernoulliana, ossia stazionaria e senza memoria ($\square$!), ed è retta dal vettore di probabilità P^n, di entropia $H_\beta(P^n) = nH_\beta(P)$.

Esercizio. Si ha $H_\beta(P^n) = nH_\beta(P)$, $D_\beta(P^n \| Q^n) = nD_\beta(P\|Q)$,$H_\beta(P^n\|Q^n) = nH_\beta(P\|Q)$. Basta ricordare che il logaritmo di un prodotto è la somma dei logaritmi; della prima proprietà daremo fra poco una versione più generale.

Se codifichiamo le n-uple alla Huffman (se codifichiamo Ξ^n alla Huffman), la lunghezza media EL_n delle parole di codice è compresa fra $H_\beta(P^n) = nH_\beta(P)$ e $H_\beta(P^n) + 1$, e dunque:

$$H_\beta(P) \leq \frac{EL_n}{n} \doteq R_n < H_\beta(P) + \frac{1}{n}$$

La "lunghezza media normalizzata" R_n si chiama il *tasso* del codice (*rate* in inglese): esso dice il numero di bit, o più in generale di β-it, usati in media per codificare *ciascuna* lettera primaria. Più il tasso è basso, più il codice è economico. In generale, il tasso di un codice UD non può "battere" l'entropia ($\square$!):

$$R_n \geq H_\beta(P)$$

Usare un parametro normalizzato come il tasso consente di confrontare in maniera equa codici di complessità diversa. Al posto del tasso, si considera talvolta il *rapporto di compressione*, che è l'inverso del tasso, e che conviene render massimo.

Teorema asintotico. Al crescere della lunghezza di blocco n il tasso di un un codice di Huffman tende all'entropia $H_\beta(P)$.

Il teorema asintotico, la cui dimostrazione è ormai men che banale, ha un contenuto profondo, e chiarisce il significato "operativo" dell'entropia di Shannon come *tasso ideale* di codifica, cui si deve mirare, anche se, per ragioni di complessità tiranna, non sempre esso si può raggiungere.

Esercizio. Anche i codici di Shannon-Fano sono *asintoticamente ottimi*, nel senso che il loro tasso tende all'entropia $H_\beta(P)$.

Esercizio: miscodifica. Il tasso di un codice di Shannon-Fano costruito per la sorgente estesa Q^n ma usato per codificare la sorgente estesa P^n tende all'entropia mista $H_\beta(P\|Q) = D_\beta(P\|Q) + H_\beta(P)$.

Un risultato analogo vale per i codici di Huffman, ma è molto più difficile da dimostrare. Ciò chiarisce il significato "operativo" sia dell'entropia mista, sia della divergenza, che è lo "spreco" di cifre secondarie in cui si incorre quando si sbaglia il codificatore di Huffman. Il fatto che invertendo i suoi due argomenti il valore della divergenza possa cambiare non dovrebbe ormai stupire, visto che le due situazioni di mala codifica descritte sono diverse.
L'esercizio seguente mostra che i *codici a blocco*, nei quali non sono presenti elementi a lunghezza variabile, non consentono la compressione dei dati, ma solo la loro mera *trascrizione*. Sotto, n indica la lunghezza dei blocchi primari che vengono codificati, mentre l indica la lunghezza fissa delle parole di codice; usiamo la minuscola per sottolineare il fatto che tale lunghezza non ha più nulla di casuale, ed è la stessa qualunque sia il blocco di n lettere primarie che viene codificato. Un esempio celeberrimo di codice binario di trascrizione dei dati è l'ASCII.

Esercizio: codici a blocco e trascrizione dei dati. Per n fissato, il tasso ottimo $\frac{l}{n}$ di un codice a blocco vale $\frac{\lceil \log_\beta K^n \rceil}{n}$, e dunque al crescere di n tende a $\log_\beta K$. Cenno: se si vogliono evitare gli errori di decodifica, occorrono almeno tanti blocchi secondari quanti sono i blocchi primari, e dunque $\beta^l \geq K^n$.

A differenza dei codici a blocco, i *codici a lunghezza variabile*, come il codice di Huffman, consentono una vera e propria compressione dei dati. Fanno eccezione le sorgenti totalmente casuali (P uniforme, $H(P) = \log K$): esse sono incomprimibili, o come anche si dice, sono prive di *ridondanza*. Vedremo più avanti che è possibile comprimere i dati anche con codici a blocco, purché si "allenti" in maniera opportuna il criterio di attendibilità.

II. CODIFICA DELLE SORGENTI NATURALI

8. L'entropia condizionata

La compressione dell'informazione emessa da una sorgente "naturale", non vincolata dalla restrittiva ipotesi della mancanza di memoria, è un capitolo assai delicato della teoria dell'informazione. Nelle sezioni che seguono lo affronteremo senza strafare, e generalizzzeremo il concetto di entropia di una sorgente d'informazione. Il lettore non se l'abbia a male se cominciamo in tono minore, con una nuova digressione che approfondisce il nostro discorso sulle misure d'informazione. Il lettore rammenti che conviene vedere l'entropia del vettore di probabilità P come una misura dell'incertezza "in esso contenuta". Più avanti convalideremo (finalmente!) quest'interpretazione.

La variabile aleatoria Z oggetto del nostro discorso potrebbe ben avere carattere bidimensionale: $Z = XY \in \mathcal{A} \times \mathcal{B}$, $X \in \mathcal{A}$, $Y \in \mathcal{B}$. Nel caso di una variabile aleatoria bidimensionale (di una coppia aleatoria) le probabilità congiunte vengono di solito organizzate in una matrice P_{XY} a $|\mathcal{A}|$ righe e $|\mathcal{B}|$ colonne; l'entrata all'incrocio della riga a-esima (intestata ad a) e della colonna b-esima (intestata a b) è $p_{ab} \doteq \text{Prob}\{X = a, Y = b\}$.

Definizione. L'*entropia congiunta* della coppia aleatoria XY è

$$H(XY) = -\sum_a \sum_b p_{ab} \log p_{ab}$$

L'entropia congiunta *non* è un nuovo tipo di entropia, semplicemente si mette in evidenza il carattere bidimensionale dell'argomento di H. Allo stesso modo si può parlare di entropie congiunte di terne, o più genericamente di n-uple aleatorie.

Invece della matrice delle probabilità congiunte, la distribuzione di probabilità della coppia XY può venir assegnata tramite il vettore P_X delle probabilità marginali della prima componente aleatoria X ($P_X(a) \doteq \text{Prob}\{X = a\}$) e tramite la *matrice stocastica* W ad $|\mathcal{A}|$ righe e $|\mathcal{B}|$ colonne delle *probabilità condizionate* $p_{b|a} \doteq \text{Prob}\{Y = b|X = a\}$.

Esercizio-richiamo. Data P_{XY} costruisci P_X e W; date P_X e W costruisci P_{XY} (usa le formule $\text{Prob}\{X = a\} = \sum_b \text{Prob}\{X = a, Y = b\}$ e $\text{Prob}\{X = a, Y = b\} = \text{Prob}\{X = a\}\text{Prob}\{Y = b|X = a\}$; le righe di W che corrispondono a probabilità $P_X(a)$ nulle sono del tutto irrilevanti).

Esercizio-richiamo. Data la matrice delle probabilità congiunte P_{XY} costruisci i due vettori delle probabilità marginali P_X e P_Y ($P_Y(b) = \text{Prob}\{Y = b\}$: basta sommare le righe o, rispettivamente, le colonne di P_{XY}, ossia basta "saturare" l'indice di colonna o, rispettivamente, l'indice di riga). A partire da P_X e P_Y possiamo costruire mediante prodotti una nuova matrice di probabilità congiunte $P_X \times P_Y$. Si ha $P_{XY} = P_X \times P_Y$ sse le componenti aleatorie X e Y sono *indipendenti.*

Le righe di W sono vettori di probabilità su $\mathcal{B}$ (probabilità condizionate, ma pur sempre probabilità); a ciascun vettore compete la propria entropia: $H(Y|X = a) = -\sum_b p_{b|a} \log p_{b|a}$.

Nulla di nuovo sotto il sole: sono entropie *normali* di probabilità condizionate. Esse dicono qual è l'incertezza sulla seconda componente, se è noto che il valore assunto dalla prima componente è proprio a. Qual è l'incertezza *media* sulla seconda componente?

Definizione. L'*entropia condizionata* della componente aleatoria Y data che sia la componente aleatoria X è :

$$H(Y|X) = \sum_{a \in \mathcal{A}} \text{Prob}\{X = a\} H(Y|X = a)$$

$$= -\sum_a \sum_b \text{Prob}\{X = a, Y = b\} \log W_{b|a} \equiv -\sum_a \sum_b p_{a,b} \log p_{b|a}$$

con la convenzione che sono nulli gli addendi per i quali $\text{Prob}\{X = a\} = 0$.

Finalmente una nozione davvero nuova!

Teorema: $H(XY) = H(X) + H(Y|X) = H(Y) + H(X|Y)$
($\square$! è una mera verifica sulle formule.)

La nostra euristica funziona benissimo: l'incertezza sulle due componenti è l'incertezza sulla prima componente più l'incertezza sulla seconda nota che sia la prima.

Nulla vieta che in $H(X|Y)$ X e/o Y abbiano carattere multidimensionale. Per induzione, la formula precedente si generalizza come segue:

Regola della catena (chain rule):

$$H(X_1, X_2, \ldots, X_n) = H(X_1) + H(X_2|X_1) + H(X_3|X_1, X_2) + \ldots + H(X_n|X_1, X_2, \ldots, X_{n-1})$$

L'euristica dell'incertezza torna benissimo con i risultati che seguono; anzi, se non valessero meglio sarebbe abbandonarla.

Teorema: $0 \leq H(Y|X) \leq H(Y)$
con l'uguaglianza a sinistra sse $Y = f(X)$ (la Y è funzione deterministica della X)
con l'uguaglianza a destra sse X e Y sono indipendenti.

Dimostrazione. A sinistra: pensa alle entropie normali $H(Y|X = a)$ che mediate danno l'entropia condizionata; perchè sia $H(Y|X) = 0$ la matrice W dev'essere binaria, con un 1 per ciascuna riga (per ciascun vettore di probabilità condizionata).
A destra: facendo intervenire la divergenza si ha $H(Y) - H(Y|X) = D(P_{XY} \| P_X \times P_Y)$ (è una verifica!). Usa la non negatività della divergenza informazionale. $\square$

Corollario (subadditività dell'entropia): $H(XY) \leq H(X) + H(Y)$
con l'uguaglianza sse X e Y sono indipendenti.

Esercizio ripreso: formula di riduzione (*diramazione*). Stavolta l'indice nelle entropie non indica la base dei logaritmi, ma la dimensione dei vettori di probabilità in argomento.

$$H_K(p_1, p_2, p_3, \ldots, p_K) = H_{K-1}(p_1 + p_2, p_3, \ldots, p_K) + (p_1 + p_2)h(\frac{p_1}{p_1 + p_2})$$

Cenno: Introduci la variabile aleatoria X^* che "ingrossa" la X di partenza e poi svolgi in due modi l'entropia congiunta $H(X, X^*)$.
La formula consente di riportare il computo di un entropia sull'alfabeto $\mathcal{A}$ di K lettere al computo di $K - 1$ entropie binarie (per cui basta la tabella del capitolo 1)

Esercizio: disuguaglianza di Fano. Siano X e X^* due variabili aleatorie sullo stesso alfabeto $\mathcal{A}$ (o anche: sia X, X^* una coppia aleatoria sull'alfabeto $\mathcal{A}^2$); converrà pensare a X^* come a una "ricostruzione statistica" (stima statistica) di X e a $\epsilon = \text{Prob}\{X \neq X^*\}$ come a una *probabilità d'errore* (di errata ricostruzione). Si ha:

$$H(X|X^*) \leq h(\epsilon) + \epsilon \log(K-1)$$

ossia: l'incertezza su X data che sia la sua ricostruzione è l'incertezza se sia avvenuto o meno un errore, più, se l'errore è ahimè avvenuto, l'incertezza fra le K lettere diverse dalla lettera ricostruita. Cenno: Aggiungi l'indicatore di successo binario Z che vale 1 se la ricostruzione è corretta, e svolgi in due modi l'entropia congiunta (e condizionata) $H(X, Z|X^*)$.

9. Sorgenti stazionarie

Sia Ξ una sorgente d'informazione la quale emette lettere aleatorie a istanti discreti: $X_1, X_2, \ldots, X_n, \ldots$, dove X_n è la lettera aleatoria di $\mathcal{A}$ emessa all'istante n.

Definizione. L'entropia della sorgente Ξ è

$$H(\Xi) \doteq \lim_n \frac{H(X_1, X_2, \ldots, X_n)}{n}$$

(se il limite non esiste la sorgente è sprovvista di entropia).

Esercizio. Se Ξ è bernoulliana $H(\Xi) = H(P)$.

La sorgente Ξ si dice *stazionaria* se "eventi traslati nel tempo conservano la loro probabilità". In termini formali basta richiedere, per ogni n e Δ interi e per ogni n-upla di uscite $x_1, x_2, \ldots, x_n$:

$$P^n\{X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n\} = P^n\{X_{1+\Delta} = x_1, X_{2+\Delta} = x_2, \ldots, X_{n+\Delta} = x_n\}$$

Le sorgenti stazionarie sono ragionevolmente adeguate se si vuole modellare una lingua naturale. Un'approssimazione più rozza si accontenta di un modello *markoviano di memoria 3*, col risultato che segue, che è nella lingua di Shakespeare. L' esempio "suona" abbastanza inglese, ma purtroppo non vuol dir nulla; in realtà nelle lingue naturali ci sono dipendenze dettate dalla semantica che provocano fenomeni di memoria a largo raggio. Per esser pignoli, neppure la stazionarietà è garantita (le lingue evolvono nel tempo, certe parole passano di moda anche nel breve arco di una novella, ciò che può far via via calare la probabilità di vedere le lettere che le compongono). Per i più esperti di calcolo della probabilià aggiungiamo che non è garantita neanche l'*ergodicità*, basti pensare a cosa succede se ci sono inserti in un'altra lingua, come il latino in una grammatica latina, oppure il "matematichese" in un libro pieno di formule (senza entrare nei dettagli, una sorgente non ergodica si "spezza" in due o più componenti ognuna regolata dalle proprie leggi statistiche, che funzionano a turno, alternadosi in maniera casuale.

Esempio THE GENERATED JOB PROVIDUAL BETTER TRAND THE DISPLAYED CODE, ABOVERY UPONDULTS WELL THE THESTICAL CODERST DO SHOCK.

Nel teorema che segue poniamo $H_n = \frac{H(X_1, X_2, \ldots, X_n)}{n}$, $h_n = H(X_n|X_1, X_2, \ldots, X_{n-1})$.

Teorema. Le sorgenti stazionarie sono provviste di entropia, e si ha:

$$H(\Xi) = \lim H_n = \lim_n h_n$$

Il teorema è una conseguenza immediata delle divagazioni in calce che seguono.

Divagazioni. Sia a_n una successione e sia $\overline{a}_n$ la successione delle medie aritmetiche, $\overline{a}_n = \frac{a_1 + a_2 + \ldots + a_n}{n}$.

Lemma di Cesaro. Se a_n ha limite e converge ad a, ad a converge anche la successione delle medie $\overline{a}_n$.

Dimostrazione. Fissiamo $\epsilon > 0$. Sia $|a_n - a| \leq \epsilon$ per $n > \nu$ (o meglio per $n > \nu_\epsilon$; rammenta che a_n ha limite). Allora:

$$|\overline{a}_{\nu+n} - a| = \frac{|a_1 + a_2 + \ldots + a_{\nu+n} - (\nu + n)a|}{(\nu + n)}$$

$$\leq \frac{|a_1 - a| + \ldots + |a_\nu - a| + |a_{\nu+1} - a| + \ldots + |a_{\nu+n} - a|}{\nu + n} \leq$$

$$\leq \frac{|a_1 - a| + \ldots + |a_\nu - a| + n\epsilon}{\nu + n} < \frac{|a_1 - a| + \ldots + |a_\nu - a|}{\nu + n} + \epsilon \leq 2\epsilon$$

per n sufficientemente grande (per $\nu + n \equiv \nu_\epsilon + n$ sufficientemente grande). □

Se a_n è monotona (vedremo subito che h_n lo è), il limite esiste e dunque si può applicare il lemma.

Dimostrazione del teorema. Posto $h_n = a_n$, per cui $H_n = \overline{a}_n$ (□ !), basta dimostrare che la h_n è non crescente. E infatti, togliendo il condizionamento X_1 e traslando (anticipando) i tempi di -1 per stazionarietà:

$$h_{n+1} = H(X_{n+1}|X_1, X_2, X_3, \ldots, X_n) \leq H(X_{n+1}|X_2, X_3, \ldots, X_n)$$

$$= H(X_n|X_1, X_2, \ldots, X_{n-1}) = h_n$$

□

Esercizio a completamento del teorema. Le due successioni H_n e h_n sono non crescenti, e la prima domina la seconda. Ciò segue subito dal seguente:

Lemmino. Se la successione a_n è non crescente:
1) $\overline{a}_n$ è essa stessa non crescente
2) $\overline{a}_n$ domina a_n

Dimostrazione della 1):

$$\overline{a}_{n+1} - \overline{a}_n = \frac{(a_1 - a_{n+1}) + (a_2 - a_{n+1}) + \ldots + (a_n - a_{n+1})}{n(n+1)} \geq 0$$

della 2):

$$\overline{a}_n - a_n = \frac{(a_1 - a_n) + (a_2 - a_n) + \ldots + (a_n - a_n)}{n} \geq 0$$

$\square$

($\square$! rienuncia il lemmino per successioni non decrescenti.)

10. Compressione e ridondanza.

Esercizio. Codifica con un codice di Huffman le n-uple di una sorgente Ξ provvista di entropia. Dimostra che il tasso R_n del codice tende a $H(\Xi)$ al divergere di n.

Esercizio. Se la sorgente è stazionaria, il tasso di un codice UD è $\geq H(\Xi)$ (ricorda che la successione H_n non può crescere).

> La comparsa dell'entropia nei teoremi di codifica è sistematica: la *legge generale dell'entropia* afferma, per dirla in maniera non formale, che il tasso di qualunque codifica attendibile (che consenta mediante decodifica la ricostruzione integrale dell'informazione codificata) non può scendere sotto l'entropia. La differenza fra tasso effettivo e *tasso ideale*, che è appunto l'entropia, si chiama la *ridondanza del codice*; ad esempio, la ridondanza di un codice binario di Huffman che codifica le n-uple di una sorgente stazionaria è inferiore a un ennesimo di bit.

Esercizio. Se Ξ è stazionaria, $\forall n \; 0 \leq H(X_n|X_1 \ldots X_{n-1}) \leq H(X_n) = H(X_1) = H(P)$. Si ha allora $0 \leq H(\Xi) \leq H(P)$; $H(\Xi) = H(P)$ sse Ξ è anche priva di memoria (X_n non dipende da $X_1 \ldots X_{n-1}$). Dunque, com'è naturale, la presenza di memoria diminuisce l'incertezza sull'evoluzione della sorgente stazionaria.

La *ridondanza* di una sorgente d'informazione Ξ provvista di entropia è data dalla differenza $logK - H(\Xi)$ fra il tasso ideale dei codici di trascrizione e il tasso ideale dei codici di compressione. Tanto maggiore la ridondanza, tanto più comprimibile è la sorgente. Se l'informazione primaria viene codificata da un "buon" codice di compressione dei dati, le cifre secondarie emesse dal codificatore costituiscono l'output di una nuova sorgente di informazione, che è "quasi" priva di ridondanza ("quasi" incomprimibile: altrimenti l'informazione secondaria si potrebbe ricodificare e sensibilmente comprimere con un secondo codice di sorgente, smentendo la bontà del primo codice). Anche se per semplicità ci siamo dovuti limitare alla classe delle sorgenti stazionarie, l'ultimo esercizio mostra che le uniche sorgenti incomprimibili (a ridondanza nulla) sono le sorgenti *totalmente casuali* (bernoulliane, ossia stazionarie e senza memoria, regolate da un vettore di probabilità P uniforme). Ne viene che, se si adopera un "buon" codice di sorgente, l'informazione secondaria costituita dai simboli delle parole di codice è "pressocché" totalmente casuale.

III. CODICI DI TUNSTALL

11. Segmentazione di testi

Passare dai codici B-LV a quelli LV-B, in cui i messaggi sono a lunghezza variabile LV, mentre le parole di codice sono blocchi di lunghezza fissa B, dà un po' l'impressione di camminare sul soffitto con la testa in basso: una ginnastica che fa venire il fiatone.

Il nostro scopo di partenza è descrivere regole che consentano la *segmentazione* (*parsing*) di un *testo*, ossia di una sequenza infinita sull'alfabeto $\mathcal{A}$ (meglio: semi-infinita, infinita a destra).

Conviene pensare che le lettere della sequenza vengano via via generate da una sorgente di informazione; potenzialmente la sorgente non si inceppa mai, e continua a emettere lettere in output per l'eternità.

Ci spiegheremo con un esempio. Sia $\mathcal{A} = \{a, b, c\}$; fissiamo una *famiglia di messaggi*, ossia un sottinsieme finito di $\mathcal{A}^* - \{\lambda\}$ (λ è la sequenza "nulla" di lunghezza zero); prendiamo, per dire, $\mathcal{M} = \{a, aa, b, ca, cc\} = \{a, a^2, b, ca, c^2\}$. Il testo da segmentare sia $caccbb\ldots$; la segmentazione in messaggi di $\mathcal{M}$ è $ca|cc|b|b|\ldots$. I messaggi del nostro $\mathcal{M}$ (gli elementi base della segmentazione) hanno tuttavia due pecche, la prima grave, la seconda non tanto.

Primo difetto: Non possiamo segmentare il testo $cacb\ldots$ per mancanza di messaggi ($\Box$!). *Secondo difetto*: Ci sono modi diversi di segmentare il testo $caaaaa\ldots$ per sovrabbondanza di messaggi ($\Box$!).

Esercizio: da' una definizione formale di segmentazione come applicazione di $\mathcal{A}^\infty$ in $\mathcal{M}^\infty$.

Al secondo difetto potremmo rimediare con una *regola di preferenza*, deterministica o anche casuale. Invece saremo perfino troppo esigenti e pretenderemo che le nostre famiglie di messaggi siano esenti da entrambe le pecche.

Definizione: Una famiglia di messaggi $\mathcal{M}$ si dice *esauriente* quando ogni testo su $\mathcal{A}$ infinito a destra (in pratica: sufficientemente lungo) ha almeno un prefisso in $\mathcal{M}$.

Definizione: Una famiglia di messaggi $\mathcal{M}$ si dice *a prefisso* quando due messaggi di $\mathcal{M}$ non possono essere prefisso l'uno dell'altro.

Ancora una volta, con rincrescimento, facciamo notare che la cosiddetta proprietà del prefisso meglio si chiamerebbe proprietà del non-prefisso, poichè ne predica l'assenza.

Qualunque testo ha *almeno* una segmentazione su $\mathcal{M}$ se e solo se $\mathcal{M}$ è esauriente ($\Box$!). Qualunque testo ha *al più* una segmentazione su $\mathcal{M}$ se e solo se $\mathcal{M}$ è a prefisso ($\Box$)! .

Esercizio. L'operazione di segmentazione può essere facilitata dalla costruzione dell'*albero dei messaggi* (figura). Se vale la proprietà del prefisso i messaggi corrispondono a foglie e consentono di evitare ritardi di segmentazione (più avanti troveremo conveniente chiamarli *ritardi di codifica*).

Definizione: Una famiglia di messaggi $\mathcal{M}$ si dice *completa* se e solo se essa è esauriente e a prefisso.

Qualunque testo ha *esattamente* una segmentazione su $\mathcal{M}$ se e solo se la famiglia $\mathcal{M}$ è completa ($\Box$!).

Abbozziamo ora un algoritmo iterativo che consente la costruzione di famiglie complete. Fissato il numero j di *estensioni* desiderate ($j \geq 0$) si giunge a una famiglia $\mathcal{M}_j$ di $K + j(K-1)$ messaggi.

Passo 0: $\mathcal{M}_0 = \mathcal{A}$.
...
Se $i < j$, rimani in ciclo, altrimenti chiudi.
Passo i: Da $\mathcal{M}_{i-1}$ togli un messaggio m (il *messaggio di estensione*), e sostituiscilo con i K messaggi del tipo ma, $a \in \mathcal{A}$.

$\mathcal{M}_j$ è una famiglia completa ($\Box$!). Qualunque famiglia completa può pensarsi costruita con l'algoritmo appena descritto ($\Box$! procedi a ritroso, partendo dal messaggio più lungo: nell'albero dei messaggi la fronda più esterna che lo comprende è completa, la famiglia essendo esauriente; mozziamo la fronda ... regredendo si arriva ad $\mathcal{A}$). Come corollario: una famiglia completa ha necessariamente numerosità del tipo $K + j(K-1)$; ad esempio, per $|\mathcal{A}| = K = 3$, non ci sono famiglie complete con un numero pari di messaggi.

Esercizio-complemento. Le famiglie *esaurienti* verificano la *contro-disuguaglianza* di Kraft:

$$\sum_{1 \leq i \leq |\mathcal{M}|} K^{-n_i} \geq 1$$

(Dimostra che si ha $\sum_i K^{n-n_i} \geq K^n$, dove n è la lunghezza del messaggio più lungo.) Poichè le famiglie *a prefisso* verificano la disuguaglianza di Kraft col $\leq$, ne viene che le famiglie *complete* verificano l'*uguaglianza* di Kraft:

$$\sum_{1 \leq i \leq |\mathcal{M}|} K^{-n_i} = 1$$

12. Alberi di Tunstall

Supporremo d'ora in avanti che il testo da segmentare venga via via generato da una sorgente bernoulliana (stazionaria e senza memoria) Ξ retta dal vettore di probabilità $P = (p_1, p_2, \ldots, p_K)$, $p_i = \text{Prob}\{X_t = a_1\}$, dove X_t è la lettera aleatoria emessa dalla sorgente all'istante t, $t \geq 1$. Perseguiremo l'identificazione fra la famiglia completa di messaggi costruita con j estensioni e l'albero corrispondente. Fra i nodi dell'albero dei messaggi c'è anche, a *profondità* 0, la *radice*, e ci sono, a profondità variabile, le *foglie*, ossia i nodi terminali che corrispondono ai messaggi della famiglia completa. Ogni nodo può venir *contrassegnato* (o etichettato, *labelled*) mediante il prefisso di messaggio che gli compete, e *pesato* mediante la probabilità bernoulliana di quel prefisso; la radice viene etichettata mediante la sequenza nulla λ di lunghezza zero (prefisso "universale" di ogni messaggio) e viene pesata con la probabilità 1. I pesi decrescono via via che dalla radice ci spostiamo verso le foglie. Le probabilità delle foglie sommano a 1 ($\Box$!). Poniamoci di fronte alla sorgente d'informazione da un punto di vista diverso, quello di un osservatore cui interessano i messaggi e non le lettere singole. Ciò che egli percepisce è la sorgente dei messaggi Ξ_j, le cui $|\mathcal{M}_j| = K + j(k-1)$ "lettere" sono i messaggi di $\mathcal{M}_j$.

Esempio. Sia $\mathcal{A} = \{a, b, c\}$, $P = (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$, $j = 2$, $\mathcal{M}_2 = \{a, ba, bb, bca, bcb, bcc, c\}$. L'alfabeto della sorgente estesa Ξ_2 è $\mathcal{M}_2$ e le probabilità delle sette "lettere" di Ξ_2 sono, nell'ordine: $P_j = (\frac{1}{2}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}, \frac{1}{64}, \frac{1}{4})$.

E' un fatto notevole (e molto comodo dal punto di vista matematico) che anche la sorgente dei messaggi Ξ_j sia stazionaria e senza memoria ($\Box$!)

La lunghezza media dei messaggi della famiglia completa, ossia la profondità media delle foglie dell'albero, è

$$EN_j = \sum_{m \in \mathcal{M}} |m| P^*(m)$$

dove $|m|$ denota la lunghezza del messaggio m; il simbolo P^* al posto di $P^{|m|}$ nell'addendo i-esimo ci serve solo per alleggerire la scrittura.

Nella costruzione di una famiglia completa intervengono i messaggi di estensione, o (*nodi di estensione*): siano essi, nell'ordine in cui sono stati usati, $m_0^* = \lambda, m_1^*, m_2^*, \ldots, m_j^*$, e siano $p_0^* = 1, p_1^*, p_2^*, \ldots, p_j^*$ le loro probabilità (fra i nodi di estensione abbiamo incluso anche la radice). Se $p_1^* \geq p_2^* \geq \ldots \geq p_j^*$ diremo che la costruzione dell'albero completo $\mathcal{M} = \mathcal{M}_j$ è stata *regolare*.

D'ora in avanti, *senza restrizione effettiva* ($\Box$!), supporremo che gli alberi di cui parliamo siano costruiti tutti in maniera regolare. (Cenno: pensa ai messaggi di estensione come nodi nell'albero di codifica: procedendo lungo i rami a partire dalla radice, la probabilità dei nodi diminuisce. Per chiarire: figura.)

La lunghezza media EN_j di un albero completo si lascia esprimere in termini delle probabilità dei nodi di estensione:

$$EN_j = \sum_{0 \leq i \leq j} P^*(m_i^*)$$

($\Box$! Per induzione su j; ricorda che la sorgente è bernoulliana.)

Per noi è di importanza cruciale il seguente problema di ottimazione:

Problema di ottimazione delle famiglie complete (ossia degli alberi completi): Dato j costruisci $\mathcal{M}_j$ in modo da render massima la lunghezza media EN_j.

Per quanto appena detto, il problema di ottimazione si traduce nel render massima la somma $\sum_i P^*(m_i^*)$. Gli *alberi di Tunstall* sono particolari alberi completi che si ottengono aggiungendo nell'algoritmo di costruzione delle famiglie complete il vincolo che segue:

Al passo i-esimo estendi il messaggio di probabilità massima.

(O meglio: estendi *un* messaggio di probabilità massimale, perchè potrebbero esserci dei pareggi; nota che la costruzione degli alberi di Tunstall nasce regolare.)

Teorema: Gli alberi di Tunstall ottenuti con j estensioni risolvono il problema dell'ottimalità degli alberi completi.

Dimostrazione. Dimostriamo che un albero completo (saturo) $\mathcal{C}^*$ a j estensioni se non è di Tunstall non può essere ottimo.
Come detto sopra, possiamo ricostruire l'albero inserendo i successivi nodi di estensione in ordine di probabilità non decrescente. Sia i il primo passo in cui la costruzione di $\mathcal{C}^*$ viola la regola di Tunstall: $P^*(u_i^*) < P^*(u_i^T)$, dove u_i^T pur era un nodo disponibile. Poichè i nodi di estensione nell'ordine d'uso hanno probabilità non decrescente, la foglia u_i^T, improvvidamente scartata, non può più venir usata come nodo di estensione nella costruzione dell'albero di $\mathcal{C}^*$, perchè ha una probabilità troppo alta, e dunque foglia rimane fino alla fine. Se dall'albero di codifica di $\mathcal{C}^*$ "stacchiamo" il sotto-albero radicato in u_i^* e lo "riattacchiamo" nella foglia u_i^T troviamo un nuovo albero completo. La sua lunghezza media ovviamente batte $\mathcal{C}^*$ (confronta la somma delle probabilità bernoulliane dei rispettivi nodi di estensione). $\square$

13. Codici completi e codici di Tunstall

Il codice viene specificato tramite una famiglia completa $\mathcal{M} = \mathcal{M}_j$ di numerosità $K + j(K-1)$. L'idea è che il codificatore segmenta il testo emesso dalla sorgente e via via che i messaggi di $\mathcal{M}$ vengono riconosciuti associa loro una parola di codice. Se $\mathcal{A}$ è l'alfabeto primario su cui sono costruiti i messaggi da codificare, l'alfabeto secondario su cui sono costruite le parole di codice è $\mathcal{B}$ di numerosità $\beta \geq 2$ (spesso $\mathcal{B} = \{0, 1\}$). Le parole di codice sono vincolate ad essere *a blocco*, il che vuol dire che hanno tutte la stessa lunghezza l. Il codificatore, che si servirà utilmente dell'*albero di codifica* (dell'albero dei messaggi), attua un'applicazione da $\mathcal{M}$ in $\mathcal{B}^l$ (cfr. la figura). Grazie al fatto che la famiglia $\mathcal{M}$ è completa, l'operazione di codifica è univoca e senza ritardi. Per garantire l'univoca decodificabilità basterà richiedere che sia $H^l = |\mathcal{B}|^l \geq |\mathcal{M}| = K + j(K-1)$, ciò che consente di evitare parole di codice ripetute (di costruire un codificatore iniettivo). I codici LV-B basati su famiglie complete e soddisfacienti le modeste richieste che abbiamo posto al codificatore si chiamano essi stessi *completi*: essi garantiscono codificabilità e decodificabilità istantanee e saranno il nostro modello di codice LV-B attendibile.

Attenti, perchè nei codici LV-B problematica è la codifica, e non la decodifica! Notate che il criterio di attendibilità che ci siamo imposti è molto esigente: pretendiamo che la codifica sia istantanea, e non ci accontentiamo che sia univoca, come sarebbe bastato per conservare la simmetria con il caso LV-B.

Per quanto riguarda l'economicità del codice completo, il parametro fondamentale è il suo *tasso* (misurato in H-it), ossia il rapporto fra la lunghezza di blocco l delle parole di codice e la lunghezza media dei messaggi (numero medio di H-it secondari dedicati a ciascun K-it primario):

$$R_j = \frac{l}{E|M|} = \frac{l}{EN_j}$$

D'ora in avanti, *senza perdita di ottimalità* ($\Box$!; conviene avere parole di codice corte), supporremo:

$$l = \lceil \log_\beta |\mathcal{M}| \rceil = \lceil \log_\beta [K + j(K-1)] \rceil$$

Ciò supposto, per calcolare il tasso basta conoscere $\mathcal{M}$, che, in questo senso, *è* il codice (ne è insomma la parte essenziale). Per ottenere codici economici, abbiamo bisogno di alti rapporti di compressione, ossia di tassi bassi, ossia di messaggi mediamente *lunghi*. Il problema di ottimazione degli alberi completi à anche quello di ottimazione dei codici completi: dato j costruisci $\mathcal{M}$ in modo da render massima la lunghezza media EN_j.

C'è una sgradevole rottura di simmetria rispetto al caso B-LV: avremmo dovuto fissare l, simmetrico di n, e poi render ottimo il tasso; purtroppo qui l, che si calcola a partire da j, non è un parametro "libero" per come funziona l'algoritmo che costruisce le famiglie complete.

I *codici di Tunstall* sono particolari codici completi che si ottengono dalle famiglie di Tunstall. Ovviamente:

Teorema: I codici di Tunstall ottenuti con j estensioni risolvono il problema dell'ottimalità dei codici completi.

Esempio 1. Sia $K = 2$, $\beta = 5$, $j = 3$, $P = (\frac{1}{2}, \frac{1}{2})$. I messaggi sono $\mathcal{M}_3 = \{a^3, a^2b, ab, ba, b^2\}$ di probabilità $P^{(3)} = (\frac{1}{8}, \frac{1}{8}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ e di parole di codice quinarie $\{0, 1, 2, 3, 4\}$. La lunghezza media dei messaggi è $\frac{9}{4}$; il tasso in 5-it è $\frac{4}{9}$, ossia un po' meno di mezza cifra quinaria per codificare ciascuna lettera primaria. Figura.

Esempio 2. Sia $K = 3$, $\beta = 3$, $j = 3$, $P = (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$. I messaggi sono
$\mathcal{M}_3 = \{a^3, a^2b, a^2c, ab, ac, ba, b^2, bc, c\}$ oppure $\mathcal{M}'_3 = \mathcal{A}^2$. In entrambi i casi la lunghezza media dei messaggi è 2 e il tasso è 1 (nel secondo caso abbiamo un codice *a blocco*, le lunghezze non variano). Se $\beta = 2$ le nove parole di codice hanno lunghezza 4: è un vero peccato che ben sette quaterne binarie rimangano inutilizzate! Nel caso $\beta = 3$ ci era andata bene e non avevamo perso neppure una coppia ternaria.

I due ultimi esempi ci daranno il destro di fare dei commenti.

Esempio 3. Sia $|\mathcal{A}| = |\mathcal{B}| = 2$, P uniforme, $j = 0, 1$. I due codici di Tunstall sono, rispettivamente, $\mathcal{C}_0 = \{a \to 0, b \to 1\}$ e $\mathcal{C}_1 = \{aa \to 00, ab \to 01, b \to 1\}$, di tassi $R_0 = R_1 = 1$. Il secondo codice, sia pur ottimo per $j = 2$, non va usato perchè più complesso del primo, ma altrettanto economico.

E' un brutto scherzo che ci ha giocato la nostra decisione, peraltro inevitabile, di fissare j come parametro libero; c'è di peggio: l'esempio 4 mostra che a j diversi può corrispodere la stessa lunghezza l delle parole di codice.

Esempio 4. Sia $|\mathcal{A}| = 3$, $|\mathcal{B}| = 2$, $j = 1, 2$. In entrambi i casi $\lceil \log(3+2j) \rceil = 3$; anche senza precisare P, il tasso è peggiore nel secondo caso ($\square$!). Tutte le volte che cresce j senza che aumenti l peggioriamo il tasso e aumentiamo la complessità.

Esempio 5. Sia $|\mathcal{A}| = 3$, $|\mathcal{B}| = 5$, $p_a \approx 1$, $j = 1$. Il codice di Tunstall è
$\mathcal{C}_1 = \{aa \to 0, ab \to 1, ac \to 2, b \to 3, c \to 4\}$, di tasso $R_1 \approx \frac{1}{2}$.
Con la regola di preferenza: "In caso di dubbio segmenta al messaggio più lungo" consideriamo anche il codice non completo ma esauriente (con ritardi di codifica, cfr. la figura): $\mathcal{C}^* = \{aaa \to 0, aa \to 1, a \to 2, b \to 3, c \to 4\}$, di tasso $R^* \approx \frac{1}{3}$. Il secondo codice ha la stessa complessità del primo, ma comprime di più i dati, e dunque *batte il codice di Tunstall.*

E' un'allarmante rottura di simmetria con il caso B-LV, dove i ritardi di decodifica non pagavano: invece nel caso LV-B i ritardi di decodifica talvolta pagano! Il lettore versato in probabilità potrà notare che, nel caso del codice incompleto, la sorgente dei messaggi non è più bernoulliana, a differenza della sorgente delle lettere singole; lo stesso tasso varia nel tempo. Una grana tecnica da non sottovalutare: e in effetti sui codici esaurienti ma incompleti si sa piuttosto poco.

I codici con elementi a lunghezza variabile, come i codici di Huffman o quelli di Tunstall, presentano pesanti problemi ingegneristici di sincronizzazione nel caso di "trasmissione dei dati in tempo reale"; più esattamente i codici B-LV, come quello di Huffman, hanno problemi in fase di decodifica, mentre quelli LV-B in fase di codifica. Di solito si ricorre a "memorie-tampone", o *buffer* di transito, ma per quanto esse siano sovradimensionate, e dunque inutilmente costose, una lunga tratta di lettere improbabili le fa comunque "traboccare" (*overflow*), e con ciò introduce errori in uno schema sulla carta perfetto.

14. Risultati asintotici

Partiamo da un paio di lemmi.

Siano $P = (p_1, p_2, \ldots, p_{|\mathcal{A}|})$ e $Q = (q_1, q_2, \ldots, q_{|\mathcal{A}|})$ due vettori di probabilità sull'alfabeto $\mathcal{A} = \{a_1, a_2, \ldots, a_{|\mathcal{A}|}\}$. Senza reale restrizione assumiamo $p^- = \min p_i > 0$. Se E è un evento di probabilità non nulla p, la quantità $-\log_\beta p$ viene talvolta chiamata l'*auto-informazione* dell'evento E (misurata in β-its). Dunque l'entropia di Shannon $H_\beta(P)$ è una media *pesata* di auto-informazioni; se si media rispetto a Q invece che rispetto a P

si ottiene l'entropia mista $H_\beta(Q\|P)$. Il lemma che segue può venir usato per limitare auto-informazione, entropia e entropia mista quando P è "quasi uniforme" (quando α è $\gg \frac{1}{|\mathcal{A}|}$; p^- and p^+ sono la componente più piccola e più grande di P, p è una componente generica):

Lemma 1. Sia $p^- \geq \alpha p^+$ $(0 < \alpha \leq 1)$. Allora

$\log_\beta |\mathcal{A}| + \log_\beta \alpha \leq -\log_\beta p \leq \log_\beta |\mathcal{A}| - \log_\beta \alpha$

$\log_\beta |\mathcal{A}| + \log_\beta \alpha \leq H_\beta(P) \leq \log_\beta |\mathcal{A}|$

$\log_\beta |\mathcal{A}| + \log_\beta \alpha \leq H_\beta(Q\|P) \leq \log_\beta |\mathcal{A}| - \log_\beta \alpha$

Dimostrazione. Si ha $p^- \leq \frac{1}{|\mathcal{A}|} \leq p^+$. Grazie alle disuguaglianze:

$p \leq p^+ \leq \frac{1}{\alpha} p^- \leq \frac{1}{\alpha|\mathcal{A}|}$, e

$p \geq p^- \geq \alpha p+ \geq \alpha \frac{1}{|\mathcal{A}|}$

limitiamo l'auto-informazione. Mediando rispetto a P e a Q si trovano le limitazioni per le due entropie. □

Nel caso dell'entropia abbiamo rinunciato alla limitazione superiore $\log_\beta |\mathcal{A}| - \log_\beta \alpha$, perchè sappiamo far di meglio: $H_\beta(P) \leq \log_\beta |\mathcal{A}|$. Le limitazioni inferiori possono essere molto deboli, perfino negative se $\alpha < \frac{1}{|\mathcal{A}|}$.

Costruiamo un albero con j estensioni; la P può venir estesa ai suoi nodi per stazionarietà e mancanza di memoria. Sia $P^{(j)}$ il vettore di probabilità delle $K + j(K-1))$ foglie f_s dell'albero. Nel lemma, che generalizza l'uguaglianza $H(P^n) = nH(P)$, EN_j è la profondità media delle foglie, $EN_j = \sum_s P^{(j)}(f_s)n(f_s)$.

Lemma 2. $H_\beta(P^{(j)}) = EN_j H_\beta(P)$

Dimostrazione. Per induzione su $j \geq 0$, dopo aver osservato che $EN_{j+1} = EN_j + p^*$, dove p^* è la probabilità del nodo esteso $(EN_0 = 1)$:

$H_\beta(P^{(j+1)}) = H_\beta(P^{(j)}) + p^* \log_\beta p^* - \sum_i (p^* p_i) \log_\beta (p^* p_i) =$

$= H_\beta(P^{(j)}) + p^* H_\beta(P) = (EN_j + p^*) H_\beta(P) = EN_{j+1} H_\beta(P)$ □

L'esercizio che segue generalizza il lemma al caso in cui la profondità media delle foglie, $EN_j^{(Q)}$, sia calcolata rispetto a una probabilità Q diversa da P ($Q^{(j)}$ diversa da $P^{(j)}$): $EN_j^{(Q)} = \sum_s Q^{(j)}(f_s)n(f_s)$. La dimostrazione ricalca passo dopo passo quella del lemma.

Esercizio. $H_\beta(Q^{(j)}\|P^{(j)}) = EN_j^{(Q)} H_\beta(Q\|P)$

Il lemma di Tunstall è cruciale per dimostrare l'ottimalità asintotica dei codici di Tunstall. Afferma grosso modo che il vettore delle probabilità delle foglie di Tunstall è "quasi uniforme" se $j \gg 1$ (se $p^- \gg \frac{1}{K+j(K-1)}$).

Lemma di Tunstall. Siano p_j^- and p_j^+ la probabilità delle foglie minima e massima in un codice di Tunstall ottenuto con j estensioni. Sia p^- la probabilità minima delle lettere singole ($p^- = p_0^-$). Allora:

$$p_j^- \geq p^- p_j^+$$

Dimostrazione. Ricorriamo di nuovo a un'induzione. Per $j = 0$ l'asserto è ovviamente vero. Sia vero per j e fissiamo l'attenzione su $j+1$. Nota che p_j^+ è la probabilità della foglia estesa, poichè stiamo estendendo alla Tunstall. Ci sono due casi: il nuovo minimo coincide con il vecchio, ossia $p_{j+1}^- = p_j^- \geq p^- p_j^+$ (la disuguaglianza è l'ipotesi induttiva), oppure il nuovo minimo è raggiunto da una nuova foglia, e allora $p_{j+1}^- = p^- p_j^+$ (le nuove foglie hanno probabilità del tipo pp_j^+, dove p è la probabilità di una generica lettera singola). Dunque in entrambi i casi $p_{j+1}^- \geq p^- p_j^+$. Ciò basta poichè si ha $p_j^+ \geq p_{j+1}^+$ (nel nuovo albero le probabilità o rimangono tal quali, o diminuiscono). □

Teoremi sul tasso

Il tasso in β-it di un codice completo costruito con j estensioni è

$$R_j = \frac{\lceil \log_\beta (K + j(K-1)) \rceil}{EN_j}$$

Poichè (lemma 2) $EN_j = H_\beta(P) H_\beta(P^{(j)}) \leq H_\beta(P) \log_\beta (K + j(K-1))$, si ha:

Teorema inverso. Il tasso R_j di un codice β-ario completo è $\geq H_\beta(P)$.

Anzi, $R_j \geq \frac{\lceil \log_\beta (K+j(K-1)) \rceil}{\log_\beta (K+j(K-1))} H_\beta(P)$.

Neanche stavolta l'entropia viene battuta!

Se il codice è di Tunstall $EN_j = H_\beta(P) H_\beta(P^{(j)}) \geq H_\beta(P)(\log_\beta (K + j(K-1)) + \log_\beta p^-))$ (usa il lemma 1 e il lemma di Tunstall), per cui:

Teorema diretto. Il tasso R_j di un codice β-ario di Tunstall è $\leq \gamma_j H_\beta(P)$, dove γ_j tende a 1.

Prendi $\gamma_j = \frac{\lceil \log_\beta (K+j(K-1)) \rceil}{\log_\beta (K+j(K-1)) + \log_\beta p^-}$; come immediato corollario si ha il seguente:

Teorema asintotico. Il tasso R_j di un codice β-ario di Tunstall tende a $H_\beta(P)$ al divergere del numero j delle estensioni.

Anche stavolta il tasso ottimo tende all'entropia!

Esercizio: miscodifica. Il tasso di un codice β-ario di Tunstall costruito per la sorgente bernoulliana P, ma usato con la sorgente bernoulliana Q, tende all'entropia mista $H_\beta(Q\|P)$ al divergere del numero j delle estensioni.

Dimostrazione. Il "mistasso" è $R_j^{(Q)} = \frac{\lceil \log_\beta(K+j(K-1)) \rceil}{EN_j^{(Q)}}$. Ricalca le dimostrazioni dei teoremi precedenti sostituendo all'entropia pura l'entropia mista.

IV. CODIFICA UNIVERSALE

15. Classi di composizione, o tipi

La divagazione stavolta ha natura combinatoria: a essa ritorneremo nel prossimo capitolo.

Se x è un'n-upla costruita sull'alfabeto $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$, la sua *composizione* è il vettore intero a K componenti $(n_1, n_2, \ldots, n_K)$ dove n_i indica il numero di *presenze* (*occurrences*) in x della lettera a_i. Due n-uple che hanno la stessa composizione sono permutazione l'una dell'altra; avere la stessa composizione è una relazione di equivalenza fra n-uple che ripartisce $\mathcal{A}^n$ in classi di equivalenza. Tali classi si chiamano *classi di composizione* o più concisamente *tipi*. Insomma un tipo T comprende un'n-upla x assieme a tutte le sue permutazioni; esso è individuato dalla K-upla intera $(n_1, n_2, \ldots, n_K)$, $n_i \geq 0$, $n_1 + n_2 + \ldots + n_K = n$. La numerosità di T, com'è facile vedere, è un coefficiente multinomiale (per ora lo diciamo solo per inciso, l'importanza di questo fatto apparirà chiara appena nel prossimo capitolo):

$$|T| = \binom{n}{n_1, n_2, \ldots, n_K}$$

Il numero dei tipi in $\mathcal{A}^n$, $\gamma_n = \gamma_n(K)$, è subito maggiorabile:

$$\gamma_n \leq (n+1)^K$$

($\square$! Cenno: una composizione è una K-upla di interi tratti dall'"alfabeto" $\{0, 1, \ldots, n\}$ e vincolati a sommare ad n; se dimentichiamo il vincolo ... Per scrupolo, in chiusura di sezione daremo il numero esatto dei tipi di $\mathcal{A}^n$.)

Dunque γ_n cresce al più polinomialmente con n, mentre il numero di tutte n-uple ha crescita esponenziale. In termini euristici: in un ambiente nel quale le crescite sono tipicamente esponenziali, esplosive, *ci sono pochi tipi distinti*!

Esercizio. Il numero esatto dei tipi distinti è :

$$\gamma_n = \binom{n+K-1}{K-1}$$

In particolare, nel caso binario: $\gamma_n = \gamma_n(2) = n + 1$.
Cenno: Si tratta di contare le K-uple sull'alfabeto $\{0, 1, \ldots, n\}$ che sommano a n. Ad esempio, per $\mathcal{A} = \{a, b, c\}$, $K = 3$, $n = 4$, le composizioni $\{2, 1, 1\}$ e $\{4, 0, 0\}$ sono identificate dalle "configurazioni" a stelle e barre $**|*|*$ e $****||$, con n stelle e $K - 1$ barre; quante sono queste configurazioni?

Ne viene (□!) che γ_n è un polinomio in n di grado $K-1 = |\mathcal{A}|-1$.

L'esercizio che segue verrà fra a buon uso nella prossima sezione.

Esercizio. Sia X^n un'n-upla aleatoria di lettere emesse da una sorgente bernoulliana (stazionaria e senza memoria) retta dalla probabilità P, e sia J il numero d'ordine aleatorio del tipo cui essa appartiene: $X^n \in T_J$. Allora $H(X^n|J) = \sum_j P^n(T_j)\log|T_j|$. ($J$ è una funzione deterministica di X^n, per cui $H(J|X^n) = 0$; $1 \le j \le \gamma_n$; la numerazione dei tipi da 1 a γ_n è arbitraria.)
Cenno: Le sequenze di un tipo hanno tutte la stessa probabilità P^n, per cui $H(X^n|J = j) = \log|T_j|$.

16. Codici multinomiali e codifica universale

Nei codici di Shannon-Fano, di Huffman e di Tunstall l'entropia mista ha un ruolo fisso nei teoremi di miscodifica. Ciò fa congetturare che essa conservi lo stesso ruolo tutte le volte che si ha a che fare con codici asintoticamente ottimi. Questa congettura si rivelerà clamorosamente sbagliata.

Supponiamo di voler comprimere l'estensione n-esima della sorgente bernoulliana di cui all'esercizio tramite un codice B-LV; l'alfabeto secondario su cui sono costruite le parole di codice abbia numerosità β. Preliminarmente numeriamo i tipi di $\mathcal{A}^n$ da 1 a γ_n, anzi da 0 a $\gamma_n - 1$, e all'interno di ciascun tipo T numeriamo le sue n-uple da 0 a $|T|-1$.

Tali numerazioni sono operazioni davvero preoccupanti: il lettore avrà già capito che il codice di cui stiamo per parlare è di interesse solo teorico. E tuttavia non resterà deluso!

A ciascuna n-upla x emessa dalla sorgente il codificatore di sorgente associa una parola di codice $c(x)$ composta da un prefisso e da un suffisso, $c(x) = c^p(x)c^s(x)$. Il prefisso dice il numero d'ordine del tipo cui x appartiene, il suffisso dice il numero d'ordine di x all'interno del tipo. Possiamo supporre che il prefisso $c^p(x)$ sia una stringa di lunghezza fissa $l^p = \lceil \log_\beta \gamma_n \rceil$ e il suffisso $c^s(x)$ sia una stringa di lungheza fissa all'interno del tipo $l^s_j = \lceil \log_\beta |T_j| \rceil$ (□ ! trascrivi le numerazioni in β-ario, aggiungendo se serve zeri in testa).

Esempio. Prendi $K = 2$, $n = 3$. I tipi sono allora $\{aaa\}$, $\{aab, aba, baa\}$, $\{abb, bab, bba\}$, e $\{bbb\}$, numerati 00, 01, 10 e 11, rispettivamente; al tipo numero 00 corrisponde la composizione $(3, 0)$, al tipo numero 01 corrisponde la composizione $(2, 1)$, ecc. All'interno del tipo le sequenze vengono numerate con il solo 0 se il tipo è un singoletto, altrimenti da 00 a 10. La parola di codice di aab è allora 0100, i primi due bit di prefisso, gli ultimi due di suffisso.

La lunghezza media delle parole di codice è :

$$EL_n = \sum_x P^n(x)l(c(x)) = \lceil \log_\beta \gamma_n \rceil + \sum_j P^n(T_j)\lceil \log_\beta |T_j| \rceil < \log_\beta \gamma_n + H(X^n|J) + 2$$

(rammenta l'esercizio; l'entropia è misurata in β-it). Dunque, per il tasso R_n misurato in β-it del codice multinomiale, l'entropia condizionata $H(X^n|J)$ non potendo superare l'entropia assoluta $H(X^n) = nH(P)$:

$$R_n = \frac{EL_n}{n} < \frac{\log_\beta \gamma_n}{n} + H(P) + \frac{2}{n}$$

Ciò prova l'ottimalità asintotica dei codici multinomiali: $R_n \to H(P)$. (I codici multinomiali sono codici a prefisso, come il lettore avrà già notato, per cui è ovvio a priori che $R_n \geq H(P)$.)

Il lettore che affronti il problema della miscodifica resterà stupito: nella costruzione del codice multinomiale il vettore di probabilità P proprio non c'entra, per cui lo stesso identico codice è asintoticamente ottimo per *qualunque* sorgente bernoulliana su $\mathcal{A}$! Per sottolineare questo fatto così poco intuitivo, diciamo che il codice multinomiale è un codice *universale*, o, per esser più precisi, che è un codice universale per la classe delle sorgenti bernoulliane di alfabeto $\mathcal{A}$.

L'esistenza dei codici universali è davvero un fatto contro-intuitivo: si pensi a tutte le nostre chiacchiere (nostre e di Samuel Morse) sulla necessità di assegnare parole di codice corte ai messaggi molto probabili!
Rimane aperto il problema di trovare codici che siano universali e insieme pratici. La risposta più convincente, o almeno più nota, sono i codici di Ziv-Lempel, o di Lempel-Ziv, largamente applicati. Quando si usa un codice di sorgente per trascrivere e insieme comprimere i dati, la statistica della sorgente che si vuol codificare di solito non è nota. Ci sono tre soluzioni: la codifica *a doppia passata* (il testo viene esaminato una prima volta per capirne le caratteristiche statistiche, poi si torna indietro e lo si codifica: di fatto i codici di Huffman è così che vengono usati); la codifica *adattativa* (via via che la statistica del testo da codificare si fa più chiara, il codice si automodifica; esistono varianti adattative sia del codice di Huffman, sia del codice di Tunstall); la codifica *universale* (non c'è bisogno di sapere quale sia la statistica del testo, ciò che risolve il problema alla radice).
Al posto di "universale", gli statistici preferiscono usare il loro termine gergale "robusto".