Teoria dell'Informazione/RTI **Anno Accademico 2022–2023**

Lezione 5

Ugo Vaccaro

Il risultato principale che abbiamo scoperto nella lezione scorsa è il seguente: data una sorgente DSSM $X_1, X_2, \ldots, X_i, \ldots$ con alfabeto sorgente X e distribuzione di probabilità $P = \{P(x) : x \in \mathsf{X}\}$, *ogni* funzione di codifica $\mathsf{c} : X \to \{0,1\}^*$ unicamente decifrabile ha lunghezza media che soddisfa la diseguaglianza

$$\sum_{x \in X} P(x)\ell(x) \geq H(P),$$

dove $\ell(x)$ è la lunghezza della codifica $\mathsf{c}(x)$ del simbolo sorgente $x \in \mathsf{X}$.

Ci chiediamo ora se esistono codifiche UD la cui lunghezza media è "vicina" a $H(P)$. La risposta è si, ed abbiamo bisogno di una nuova definizione e di un risultato intermedio per giustificarla.

Definizione 1 *Una codifica* $\mathsf{c} : \mathsf{X} \to \{0,1\}^*$ *è detta* prefisso *se* $\forall x, y \in \mathsf{X}$ *con* $x \neq y$*, la parola codice* $\mathsf{c}(x)$ non è *inizio (prefisso) della parola codice* $\mathsf{c}(y)$*.*

Esempio 1 *Se* $\mathsf{X} = \{a, b, c\}$*, la codifica che assegna* $\mathsf{c}(a) = 101, \mathsf{c}(b) = 01, \mathsf{c}(c) = 1$ *non è prefisso in quanto la codifica di* $\mathsf{c}(c)$ *è inizio della codifica di* $\mathsf{c}(a)$*. Invece la codifica che assegna* $\mathsf{c}(a) = 001, \mathsf{c}(b) = 01, \mathsf{c}(c) = 1$ *è prefisso.*

É chiaro che una codifica prefisso è anche UD, in quanto leggendo da sinistra a destra una qualunque sequenza codice, appena individuiamo una parola codice siamo certi che essa è quella corretta, in quanto non essendo inizio di nessun altra parola codice non vi è alcuna ambiguità nella sua individuazione. In più, le codifiche prefisso hanno ulteriori proprietà utili dal punto di vista pratico, ad esempio, impediscono che si creino lunghi ritardi nella decodifica. Per illustrare il concetto, immaginiamo di avere una codifica $\mathsf{c} : \mathsf{X} \to \{0,1\}^*$ di questo tipo: $\mathsf{X} = \{a, b, c\}$, $\mathsf{c}(a) = 1, \mathsf{c}(b) = 10, \mathsf{c}(c) = 00$. Questa codifica si può vedere essere unicamente decifrabile (quindi da ogni sequenza di parole codice è possibile dedurre la sequenza sorgente originaria) ma non è prefisso in quanto $(b) = c(a)0$. Immaginiamo ora ricevere la sequenza binaria $100000000000\ldots$. Essa si può decomporre nelle parole codice 1 00 00 00 00 00 0… che darebbe origine alla sequenza sorgente che inizia con *accccc*, oppure si può decomporre nelle parole codice 10 00 00 00 00 00… che darebbe origine alla sequenza sorgente che inizia con *bccccc*. Per sapere se l'inzio corretto della decodifica è con la lettera a o con la lettera b, dovremmo attendere la fine della della sequenza binaria (ovvero, dovremmo prima leggerla tutta). Se il numero di 0 in essa è pari, allora l'inizio corretto è con la lettera sorgente a, altrimenti l'inizio corretto è con la lettera b. Ovviamente, ciò è inaccettabile dal punto di vista pratico.

Il seguente risultato và sotto il nome di Diseguaglianza di Kraft.

Teorema 1 *Siano* $\ell_1, \ldots, \ell_m$ *interi tali che* $\sum_{i=1}^m 2^{-\ell_i} \leq 1$*. Allora, per ogni alfabeto sorgente* $\mathsf{X} = \{x_1, \ldots, x_m\}$ *è possibile costruire una codifica* $\mathsf{c} : \mathsf{X} \to \{0,1\}^*$ *binaria, prefisso, con parole codice di lunghezza* $\ell_1, \ldots, \ell_m$*.*

Dimostrazione. Sia $L = \max\{\ell_1, \ldots, \ell_m\}$. Dall'ipotesi sappiamo che

$$2^L \geq \sum_{i=1}^m 2^{L-\ell_i} = 2^{L-\ell_1} + 2^{L-\ell_2} + \ldots + 2^{L-\ell_m},$$

ovvero

$$2^L - 2^{L-\ell_1} - 2^{L-\ell_2} - \ldots - 2^{L-\ell_m} \geq 0. \tag{1}$$

Sia $\ell_1 \leq \ldots \leq \ell_m$. Per costruire la codifca prefisso promessa dal Teorema, procediamo nel modo seguente. Tra tutte le 2^L sequenze binarie di lunghezza L, scegliamone una in modo arbitrario e prendiamoci i suoi primi ℓ_1 bits per formare la prima parola codice $\mathsf{c}(x_1)$ di lunghezza ℓ_1. Ciò ci impedirà di usare (per future scelte) *tutte* le sequenze binarie che hanno i primi ℓ_1 uguali alla parola scelta (ciò al fine di garantire la condizione prefisso, ovvero che la parola codice $\mathsf{c}(x_1)$ che abbiamo scelto per codificare x_1, di lunghezza ℓ_1, non sia inizio di nessun altra parola). Tali sequenze che eliminiamo saranno in numero di $2^{L-\ell_1}$. Quindi, ne avevamo inizialmente a disposizione 2^L, dopo aver scelto la prima parola ne rimangono a nostra disposizione solo $2^L - 2^{L-\ell_1}$, che in virtù della (1) è un numero > 0.

In altri termini, possiamo scegliere ancora un'altra sequenza lunga L e prenderne i primi primi ℓ_2 bits per formare la seconda parola codice $\mathsf{c}(x_2)$ di lunghezza ℓ_2. Di nuovo, ciò ci impedirà di usare (per future scelte) *tutte* le sequenze binarie che hanno i primi ℓ_2 uguali alla parola scelta (ciò sempre al fine di garantire la condizione prefisso, ovvero che la parola codice $\mathsf{c}(x_2)$ che abbiamo scelto per codificare x_2, di lunghezza ℓ_2, non sia inizio di nessun' altra parola). Tali sequenze che eliminiamo saranno in numero di $2^{L-\ell_2}$. Quindi, dalle 2^L di partenza, dopo aver scelto la prima e seconda parola ne rimangono a nostra disposizione solo $2^L - 2^{L-\ell_1} - 2^{L-\ell_2}$, che in virtù della (1) è *ancora* un numero > 0. Possiamo ovviamente iterare sino a quando non abbiamo completato la costruzione di tutte le m parole $\mathsf{c}(x_1), \mathsf{c}(x_2), \ldots, \mathsf{c}(x_m)$ della codifica. □

Vediamo un esempio.

Sia $\mathsf{X} = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ e $\ell_1 = 2, \ell_2 = 2$, $\ell_3 = \ell_4 = \ell_5 = \ell_6 = 4$. Vale che

$$\sum_{i=1}^{6} 2^{-\ell_i} = 2^{-1} + 2^{-2} + 2^{-4} + 2^{-4} + 2^{-4} + 2^{-4} = 1$$

1. Scegliamo $\mathsf{c}(x_1) = 0$ ed eliminiamo da $\{0,1\}^4$ tutte le 2^3 sequenze che iniziano con 0. Ce ne rimangono $2^4 - 2^3 > 0$.

2. Scegliamo $\mathsf{c}(x_2) = 10$ ed eliminiamo da $\{0,1\}^4$ tutte le 2^2 sequenze che iniziano con 10. Ce ne rimangono $2^4 - 2^3 - 2^2 > 0$.

3. Scegliamo $\mathsf{c}(x_3) = 1100$ ed eliminiamo da $\{0,1\}^4$ tutte le 2^0 sequenze che iniziano con 1100. Ce ne rimangono $2^4 - 2^3 - 2^2 - 2^0 > 0$.

4. Scegliamo $\mathsf{c}(x_4) = 1101$ ed eliminiamo da $\{0,1\}^4$ tutte le 2^0 sequenze che iniziano con 1101. Ce ne rimangono $2^4 - 2^3 - 2^2 - 2^0 - 2^0 > 0$.

5. Scegliamo $\mathsf{c}(x_5) = 1110$ ed eliminiamo da $\{0,1\}^4$ tutte le 2^0 sequenze che iniziano con 1110. Ce ne rimangono $2^4 - 2^3 - 2^2 - 2^0 - 2^0 - 2^0 > 0$.

6. Scegliamo $\mathsf{c}(x_6) = 1111$ ed eliminiamo da $\{0,1\}^4$ tutte le 2^0 sequenze che iniziano con 1111. Ce ne rimangono $2^4 - 2^3 - 2^2 - 2^0 - 2^0 - 2^0 = 0$.

La codifica che abbiamo costruito sarà: $\mathsf{c}(x_1) = 0, \mathsf{c}(x_2) = 10, \mathsf{c}(x_3) = 1100, \mathsf{c}(x_4) = 1101, \mathsf{c}(x_5) = 1110, \mathsf{c}(x_6) = 1111$ che è chiaramente **prefisso**.

La Diseguaglianza di Kratf appena vista fà il paio con il risultato visto nella scorso lezione, ovvero la Diseguaglianza di McMillan. Infatti la Diseguaglianza di McMillan affermava che se $\ell_1, \ldots, \ell_m$ sono le lunghezze delle parole codice di una generica codifica binaria UD, allora $\sum_{i=1}^{n} 2^{-\ell_i} \leq 1$. Ma, d'altra parte, se $\sum_{i=1}^{n} 2^{-\ell_i} \leq 1$ abbiamo appena appreso che possiamo costruire una codifica binaria prefisso con lunghezze delle parole codice pari a $\ell_1, \ldots, \ell_m$. Detto in altri termini, dato una generica codifica UD, possiamo *sempre* trovare una codifica prefisso (che abbiamo visto essere una proprietà molto utile in pratica) con le *stesse* lunghezze di parola codice di quella UD (e quindi

con la *stessa* lunghezza media). Ovverosia, non si perde di ottimalità se ci limitiamo a minimizzare la lunghezza media di parola codice all'interno della ristretta (ma *più* utile) classe di codifiche prefisso.

Ritorniamo alla nostra sorgente DSSM con distribuzione di probabilità $P = \{P(x) : x \in \mathsf{X}\}$. Esaminando con attenzione la prova della lezione scorsa che ci ha permesso di verificare che $\sum_{x\in\mathsf{X}} P(x)\ell(x) \geq H(X)$, per ogni codifica UD, possiamo affermare che vale la uguaglianza $\sum_{x\in\mathsf{X}} P(x)\ell(x) = H(X)$ (che è ovviamente il meglio in cui possiamo sperare) se e solo se $\sum_{x\in\mathsf{X}} P(x) \log \frac{P(x)}{2^{-\ell(x)}} = 0$, ovvero se e solo se le lunghezze $\ell(x)$ della codifica soddisfano la relazione $2^{-\ell(x)} = P(x)$ (o equivalentemente $\ell(x) = \log \frac{1}{P(x)}$) per ogni $x \in \mathsf{X}$. Il problema è che la quantità $\log \frac{1}{P(x)}$ non è in generale un intero. Quindi, in generale, un codice UD con lunghezze di parole codice $\ell(x) = \log \frac{1}{P(x)}$ potrebbe non esistere. Vediamo se ne esiste uno con lunghezze "vicine" al valore $\log \frac{1}{P(x)}$, ovvero se esiste un codice UD con lunghezze di parola codice $\ell(x) = \left\lceil \log \frac{1}{P(x)} \right\rceil$. Verifichiamo ciò usando la diseguaglianza di Kraft. Si ha

$$\sum_{x\in\mathsf{X}} 2^{-\ell(x)} = \sum_{x\in\mathsf{X}} 2^{-\left\lceil \log \frac{1}{P(x)} \right\rceil} \leq \sum_{x\in\mathsf{X}} 2^{-\log \frac{1}{P(x)}} = \sum_{x\in\mathsf{X}} P(x) = 1.$$

Pertanto, dalla diseguaglianza di Kraft sappiamo che esiste una codifica prefisso, e quindi *a fortiori* UD in cui ogni $x \in \mathsf{X}$ ha associato una parola codice lunga $\ell(x) = \left\lceil \log \frac{1}{P(x)} \right\rceil$. Calcoliamo la sua lunghezza media. Si ha

$$\sum_{x\in\mathsf{X}} P(x)\ell(x) = \sum_{x\in\mathsf{X}} P(x) \left\lceil \log \frac{1}{P(x)} \right\rceil < \sum_{x\in\mathsf{X}} P(x) \left(\log \frac{1}{P(x)} + 1 \right) = H(P) + 1.$$

Possiamo quindi riassumere ciò che abbiamo provato, nella lezione precedente ed in questa, nel seguente importante risultato:

Teorema 2 *Per ogni codifica binaria* $\mathsf{c} : \mathsf{X} \to \{0,1\}^*$ *UD dei simboli di una sorgente DSSM con distribuzione* $P = \{P(x) : x \in \mathsf{X}\}$ *vale che la sua lunghezza media soddisfa la diseguaglianza*

$$\sum_{x\in\mathsf{X}} P(x)\ell(x) \geq H(P).$$

Inoltre, esiste una codifica binaria $\mathsf{c} : \mathsf{X} \to \{0,1\}^*$ *prefisso (e quindi UD) dei simboli della sorgente la cui lunghezza media soddisfa la diseguaglianza*

$$\sum_{x\in\mathsf{X}} P(x)\ell(x) < H(P) + 1.$$

Cosa succede se invece di assegnare parole codice a singole lettere sorgenti (e poi codificare sequenze di lettere sorgenti attraverso la concatenazione delle codifiche delle singole lettere che appaiono nella sequenza) assegniamo parole codice *direttamente* a sequenze sorgenti? In tal caso, è come se, invece di avere una sorgente che emette lettere in X emettesse "superlettere" (ovvero sequenze) in X^k. Qual è l'entropia di questa nuova sorgente? Denotiamo tale sorgente con $\mathbf{X}^k$. Visto che essa emette sequenze $x_1 \dots x_k \in \mathsf{X}^k$, ciascuna con probabilità

$P(x_1 \dots x_k) = \prod_{i=1}^{k} P(x_i)$ (a causa del fatto che la sorgente è DSSM), la sua entropia $H(\mathbf{X}^k)$ sarà

$$\begin{aligned}
H(\mathbf{X}^k) &= \sum_{x_1 \in \mathsf{X}, \dots, x_k \in X} P(x_1 \dots x_k) \log \frac{1}{P(x_1 \dots x_k)} \\
&= \sum_{x_1 \in \mathsf{X}, \dots, x_k \in X} P(x_1 \dots x_k) \log \frac{1}{\prod_{i=1}^{k} P(x_i)} \\
&= \sum_{x_1 \in \mathsf{X}, \dots, x_k \in X} P(x_1 \dots x_k) \left(\log \frac{1}{P(x_1)} + \dots + \log \frac{1}{P(x_k)} \right) \\
&= \sum_{x_1 \in \mathsf{X}, \dots, x_k \in X} P(x_1 \dots x_k) \log \frac{1}{P(x_1)} + \dots + \sum_{x_1 \in \mathsf{X}, \dots, x_k \in X} P(x_1 \dots x_k) \log \frac{1}{P(x_k)} \\
&= \sum_{x_1 \in \mathsf{X}} P(x_1) \log \frac{1}{P(x_1)} + \dots + \sum_{x_k \in \mathsf{X}} P(x_k) \log \frac{1}{P(x_k)} \\
&= kH(P).
\end{aligned}$$

Dal Teorema 2 applicato alla sorgente $\mathbf{X}^k$ sappiamo che per ogni codifica binaria $\mathsf{c} : \mathsf{X}^k \to \{0,1\}^*$ UD vale che

$$\sum_{\mathbf{x} \in \mathsf{X}^k} P(\mathbf{x}) \ell(\mathbf{x}) \geq H(\mathbf{X}^k) = kH(P).$$

Inoltre esiste una codifica $\mathsf{c} : \mathsf{X}^k \to \{0,1\}^*$ che soddisfa la condizione prefisso la cui lunghezza media soddisfa

$$\sum_{\mathbf{x} \in \mathsf{X}^k} P(\mathbf{x}) \ell(\mathbf{x}) < H(\mathbf{X}^k) + 1 = kH(P) + 1.$$

La quantità $\sum_{\mathbf{x} \in \mathsf{X}^k} P(\mathbf{x}) \ell(\mathbf{x})$ rappresenta il numero medio di bits che la codifica con lunghezze $\ell(\mathbf{x})$ usa per per ogni *sequenza* sorgente $x \in \mathsf{X}^k$. Se vogliamo il numero medio di bits che la codifica usa per ogni simbolo sorgente $x \in \mathsf{X}$, occorre dividere il tutto per k. Otteniamo quindi

Teorema 3 *Data una sorgente DSSM con alfabeto sorgente* X *e distribuzione di probabilità* $P = \{P(x) : x \in \mathsf{X}\}$*, per ogni codifica binaria* $\mathsf{c} : \mathsf{X}^k \to \{0,1\}^*$ *UD vale che il il numero medio di bits che la codifica usa per ogni simbolo sorgente* $x \in \mathsf{X}$ *soddisfa la diseguaglianza*

$$\frac{1}{k} \sum_{\mathbf{x} \in \mathsf{X}^k} P(\mathbf{x}) \ell(\mathbf{x}) \geq H(P).$$

Inoltre, esiste una codifica $\mathsf{c} : \mathsf{X}^k \to \{0,1\}^*$ *che soddisfa la condizione prefisso e che usa un numero di bits per simbolo sorgente per cui*

$$\frac{1}{k} \sum_{\mathbf{x} \in \mathsf{X}^k} P(\mathbf{x}) \ell(\mathbf{x}) < H(P) + \frac{1}{k}.$$

Per $k \to \infty$ ritroviamo più o meno lo stesso risultato della prima lezione, ovvero che per codificare le emissioni della sorgente in modo da poter recuperare dalla codifica ciò che è stato emesso ci occorrono (e bastano) in media $\approx H(P)$ bits per simbolo sorgente, e ciò anche nel caso in cui *non* vogliamo errori nella decodifica, e nel caso di codifiche a lunghezza variabile.

Potrebbe sembrare che il risultato appena provato sia migliore di quello visto nella prima lezione, in quanto in quel caso avevamo codifiche che ammettevano una probabilità di errore (seppur piccola). Le codifiche studiate in questa lezione hanno invece probabilità di errore esattamente uguale a 0. La differenza è le codifiche qui

studiate assegnano *ad ogni* sequenza che la sorgente può emettere una distinta parola codice, mentre nella prima assegnavamo parole codice distinte ad un sottoinsieme di tutte le sequenze sorgenti. Di fatto, però, abbiamo eliminato un problema (cioè la possibilità di avere sequenze sorgenti su cui sbagliavamo la decodifica) ma ne abbiamo creato un altro. Ricordiamo che qui assegniamo a ciascuna sequenza $\mathbf{x} \in X^k$ di probabilità $P(\mathbf{x})$ una parola codice lunga $\ell(\mathbf{x}) = \left\lceil \log \frac{1}{P(\mathbf{x})} \right\rceil$. Ora, se $P(\mathbf{x})$ è molto piccola, la quantità $\left\lceil \log \frac{1}{P(\mathbf{x})} \right\rceil$ è molto grande per cui occorrerà, ad esempio, leggere molti bits prima di poter decodificare. Riassumendo, abbiamo cancellato un "difetto" (cioè, di avere probabilità di errore piccolo) e lo abbiamo sostituito con uno differente (cioè di avere una probabilità piccola di incorrere in ritardi molto grandi nella decodifica).