APPUNTI TEORIA DELL'INFORMAZIONE

Andrea Cosentino

26 aprile 2024

Indice

1 Lezione I **4**
1.1 Introduzione 4
1.2 Una visione d'insieme 5
1.3 Modellazione 6
1.4 Problema codifica sorgente 8
1.5 Limitazioni 9

2 Lezione II **10**
2.1 Limitazioni 10
2.2 Codici istantanei 11

3 Lezione III **15**
3.1 Codice di Shannon 15
3.2 Entropia 16
3.3 Metodo di Huffman 18

4 Lezione IV **20**
4.1 Proprietà Entropia 20
4.1.1 Cambio di base del logaritmo 20
4.1.2 Entropia binaria 21
4.1.3 Disuguaglianze elementari 21
4.1.4 Upper bound entropia 22
4.2 Entropia relativa 23
4.3 Legame tra valore atteso ed entropia 24
4.4 Sardinas-Patterson 26

5 Lezione V **27**
5.1 Upper bound valore atteso 27

5.2 Primo teorema di Shannon 29
5.3 Approssimare il modello 30

6 Lezione VI **32**
6.1 Algoritmo di Huffman 32
6.2 Codici di Huffman 33

7 Lezione VII **37**
7.1 Termine dimostrazione lezione VI 37
7.2 Disuguaglianza di Kraft-McMillan 38

8 Lezione VIII **41**
8.1 Esercizi di probabilità 41
8.2 Numero di bit necessari 42
8.3 Esercizi su entropia 43

9 Lezione IX **46**
9.1 Informazione mutua 46
9.2 Data processing inequality 48
9.3 Disuguaglianza di Fano 49

10 Lezione X **50**
10.1 Canale 50
10.1.1 Canale binario senza rumore 51
10.1.2 Canale binario simmetrico 51
10.2 Capacità del canale 52

11 Lezione XI **54**
11.1 Canale binario a cancellazione 54
11.2 Codice di Fano 56

12 Lezione XII **58**
12.1 Introduzione 58
12.2 Esercizi di probabilità 59
12.3 Codici di correzione 60
12.3.1 Single parity check code 60
12.3.2 Codice ASCII 60
12.3.3 Codici pesati 61
12.3.4 Codici (M,N) 63

12.4 II Teorema di Shannon 65

13 Lezione XIII **66**
13.1 Codici di rilevazione errori 66
13.1.1 ISBN-10 66
13.1.2 UPC 67
13.2 Codici di rilevazione e correzione 68
13.2.1 Codice a ripetizione tripla 69
13.2.2 Codici di tipo (n,k) 70

14 Lezione XIV **74**
14.1 Campi di Galois 74
14.1.1 Generatore 76
14.1.2 Definizione campi di Galois 78
14.2 Codici ciclici 81
14.2.1 Matrice generatrice 82

15 Lezione XV **84**
15.1 Correzione codice ciclico 84
15.1.1 Algoritmo di correzione 84
15.2 Codici BCH 87
15.2.1 Ampliamento algebrico 88

Nel cielo non c'è niente,
le cose importanti stanno per terra.
- Uno studente di teoria dell'informazione.

Capitolo 1

Lezione I

1.1 Introduzione

Durante il corso del '900 diverse figure hanno contribuito allo sviluppo delle fondamenta della disciplina. Tra queste ricordiamo coloro che ne vengono considerati i padri:

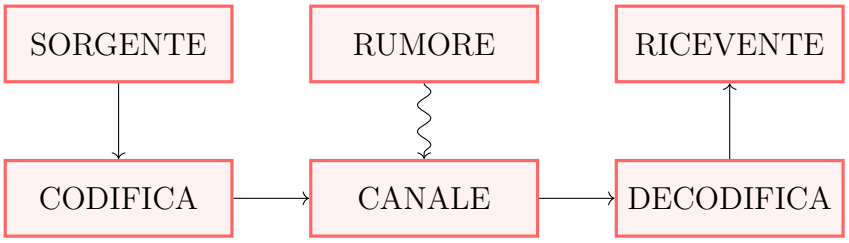
- Claude Shannon(USA): primo in assoluto. Dà una definizione in media
- Kolmogorov(URSS): arriva dopo Shannon ma dà una definizione puntuale. Espande il lavoro di Shannon.
- Chaitin e Solomonoff: arrivano allo stesso tempo di Kolmogorov ma non vengonono considerati perché Kolmogorov era più importante a livello accademico.

Durante questo corso ci proponiamo di riuscire a spedire dei dati da una sorgente a una destinazione attraverso un canale che può essere affetto da rumore.

Obiettivi del corso:

- Sfruttare al massimo il canale
- Gestire i bit persi nella trasmissione

1.2 Una visione d'insieme



Shannon modella l'ambiente come composto da 3 attori:

- **Sorgente**: La sorgente genera il messaggio, lo codifica e lo spedisce sul canale.
- **Canale**: Il canale è il tramite tra la sorgente e la destinazione. E' il "posto" in cui passa l'informazione. E' affetto da **rumore**.
- **Ricevente**: Riceve il messaggio codificato. E' suo compito riuscirlo a decodificare.

Vogliamo codificare messaggi sorgente

(A) Massimizzando informazioni trasmesse A OGNI utilizzo del canale (problema di **Source coding**)

(B) Minimizzando, simultaneamente al primo punto, il numero di errori di trasmissione dovuti al rumore (problema di **Channel coding**).

Shannon cerca di risolvere questo problema usando l'approccio divide et impera. Non è detto che questo approccio sia quello giusto. Infatti, non c'è garanzia che la soluzione ottimale dei due sottoproblemi sia tale che , se messe assieme, diano la soluzione ottimale per il problema. Questo perché potremmo non sfruttare possibili vantaggi di un problema sull'altro. Vale il seguente teorema:

TEOREMA di codifica sorgente e canale

L'unione delle soluzioni di source coding e channel coding (quindi dei due sottoproblemi risolti come indipendenti) dà la soluzione ottima.

Come si risolve il problema del source coding? Enunciamo, in maniera non formale per adesso, il primo teorema di Shannon.

TEOREMA I teorema di Shannon

Si può comprimere, tramite un codice, un messaggio con perdite di informazioni piccole

Questo è dovuto al fatto che l'informazione non è uniformemente distribuita. Ci sono parti della codifica inutile. Per esempio, se codifichiamoun cielo tutto azzurro, non abbiamo bisogno di dire che ogni pixel è di colore azzurro, ma ci basta dire che una porzione di una foto è tutta azzurra.
La codidfica, cioè la rimozione della ridondanza, va ad amplificare il problema del rumore. Ogni bit perso è significativo. Per risolvere il problema di channel coding useremo il secondo teorema di Shannon. Anche questo lo riportiamo, per ora, in modo informale.

TEOREMA II teorema di Shannon

Possiamo trasmettere con possibilità di errore piccola a piacere. Uitlizziamo una ridondanza, controllata in base alla distorsione del canale.

1.3 Modellazione

Cominciamo a dare una definizione formale dei vari strumenti che utilizzeremo durante il corso.
Innanzitutto vederemo il canale come una matrice stocastica, cioè una matrice tale che la somma dei contributi di una riga è pari 1.
Per esempio, data questa matrice che rappresenta un canale

IN/OUT	a	b	c	d	e
a	0.7	0	0.1	0.1	0.1
b	0	0.5	0.5	0	0
c	0.1	0.1	0.1	0.1	0.6
d	0.2	0.1	0.3	0.1	0.3
e	0.4	0.2	0.2	0.1	0.1

Sulla prima riga sono presenti tutti i simboli che la destinazione può ricevere (a,b,c,d ed e), mentre sulla prima colonna tutti i simboli che può generare la sorgente. Il numero che si trova nella posizione (i, j) indica la probabilità che la sorgente generi, e invii, il simbolo i-esimo e che il ricevente ricevi il simbolo j-esimo.
Nel nostro caso, la probabilità che inviando a si riceva c è di 0.1, cioè 10%
Si noti come la matrice identità modelli un canale "perfetto" ovvero senza distorsione (rumore).

Nella matrice appaiono i simboli "a,b,c,d,e". Questi sono i simboli prodotti dalla sorgente. I simboli prodotti dalla sorgente appartengono a $\mathbb{X}$.
I messaggi sono definiti come segue:

Sia $\mathbb{X}$ l'insieme finito di simboli che compongono i messaggi generati dalla sorgente.

Un messaggio $x = (x_1, ..., x_n)\,\epsilon\,\mathbb{X}^n$ di lunghezza n è una sequenza di n simboli sorgente.

I simboli sorgente sono poi tradotti (quindi codificati) in parole di codice prima di essere inviati sul canale.
Una **parola di codice** è una sequenza di numeri dall'insieme $\{0, \ldots, d-1\}$ dei simboli di codice, dove $d > 1$ è la base del codice.
Per effettuare la traduzione viene usata una **funzione di codifica**, che mappa i simboli sorgente in parole di codice

$$c : \mathbb{X} \to \{0, ..., d-1\}^+$$

Dove $\{0, \ldots, d-1\}^+$ è formalmente

$$\bigcup_{n=1}^{+\infty} \{0, \ldots, d-1\}^n$$

L'obiettivo che ci poniamo è di **minimizzare** $l_c(x)$, ovvero la lunghezza della parola di codice per il simbolo $x \in \mathbb{X}$.

Risulta naturale cercare di assegnare a dei simboli che sono usati più spesso una parola di codice con lunghezza minore. Viceversa, a simboli usati raramente associamo lunghezze maggiori. Questo perché il nostro obiettivo è di minimizzare la lunghezza media pesata per la probabilità di utilizzo del simbolo, in poche parole il valore atteso.

Shannon definisce come $p(x)$ la probabilità di generazione di un simbolo. Inoltre assume, per semplicità, l'indipendenza di un simbolo dall'altro. Ovvero, la generazione di un simbolo $x \in \mathbb{X}$ non influenza la generazione successiva di un simbolo $y \in \mathbb{X}$

Definiamo la variabile casuale $X : \mathbb{X} \to \mathbb{R}$. Questa rappresenta l'estrazione di 1 simbolo dalla sorgente.
p diventa quindi la distribuzione di probabilità dei simboli della sorgente, mentre definiamo P_n come $P_n(x_1, \ldots, x_n) = p(x_1) \ldots p(x_n)$. Questo vale perché l'estrazioni sono indipendenti.
P_n è la distribuzione dei messaggi $\mathbb{X}^n$.

Per avere una notazione più compatta, definiamo $\mathbb{D}$ come l'insieme $\{0, \ldots, d-1\}$ dei simboli di codice con base d. Quindi c può essere definita come

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

1.4 Problema codifica sorgente

Visti gli strumenti precedenti, possiamo definire in modo formale il problema della codifica sorgente che a inizio lezione avevamo descritto in modo non rigoroso.

PROBLEMA

Dato un modello di sorgente $< \mathbb{X}, p >$ e una base d > 1, trovare un codice $c : \mathbb{X} \rightarrow \mathbb{D}^+$ tale che il valore atteso

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

della lunghezza di parola di codice sia minimo.

Il problema, così formulato, si presta a una soluzione banale e inutile! Infatti, basta dire che $c(x) = 0$ per gni $x \in \mathbb{X}$. Bisogna introdurre delle limitazioni.

1.5 Limitazioni

La prima limitazione che introduciamo è che **il codice deve essere non singolare**. Un codice $c : \mathbb{X} \rightarrow \mathbb{D}^+$ è non signolare se a simboli della sorgente corrispondono parole di codice distinte (funzione iniettiva).

Formalmente,

$$\forall x, x^{'} \in \mathbb{X} : x \neq x^{'} \text{ vale } c(x) \neq c(x^{'})$$

Capitolo 2

Lezione II

2.1 Limitazioni

La limitazione imposta la scorsa lezione, ovvero che il codice sia non singolare, non è sufficiente. Infatti, senza nessun'altra limitazione non possiamo decodificare una parola in modo univoco.

Esempio 1. Data una sorgente che produce due simboli: A e B. Sia A codificato in 0, e B in 00. Il codice è non singolare, però non siamo sempre in grado di tradurre i messaggi. Infatti, se viene ricevuto 00 non sappiamo se corrisponde alla stringa AA o alla stringa B.

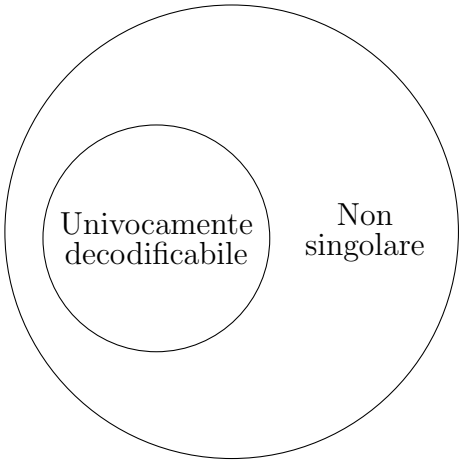
Introduciamo un'altra limitazione, questa volta sulla **estensione di un codice**. L'estensione del codice $c : \mathbb{X} \to \mathbb{D}^+$ è definita come

$$\mathbb{C} : \mathbb{X}^+ \to \mathbb{D}^+$$

Imponiamo che il codice sia **univocamente decodificabile**. Si dice che il codice è univocamente decodifcabile se e solo se la sua estensione è non singolare. Questa proprietà implica che

$$\forall\, y \in \mathbb{D}^+ \text{ c'è al più un unico messaggio } x \in \mathbb{X}^+ : \mathbb{C}(x) = y$$

Per determinare se un codice è univocamente decodificabile si usa l'algoritmo di **Sardinas-Patterson**. L'algoritmo ha complessita $O(mL)$, dove m è il numero delle parole di codice (i.e. $|\mathbb{X}|$) e L è la somma delle loro lunghezze (i.e. $\sum_{x\in\mathbb{X}} l(x)$).



In questo modo riusciamo a distinguere i simboli all'interno di una stringa in fase di decodifica. Il problema è che riusciamo solamente se abbiamo tutta la stringa. Questo non è sempre possibile (si pensi a situazioni in cui le informazioni arrivino in streaming senza terminare).
Dobbiamo aggiungere una limitazione che impedisca a una parola di codice di essere prefissa di un'altra.
Introduciamo il concetto di **codice istantaneo**. Un codice si dice istantaneo se nessuna parola è prefissa di un'altra.

2.2 Codici istantanei

FATTO

Se c è istantaneo allora è anche univocamente decodificabile

Dimostrazione 1. Sia $c : \mathbb{X} \to \mathbb{D}^+$ e $\mathbb{C}$ la sua estensione.

Possiamo escludere il caso in cui c sia non singolare. Infatti, in questo caso, avrei 2 simboli che codificano nella stessa parola di codice. Due parole uguali sono l'una il prefisso dell'altra.

A questo punto resta da dimostrare che se c non è univocamente decodificabile allora non è istantaneo.

Assumiamo che c sia non univocamente decodificabile, quindi

$$\exists\, x\,,\, x' \in \mathbb{X} \text{ con } x \neq x' |\ \mathbb{C}(x) = \mathbb{C}(x') \tag{2.1}$$

x e x' possono differire in 2 modi soltanto:

- ❒ Un messaggio è prefisso dell'altro
- ❒ C'è almeno uuna posizione in cui i 2 messaggi differiscono

Se $\mathbb{C}(x) = \mathbb{C}(x')$ e x è prefisso di x' (o viceversa) allora i restanti simboli di x dovrebbero per forza essere mappati in parola vuota.
Ma questo non è possibile, per costruzione infatti sappiamo che un simbolo non può essere mappato in una parola vuota. Ma anche se cambiassimo la nostra definizione, e ammettessimo la parola vuota, questo ci porterebbe a dire che il codice è non istantaneo, perché la parola vuota è prefisso di tutte le parole.

L'unico caso possibile è quindi il secondo.
Siccome $x \neq x'$, c'è una posizione i tale che $x_i \neq x'_i$.

Allora fino alla posizione i, ovvero per $j = 1, \ldots, i-1$ si ha che

$$\mathbb{C}(x_j) = \mathbb{C}(x^i_j)$$

Se però, $\mathbb{C}(x) = \mathbb{C}(x^i)$, allora $c(x_i)$ è prefisso di $c(x'_i)$ (o viceversa).

Questo perché, se fino alla posizione j sono uguale, e invece a i sono diverso, condizione necessaria affinché valga $\mathbb{C}(x) = \mathbb{C}(x^i)$ è che da j in poi, qualunque cosa ci sia dopo, io la codifico uguale. Questo non può succedere se $c(x_i)$ non è prefisso di $c(x^i_i)$ o viceversa.

Questo implica che il codice non è istantaneo.

Abbiamo dimostrato che

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

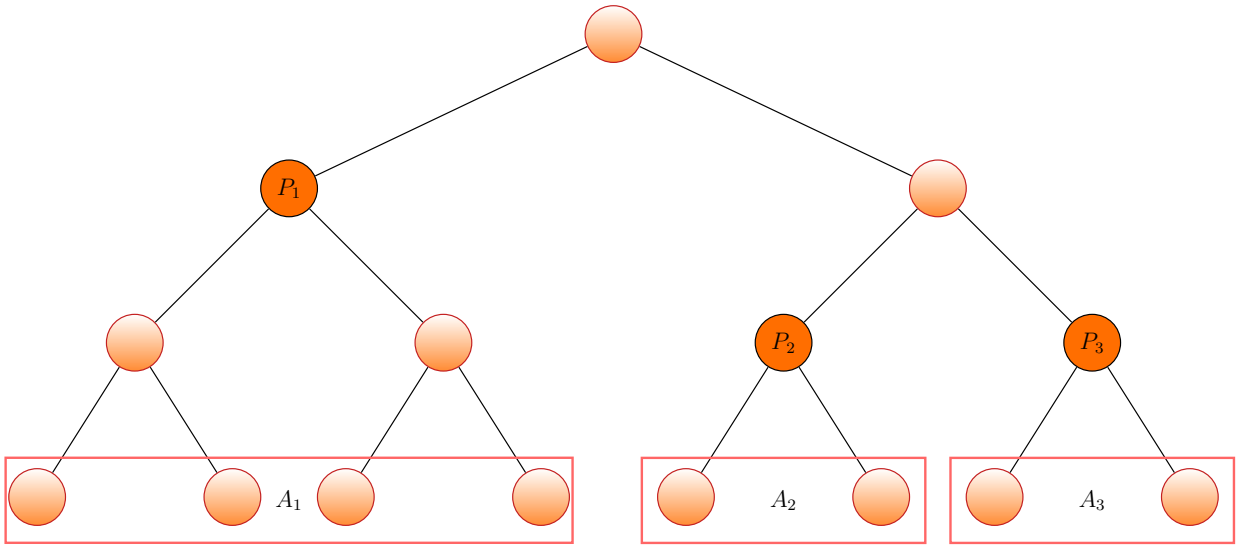


Figura 2.1: Albero cui nodi rappresentano le possibili parole

LEMMA Disuguaglianza di Kraft

Dati $\mathbb{X} = \{x_1, .., x_n\}, D > 1$ e m interi $l_1, .., l_m > 0$, esiste n codice istantaneo $c : \mathbb{X} \to \mathbb{D}^+$ tale che $l_c(x_i) = l_i$ per $i = 1, .., m \leftrightarrow$

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

Dimostriamo il lato $\to$.

Dimostrazione 2. Sia $l_{max} = \max_{i=1,..,m} l_c(x_i)$
Possiamo costruire un albero D-ario completo e di profondità l_{max}. Ogni nodo rappresenta una parola **possibile** di codice. Non è detto che sia utilizzata.

Si noti come ogni parola utilizzata (indicata con un nodo colorato in arancione) non ha come radice del sottoalbero un'altra parola valida. Cioè, se presa una parola P_i percorriamo l'albero verso l'alto non troveremo un nodo associato a un'altra parola P_j. Le foglie del sottoalbero di cui una parola P_i è radice le mettiamo nell'insieme A_i. Il numero di foglie in A_i è dato da $D^{l_{max}-l_i}$, dove l_{max} è la profondità massima dell'albero (quindi l'altezza) e l_i

è la profondità del nodo associato alla parola P_i. Sappiamo che il numero totale di foglie è $D^{l_{max}}$.

Possiamo quindi affermare quanto segue

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

Quindi

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

Dividendo per $D^{l_{max}}$ ambo i membri otteniamo

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

Capitolo 3

Lezione III

3.1 Codice di Shannon

Sia $<\mathbb{X}, p>$ il modello sorgente, dove

- $\mathbb{X} = \{x_1, \ldots, x_m\}$
- $p = \{p_1, \ldots, p_m\}$
- $L = \{l_1, \ldots, l_m\}$ con $d > 1$ e l_i è la lunghezza della parola i-esima.

Vogliamo minimizzare il valore atteso della lunghezza, volendo però ottenere un codice istantaneo. Allora devono valere le seguenti

$$\begin{cases} \min_{l_1,\ldots,l_m} \sum_{i=1}^{m} l_i p_i \\ \sum_{i=1}^{m} D^{-l_i} \leq 1 \end{cases}$$

Poiché p è una distribuzione di probabilità, vale che $\sum_{i=1}^{m} p_i = 1$. Allora

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

Possiamo imporre che

$$D^{-l_i} \leq p_i$$

Questa imposizione è totalmente arbitraria ma comunque rispetta la disequazione precedente, quindi è ammissibile. Risolviamo per l_i

$$-l_i \leq \log_D p_i$$
$$l_i \geq \log_D \frac{1}{p_i} \rightarrow l_i = \left\lceil \log_D \frac{1}{p_i} \right\rceil$$

Definzione 1. Il codice istantaneo che rispetta la condizione $l_i = \left\lceil \log_D \frac{1}{p_i} \right\rceil$ è noto come **codice di Shannon**.

3.2 Entropia

Riprendiamo il valore atteso

$$\sum_{i=1}^{m} l_i p_i$$

e sostituiamo a l_i il valore trovato prima

$$\sum_{i=1}^{m} p_i \left\lceil \log_D \frac{1}{p_i} \right\rceil$$

Se scegliamo p_i che siano potenze di D, abbiamo che

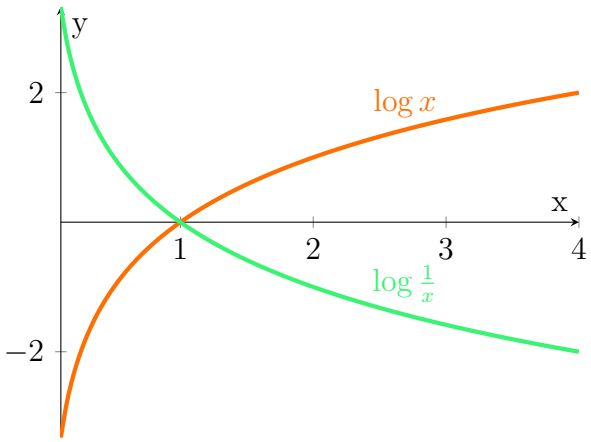
$$\log_D \frac{1}{p_I} = \left\lceil \log_D \frac{1}{p_i} \right\rceil$$

quindi il valore atteso diventa

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

L'equazione trovata corrisponde all'entropia.

Definzione 2. L'entropia è il limite inferiore alla correttezza del codice.



Dal grafico si evince come più x, cio'è p_i, è piccola, più $\log \frac{1}{x}$ è grande, e viceversa. Questo vuol dire che a simboli che hanno più probabilità di essere generati associamo una codifica più corta. Invece, simboli che hanno una probabilità minore di essere generati hanno una codifica più lunga a loro associata.

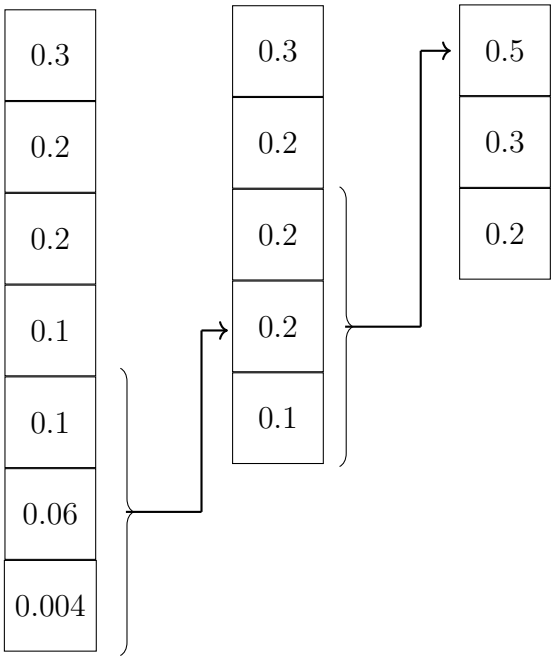
3.3 Metodo di Huffman

Usiamo il metodo di Huffman per trovare il codice istantaneo ottimo. Segue un esempio esplicativo.

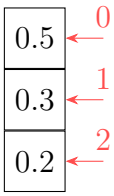
Esempio 2. Siano dati 7 simboli con $D = 1$ con

- $S = \{s_1, \ldots, s_7\}$
- $p = \{0.3, 0.2, 0.2, 0.1, 0.1, 0.06, 0.004\}$

Ordiniamo le probabilità e sommiamo insieme le ultime D, finché non rimaniamo con D probabilità.



Prendiamo le D probabilità risultanti. Quindi assegniamo un simbolo per ogni probabilità



"Srotoliamo" le probabilità e manteniamo il simbolo assegnato a una probabilità "somma" a tutte le probabilità "addendo". Poi, se ci sono simboli duplicati, aggiungiamo un simbolo, quindi:

0.3	1
0.2	2
0.2	01
0.1	03
0.1	010
0.06	011
0.004	012

0.3	1
0.2	2
0.2	01
0.2	02
0.1	03

0.5	0
0.3	1
0.2	2

Direzione di assegnazione

Capitolo 4

Lezione IV

4.1 Proprietà Entropia

Fissamo il modello $<\mathbb{X}, p>$, con $\mathbb{X}= \{x_1, \ldots, x_m\}$ e $p = \{p_1, \ldots, p_m\}$. Introduciamo la funzione iniettiva (ovvero la variabile aleatoria)

$$X : \mathbb{X} \to \{a_1, \ldots, a_m\}$$

Dove $a_1, \ldots, a_m$ sono tali che $P(X = a_i) = p_i$. Definiamo l'entropia della variabile aleatoria X, su m simboli, come

$$H_2(X) = \sum_{i=1}^{m} p_i \log_2 \frac{1}{p_i}$$

4.1.1 Cambio di base del logaritmo

Discutiamo per prima la proprietà di cambio di base. Se cambiamo la base del logaritmo otteniamo un'entropia che varia solo per un fattore moltiplicativo. Infatti,

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

Quindi cambiare la base del logaritmo corrisponde a scalare l'entropia per una costante positiva. Data un'entropia $H_b(X)$ possiamo affermare

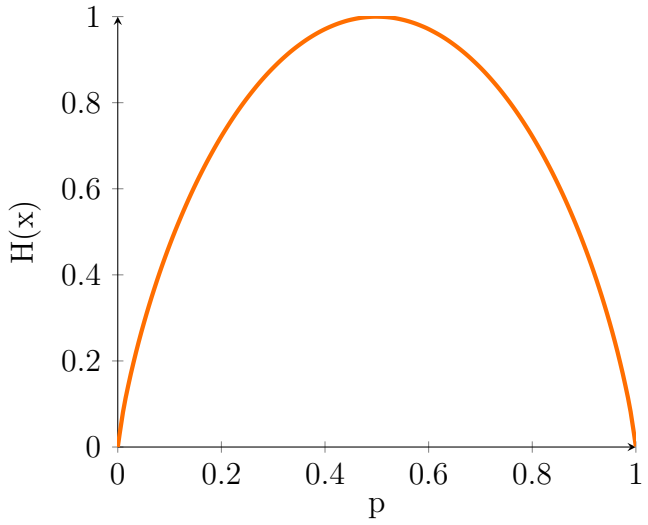
$$H_b(X) = \sum_{i=1}^{m} p_i \log_b \frac{1}{p_i} = \log_b a \cdot H_a(X)$$

4.1.2 Entropia binaria

Per entropia binaria intendiamo l'entropia della variabile X su 2 simboli. Quindi

$$H_2(X) = p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p}$$

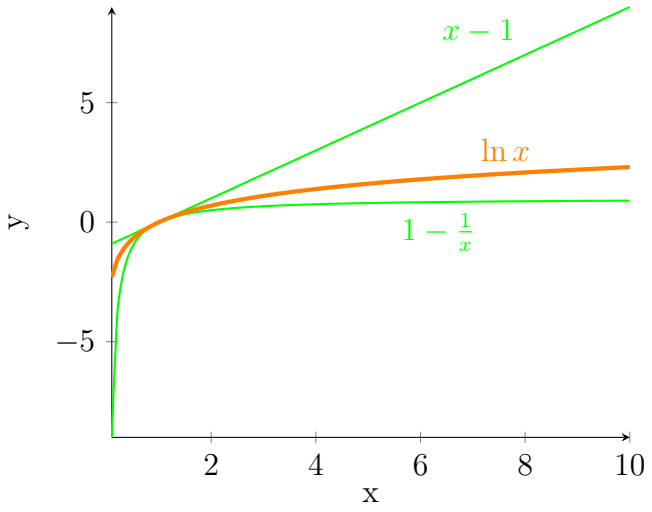
L'entropia binaria ha l'andamento qui sotto mostrato



- Se $p = 0$ l'entropia non esiste, per convenzione diciamo che $H(X) = 0$.
- Se $p = 1$ allora l'entropia vale 0.
- Se $p = \frac{1}{2}$ allora l'entropia vale 1.

4.1.3 Disuguaglianze elementari

Vale che $1 - \frac{1}{x} \leq \ln x \leq x - 1 \, \forall x > 0$.



4.1.4 Upper bound entropia

Abbiamo terminato di enunciare le tre proprietà legate all'entropia che ci serviranno nella dimostrazione di alcuni teoremi. Vediamo il primo teorema

TEOREMA Upper bound entropia

Sia X una variabile casuale che assume m valori distinti $a_1, \ldots, a_m$. Allora vale che $H_D(x) \leq \log_D m \, \forall D > 1$. Inoltre, $H(X) = \log_D m$ se e solo se X ha una distribuzione uniforme su $a_1, \ldots, a_m$.

Dimostrazione 3. Dimostrare che $H_D(X) \leq \log_D m$ equivale a dimostrare che $H_D(X) - \log_D m \leq 0$

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

$$= \sum_{i=1}^{m} p_i \log_D \frac{1}{p_I} - \log_D m \cdot \sum_{i=1}^{m} p_i$$

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

Applichiamo il cambiamento di base

$$= \sum_{i=1}^{m} p_i (\ln \frac{1}{mp_i} \frac{1}{\ln D})$$

Utilizziamo la seconda proprietà che abbiamo enunciato prima

$$\leq \sum_{i=1}^{m} p_i (\frac{1}{mp_i} - 1)(\frac{1}{\ln D}) = \frac{1}{\ln D}(\sum_{i=1}^{m} p_i \frac{1}{p_i m} - \sum_{i=1}^{m} p_i)$$

$$= \frac{1}{\ln D}(\sum_{i=1}^{m} \frac{1}{m} - \sum_{i=1}^{m} p_i) = 0$$

Dimostriamo ora la seconda parte del teorema. Assumiamo quindi che $P(X = a_i) = \frac{1}{m} \, \forall i \in 1 \dots m$

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

4.2 Entropia relativa

Introduciamo il concetto di entropia relativa

$$D(X||Y) = \sum_{s \in S} P_X(s) \log_D \frac{P_X(s)}{P_Y(s)}$$

Notiamo alcune cose

- E' chiamata D perché assomiglia a una distanza, anche se è asimmetrica (ovvero, $D(X||Y) \neq D(Y||X)$). Misura la diversità tra due distribuzioni, X e Y.
- S è un dominio generico. Deve essere uguale sia per X che per Y.

TEOREMA Non negatività entropia relativa

Per ogni coppia di variabili casuali X e Y definite su uun dominio comune S, vale la disuguaglianza $D(X||Y) \geq 0$.

Dimostrazione 4.

$$D(X||Y) = \sum_{s\in S} P_X(s) \log_D \frac{P_X(s)}{P_Y(s)} = \sum_{s\in S} P_X(s) \ln \frac{P_X(s)}{P_Y(s)} \frac{1}{\ln D}$$

$$= \frac{1}{\ln D} \sum_{s\in S} P_X(s) \ln \frac{P_X(s)}{P_Y(s)}$$

Per la terza proprietà possiamo scrivere

$$\geq \frac{1}{\ln D} \sum_{s\in S} P_X(s) \frac{P_Y(s)}{P_X(s)} = \frac{1}{\ln D} (\sum_{s\in S} P_X(s) - \sum_{s\in S} P_Y(s)) = 0$$

4.3 Legame tra valore atteso ed entropia

Enunciamo ora un teorema che lega il valore atteso con il concetto di entropia. Questo teorema serve per dare un lower bound al valore atteso. Vedremo quindi che il valore atteso almeno vale quanto l'entropia. Ciò conferma quello che abbiamo detto la lezione scorsa, cioè che l'entropia è il limite inferiore alla correttezza del codice.

TEOREMA Lower bound valore atteso

Se $c : \mathbb{X} \to \mathbb{D}^+$ è un codice istantaneo d-ario per una sorgente $<\mathbb{X}, p>$, allora vale che

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

Dimostrazione 5. Sia $Z : \mathbb{X} \to \mathbb{R}$ una variabile aleatoria casuale con distribuzione

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

Allora

$$\mathbb{E}[l_c] - H_D(X) = \sum_{x\in\mathbb{X}} p_x l_c(x) - \sum_{x\in\mathbb{X}} p(x) \log_D \frac{1}{p_x} =$$

$$= \sum_{x \in \mathbb{X}} p(x)(l_c(x) - \log_D \frac{1}{p(x)})$$

Usando il fatto che $1 \cdot l_c(x) = \log_D D \cdot l_c(x) = \log_D D^{l_c(x)}$ possiamo scrivere

$$= \sum_{x \in \mathbb{X}} p(x)(\log_D D^{l_c(x)} + \log_D p(x)) = \sum_{x \in \mathbb{X}} p(x) \log_D (\frac{p(x)}{D^{-l_c(x)}} \cdot 1)$$

Sapendo che $\sum_{x \in \mathbb{X}} \frac{D^{-l_c(x)}}{\sum_{x' \in \mathbb{X}} D^{-l_c(x')}} = 1$ scriviamo

$$\sum_{x \in \mathbb{X}} p(x) \log_D (\frac{p_x}{D^{-l_c(x)}} \frac{\sum_{x' \in \mathbb{X}} D^{-l_c(x')}}{\sum_{x'' \in \mathbb{X}} D^{-l_c(x'')}})$$

$$= \sum_{x \in \mathbb{X}} p(x)(\log_D (p_x \frac{\sum_{x' \in \mathbb{X}} D^{-l_c(x')}}{D^{-l_c(x)}})) - \log_D (\sum_{x \in \mathbb{X}} D^{-l_c(x)})$$

Sappiamo che $\frac{\sum_{x' \in \mathbb{X}} D^{-l_c(x')}}{D^{-l_c(x)}} = \frac{1}{q(x)}$

$$= \sum_{x \in \mathbb{X}} p(x)(\log_D \frac{p(x)}{q(x)}) - \sum_{x \in \mathbb{X}} p(x) \log_D \sum_{x' \in \mathbb{X}} D^{-l_c(x')}$$

Considerazioni finali:

- $\sum_{x \in \mathbb{X}} p(x)(\log_D \frac{p(x)}{q(x)})$ è l'entropia relativa, che sappiamo essere non negativa.
- Prendiamo in considerazione la parte rimanente dell'equazione. Notiamo che $\sum_{x' \in \mathbb{X}} D^{-l_c(x')}$ è < 1 per la disuguaglianza di Kraft. Il logaritmo di un numero < 0 è negativo. La sommatoria di numeri negativi è negativa. Siccome davanti c'è un meno diventa tutto positivo.
- L'espressione diventa così una somma. La somma di due quantità non negative è essa stessa non negativa. Quindi abbiamo dimostrato il teorema.

4.4 Sardinas-Patterson

L'algoritmo di Sardinas-Patterson serve a capire se un codice è univocamente decodificabile o meno. Dato un insieme di parole di codice vogliamo quindi capire se formano un codice univocamente decodificabile. L'algoritmo procede come segue:

- Prendiamo l'insieme di parole dato e lo chiamiamo S_1
- Costruiamo l'insieme S_2 in questo modo:

$$x \in S_1 : xy \in S_1 \rightarrow y \in S_2$$

  Ovvero se $x \in S_1$ è testa di una parola, metto la coda di questa parola nell'insieme S_2

- Per costruire l'insieme S_{i+1} procedo in questo modo:

$$x \in S_1 : xy \in S_i \rightarrow y \in S_{i+1}$$

$$z \in S_i : zy \in S_1 \rightarrow y \in S_{i+1}$$

- Ci fermiamo quando o troviamo un insieme vuoto (e quindi il codice è univocamente decodificabile) oppure quando nell'insieme S_i troviamo una (almeno una) parola del codice (cioè in S_1). In questo caso non è univocamente decodificabile.

Provare a fare i seguenti esercizi:

Esercizio 1. Il codice $\{A, BCA, DE, CDC, AABC, C\}$ è univocamente decodificabile?

Esercizio 2. Il codice $\{A, E, C, ABB, CED, BBEC\}$ è univocamente decodificabile?

Capitolo 5

Lezione V

5.1 Upper bound valore atteso

Sappiamo che $H_D(X) \leq \mathbb{E}[l_c]$, ma non sappiamo quanto siano vicini. La lunghezza media potrebbe essere molto distante, ciò renderebbe il codice poco efficiente. Sia $<\mathbb{X}, p>$ il modello sorgente, con $\mathbb{X} = \{x_1, \ldots, x_m\}$ e $p = \{p_1, \ldots, p_m\}$ vale il seguente teorema

TEOREMA Upper bound valore atteso

Dato il codice istantaneo c di Shannon, con lunghezze $l_i = l_c(x_i)$ tale che $l_i = \left\lceil \log_D \frac{1}{p_i} \right\rceil \ \forall i \in 1, \ldots, m$, allora

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

Dimostrazione 6.

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

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

Questo teorema ci dice che al massimo spreco un bit per simbolo rispetto all'ottimo. Questo sembra un buon risultato, però se il messaggio è lungo (cioè ha tanti simboli) perderemo tanti bit.

Definzione 3. L'inefficienza cresce linearmente con la lunghezza del messaggio.

Dati $c : \mathbb{X} \to \mathbb{D}^+$ e $C : \mathbb{X}^+ \to \mathbb{D}^+$ definiamo l'estrazione a blocchi di n. Vale il seguente fatto

$l_c(x_1, \ldots, x_n) \geq l_{c_n}(x_1, \ldots, x_n)$ Cioè la lunghezza di un messaggio dove i simboli sono trattati singolarmente è non minore della lunghezza del messaggio in cui sono trattati come blocchi. Infatti

$$l_c(x_1, \ldots, x_n) = \sum_{i=1}^{n} \left\lceil \log_D \frac{1}{p_i} \right\rceil \geq \left\lceil \sum_{i=1}^{n} \log_D \frac{1}{p_i} \right\rceil$$

$$= \left\lceil \log_D \frac{1}{\prod_i p_i} \right\rceil = \left\lceil \log_D \frac{1}{p(x_1, \ldots, x_n)} \right\rceil = l_{c_n}(x_1, \ldots, x_n)$$

Dove

$$C_n : \mathbb{X}^n \to \mathbb{D}^+$$

Quindi sostituiamo la giustapposizione di n simboli con l'estrazione di un messaggio da n simboli. In questo modo paghiamo 1 bit per ogni n simboli. Abbiamo quindi un nuovo modello sorgente, , che è più complesso!
A questo punto ci chiediamo se esiste una relazione tra $H(x_1, \ldots, x_n)$ e $H(x)$. Cominciamo con scrivere la definizione di $H(x_1, \ldots, x_n)$, ovvero

$$H(x_1, \ldots, x_n) = \sum_{x_1, \ldots, x_n} p_n(x_1, \ldots, x_n) \log_D \frac{1}{p_n(x_1, \ldots, x_n)}$$

Sappiamo che $p_n(x_1, \ldots, x_n) = \prod_i P(x_i)$ e anche che

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

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

Quindi riprendiamo l'equazione di prima e applichiamo la definizione soprastante

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

Per capire come procedere analizziamo il caso $n = 2$.

$$\sum_{x_1}\sum_{x_2}\prod_{i=1}^{2} p(x_i)(\log_2 \frac{1}{p_1} + \log_2 \frac{1}{p_2})$$

$$= \sum_{x_1}\sum_{x_2} \log_2 \frac{1}{p_1} p_1 p_2 + \log_2 \frac{1}{p_2} p_2 p_1$$

$$= \sum_{x_1}\sum_{x_2} p(x_1)p(x_2) \log_2 \frac{1}{p(x_1)} + \sum_{x_1}\sum_{x_2} p(x_1)p(x_2) \log_2 \frac{1}{p(x_2)} =$$

$$= \sum_{x_1} p(x_1) \log_2 \frac{1}{p(x_1} \sum_{x_2} p(x_2) + \sum_{x_2} p(x_2) \log_2 \frac{1}{p(x_2} \sum_{x_1} p(x_1)$$

$$= H(x_1) + H(x_2)$$

In generale possiamo quindi affermare che

$$H(x_1, \ldots, x_n) = nH(x)$$

5.2 Primo teorema di Shannon

Siamo pronti per enunciare il primo teorema di Shannon che avevamo accennato nella prima lezione.

TEOREMA Primo teorema di Shannon

Sia $\mathbb{C}_n : \mathbb{X}^N \to \mathbb{D}^+$ un codice di Shannon d-ario a blocchi per la sorgente $<\mathbb{X}, p>$, ossia $l_{\mathbb{C}_n}(xn) = \lceil Dp(xn) \rceil$ allora

$$\lim_{n\to+\infty} \frac{1}{n}\mathbb{E}[l_c] = H_D(X)$$

Dimostrazione 7. Sappiamo che

$$H_D(xn) = nH(X) \leq \mathbb{E}[l_c] < H_D(xn) = nH(X) + 1$$

Dividendo tutto per n otteniamo

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

Se $n \to +\infty$ allora $H(X) = H(X) + \frac{1}{n}$ e quindi $\mathbb{E}[l_c] = H(X)$

Il teorema precedente ci indica che il valore atteso si "schiaccia" sull'entropia col crescere della dimensione dei blocchi. Quindi se facciamo crescere la dimensione del blocco paghiamo poco in termini di bit!

5.3 Approssimare il modello

Purtroppo non conosciamo a propri il modello $<\mathbb{X}, p>$ e quindi ne dobbiamo fare una stima

$$<\mathbb{Y}, q>$$

L'entropia relativa $D(X||Y)$ ci dice l'errore pagato quando si usa la stima $\mathbb{Y}$ per $\mathbb{X}$. Vale il seguente teorema:

TEOREMA Valore atteso ed entropia relativa

Dato il modello sorgente $<\mathbb{X}, p>$, se $c : \mathbb{X} \to \mathbb{D}^+$ è codice di Shannon con $l_c(x) = \lceil Dq(x) \rceil$, dove q è una distribuzione su x, allora

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

Dimostrazione 8.

$$\mathbb{E}[l_c] = \sum_{x \in \mathbb{X}} p(x) \lceil Dq(x) \rceil < \sum_{x \in \mathbb{X}} p(x)(Dq(x) + 1)$$

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

$$= \sum_{x \in \mathbb{X}} p(x) \log_D \frac{p(x)}{q(x)} + \sum_{x \in \mathbb{X}} p(x) Dp(x) + 1$$
$$= D(X||Y) + H_D(X) + 1$$

Capitolo 6

Lezione VI

6.1 Algoritmo di Huffman

In questa lezione presentiamo un algoritmo per la costruzione di un codice di Huffman, con $D > 1$.

1. Ordina i simboli sorgente in base alla probabilità.
2. Crea modello sorgente fittizia in cui i D simboli meno probabili vengono raggruppati e sostituiti con un nuovo simbolo. La sua probabilità è pari alla somma delle probabilità dei simboli che sostituisce.
3. Se la sorgente contene più di D simboli ripeti il procedimento.

Dato il modello $<\mathbb{X}, p>$ con $|\mathbb{X}| = m$, l'algoritmo termina quando

$$|\mathbb{X}| = (D-1)k + 1$$

dove k indica l'iterazione. Ovvero, rimuoviamo k volte $d-1$ simboli, se ne resta 1 solo abbiamo terminato. Se non esiste questo k allora aggiungiamo dei simboli "dummy" con probabilità pari a 0.

6.2 Codici di Huffman

Presentiamo adesso un teorema che lega ci permette di capire se i codici di Huffman siano buoni o meno.

TEOREMA Relazione tra codice di Huffman e qualsiasi altro codice

Data una sorgente $<\mathbb{X}, p>$ e dato $D > 1$, il codice D-ario c di Huffman minimizza $\mathbb{E}[l_c]$ tra tutti i codici istantanei per la medesima sorgente. Ovvero,

$$\mathbb{E}[l_c] \leq \mathbb{E}[l_{c',c_2,\tilde{c}}]$$

Per dimostrare il teorema ci serve prima enunciare un fatto.

FATTO

Sia c' un codice D-ario di Huffman per la sorgente $\mathbb{X}' = \{xm - d + 1\}$ con probabilità $p_1 \geq \cdots \geq p_{m-d+1}$. Sia $\mathbb{X}$ la sorgente ottenuta togliendo da $\mathbb{X}'$ il simbolo x_k e aggiungendo d nuovi simboli, $x_{m-d+2} \dots x_{m+1}$, con probabilità $p_{m-d+2}, \dots, p_{m+1}$, tali che $p(.) > 0$ e anche che $p(.) < p_{m-d+1}$. Inoltre deve valere che $p_{m-d+2} + \cdots + p_{m+1} = p_k$. Allora vale che

$$c(x) = \begin{cases} c'(x) & \text{se } x \neq x_k \\ c'(x_k) \cdot i & \text{se } x = x_{m-d+2} \text{ a } x_{m+1} \forall i \in 0, \dots, d-1 \end{cases}$$

è un codice di Huffman per la sorgente.

Il fatto esposto è l'algoritmo di Huffman proposto "al contrario". Dimostriamo ora il teorema.

Dimostrazione 9. Dimostrazione per induzione.

$m = 2$, caso base.

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

Ipotesi induttiva: Huffman ottimo per $k \leq m - 1$

Fissiamo $<\mathbb{X}, p>$ (con m simboli) con $\mathbb{X} = \{\ldots, u, \ldots, v\}$ dove $p(u)$ e $p(v)$ sono minime. Definiamo $< \mathbb{X}', p' >$ con $u, v \in \mathbb{X}$ rimpiazzati da $z \in \mathbb{X}'$. Inoltre p' è tale che

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

Sia c' il codice di Huffman per $<\mathbb{X}', p'>$. Dato che $|\mathbb{X}'| = m-1$, c' è ottimale per ipotesi induttiva.
Il codice c per $\mathbb{X}$è definito come

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

Dimostriamo che c sia ottimo. Innanzitutto vale quanto segue

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

Esprimiamo il valore atteso in termini di $\mathbb{X}'$, quindi

$$= \sum_{x \in \mathbb{X}'} l_{c'}(x) p'(x) - l_{c'}(z) p'(z) + l_c(u) p(u) + l_c(v) p(v)$$

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

Raggruppiamo per $l_{c'}(z) + 1$

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

Sapendo che $p(u) + p(v) = p'(z)$

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

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

Per dimostrare l'ottimalità di c consideriamo un altro codice c_2 per $<\mathbb{X}, p>$ e verifichiamo che $\mathbb{E}[l_c] \leq \mathbb{E}[l_{c_2}]$
Sia c_2 istantaneo per $<\mathbb{X}, p>$. Siano $r, s \in \mathbb{X}$ tali che $l_{c_2}(r)$ e $l_{c_2}(s)$ sono massimi. Senza perdità di generalità possiamo assumere che r, s siano fratelli nell'albero di codifica c_2. Infatti,

- se non fossero fratelli e avessero un altro fratello (es. s ha fratello f) allora scegliamo s e f invece che s e r.
- Se non avessero fratelli possiamo sostituire le loro codifiche con quelle del padre fino a riportarci in una situazione in cui abbiano un fratello.

Definiamo ora il codice $\tilde{c}_2$.

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

Dove scambiamo la codifica di r con quella di u e quella di s con quella di v.

Esaminiamo la differenza tra c_2 e $\tilde{c}_2$

$$\mathbb{E}[l_{\tilde{c}_2}] - \mathbb{E}[l_{c_2}] =$$

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

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

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

Sapendo che

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

Possiamo affermare che

$$\mathbb{E}[l_{\tilde{c}_2}] - \mathbb{E}[l_{c_2}] \leq 0$$

Quindi

$$\mathbb{E}[l_{\tilde{c}_2}] \leq \mathbb{E}[l_{c_2}]$$

Capitolo 7

Lezione VII

7.1 Termine dimostrazione lezione VI

Introduciamo il codice c'_2, fatto come segue

$$\tilde{c}'_2 = \begin{cases} \tilde{c}_2(x) & \text{se } x \neq z \\ \omega & \text{se } x = z \end{cases}$$

Dove c'_2 è definito sulla sorgente $< \mathbb{X}', p' >$. Inoltre, dopo avere scambiato r e s con u e v, quest'ultimi sono fratelli. Quindi

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

Allora,

$$\mathbb{E}[l_{\tilde{c}_2}] = \sum_{x \in \mathbb{X}' : x \neq z} p'(x) l_{\tilde{c}_2}(x) + p(u)(l_{c'_2}(z) + 1) + p(v)(l_{c'_2}(z) + 1)$$

$$= \sum_{x \in \mathbb{X}' : x \neq z} p'(x) l_{\tilde{c}_2}(x) + p'(z) l_{c'_2}(z) + p'(z)$$

$$= \mathbb{E}[l_{c'_2}] + p'(z) \geq \mathbb{E}[l_{c'}] + p'(z)$$

Mettendo tutto insieme

$$\mathbb{E}[l_c] = \mathbb{E}[l_{c'}] + p'(z) \leq \mathbb{E}[l_{c'_2}] + p'(z) = \mathbb{E}[l_{\tilde{c}_2}] \leq \mathbb{E}[l_{c_2}]$$

$$\mathbb{E}[l_c] \leq \mathbb{E}[l_{c_2}]$$

7.2 Disuguaglianza di Kraft-McMillan

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

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

Teorema Disuguaglianza di Kraft-McMillan

lm sono le lunghezze di un codice D-ario univocamente decodificabile, per una sorgente di m simboli, se e solo se

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

Prima di dimostrare il teorema definiamo l'estensione k-esima, $\mathbb{C}_k$, di un codice c,

$$\mathbb{C}_k : \mathbb{X}^k \to \mathbb{D}^+$$

Dimostrazione 10. Per dimostrare che $\sum_{i=1}^{m} D^{-l_i} \leq 1$ implica che il codice sia univocamente decodificabile notiamo che, se vale $\sum_{i=1}^{m} D^{-l_i} \leq 1$, allora lm sono lunghezze di un codice istantaneo, quindi di un codice univocamente decodificabile.

Dimostriamo l'altro lato, ovvero che dato un codice univocamente decodificabile vale la disuguaglianza. $\forall k \geq 1$ possiamo scrivere

$$(\sum_{x \in \mathbb{X}} D^{-l_c(x)})^k = \sum_{x_1} \cdots \sum_{x_k} D^{-l_c(x_1)} \dots D^{-l_c(x_k)} = (1)$$

Questo è verifcabile provando il caso $k = 2$

$$(\sum_i a_i)^2 = (\sum_i a_i)(\sum_j a_j) = \sum_i \sum_j a_i a_J$$

$$(1) = \sum_{(xk)\in\mathbb{X}^k} D^{-(l_c(x_1)+\cdots+l_c(x_k))} = \sum_{(xk)\in\mathbb{X}^k} D^{-l_{\mathbb{C}_k}(xk)} = (2)$$

Dove

$$l_{\mathbb{C}_k}(xk) = l_c(x_1) + \cdots + l_c(x_k)$$

Introduciamo l'insieme $\mathbb{X}^k_n$, definito come segue

$$\{(xk) \in \mathbb{X}^k : l_{\mathbb{C}_k}(xk) = n\}$$

Quindi,

$$(2) = \sum_{(xk)\in\mathbb{X}^k} D^{-l_{\mathbb{C}_k}(xk)} = \sum_{n=1}^{k\cdot l_{max}} \sum_{(xk)\in\mathbb{X}^k_n} D^{-l_{\mathbb{C}_k}(xk)}$$

$$= \sum_{n=1}^{k\cdot l_{max}} |\mathbb{X}^k_n| D^{-n} = (3)$$

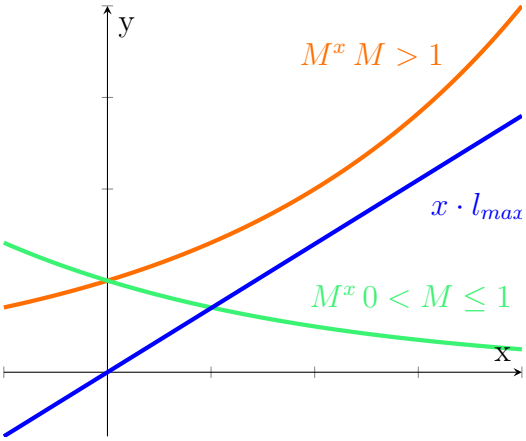
Siccome c è univocamente decodificabile, $\mathbb{C}$ è iniettiva. Quindi $|\mathbb{X}^k_n| \leq |D^n|$

$$(3) \leq \sum_{n=1}^{k\cdot l_{max}} D^n D^{-n} \leq k \cdot l_{max}$$

Allora,

$$(\sum D^{-l_c(x)})^k \leq k l_{max}$$

Poniamo ora $\sum D^{-l_c(x)} = M$ e ci chiediamo quanto valga M.



Siccome vogliamo che M^x sia sotto $x \cdot l_{max}$, allora M deve essere compreso tra 0 e 1, possiamo concludere che

$$(\sum D^{-l_c(x)})^k \leq 1$$

Capitolo 8

Lezione VIII

8.1 Esercizi di probabilità

Esercizio 3. Sapendo che la probabilità di un messaggio di essere corrotta è $\frac{1}{8}$, quanti bit mi servono per rappresentarla? Usiamo la formula

$$\log_2 \frac{1}{p_i}$$

Dato che $p_i = \frac{1}{8}$ allora ci serviranno

$$\log_2 8 = 3$$

bit.

Esercizio 4. Qual è la probabilità di ottenere 4 messaggi dove il primo è corretto e gli altri 3 no, sapendo che $p = \frac{1}{8}$ (p è la probabilità che il messaggio sia corretto)?

$$\frac{1}{8}(1 - \frac{1}{8})^3$$

Esercizio 5. Preso l'esercizio precedente, quanti bit ci servono?

$$\log_2 \frac{1}{8}(1 - \frac{1}{8})^3 = \log_2 \frac{1}{8} + \log_2 (1 - \frac{1}{8})^3 = -3 + \log_2 (1 - \frac{1}{8})^3$$

Esercizio 6. Abbiamo un dado a 6 facce lanciato 20 volte. Qual è la probabilità di...

- Fare 20 lanci e il 5 non esce mai?

$$(\frac{5}{6})^{20}$$

- Fare 20 lanci e il 5 esce una volta?

$$(\frac{5}{6})^{19} \cdot (\frac{1}{6}) \cdot 20$$

- Fare 20 lanci ed esce almeno 1 volta il 5? E' come chiedersi la probabilità opposta a quando non esce mai il 5 quindi:

$$1 - P(\text{non esce mai } 5) = 1 - (\frac{5}{6})^{20}$$

8.2 Numero di bit necessari

Quanti bit ci servono per comunicare il risultato di un certo evento? Se usiamo un codice istantaneo...

$$H(x) < \text{n}^\circ \text{ bit}$$

E se abbiamo 2 variabli, ovvero 2 risultati da comunicare?

$$H(X,Y) = \sum_{x\in\mathbb{X}}\sum_{y\in\mathbb{Y}} p(x,y)\log\frac{1}{p(x,y)}$$

$H(X,Y)$ è detta **entropia congiunta**. Quanti bit ci servono avendo un evento condizionante? Cioè, se il ricevente conosce $\mathbb{X}$, quanti bit ci servono per comunicare $\mathbb{Y}$?

$$\begin{aligned} H(Y|X) &= \sum_{x\in\mathbb{X}} p(x)H(\mathbb{Y}|\mathbb{X}=x) \\ &= \sum_{x\in\mathbb{X}} p(x)(\sum_{y\in\mathbb{Y}} p(y|x)2p(y|x)) \\ &= \sum_{x\in\mathbb{X}}\sum_{y\in\mathbb{Y}} p(x,y)2p(y|x) \end{aligned}$$

$H(Y|X)$ è detta **entropia condizionata**. Seguono due definizioni di probabilità

Definzione 4. $p(x, y)$ è detta **probabilità congiunta**, ed è la probabilità che avvenga sia x che y.

Definzione 5. $p(x) = \sum_{y \in \mathbb{Y}} p(x, y)$ è detta **probabilità marginale**.

Definzione 6. $p(y|x) = \frac{p(x,y)}{p(x)}$ è detta **probabilità condizionale**.

FATTO Chain Rule per l'entropia

Vale la seguente uguaglianza

$$H(x, y) = H(x) + H(y|x) = H(y) + H(x|y)$$

Inoltre vale anche la seguente per gli spazi condizionati

$$H(x, y|z) = H(x, z) + H(y|x, z)$$

8.3 Esercizi su entropia

Esercizio 7. Sia $x \in X$ una variabile rappresentante l'estrazione di un numero tra 0 e 9, e y definita come $y = x + 2 \mod 10$, quanto vale $H(Y|X)$? Vale 0! Infatti, se il ricevente ha gia X non dobbiamo inviare alcuna informazione. Il ricevente può calcolarsi Y da solo.

Esercizio 8. Sia $X = \{-1, 0, 1\}$ e $Y = X^2$, quanto vale $H(Y|X)$? Vale 0 per la stessa ragione di prima. E invece $H(X|Y)$? Sicuramente è $\neq 0$. Non possiamo ricavare X avendo solo Y.

Esercizio 9. Dato un sistema $S - C - R$ (sorgente-canale-ricevente). Sia M una matrice che rappresenta il canale

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

e $\mathbb{X} = \{a_1, \ldots, a_4\}$ e $p = [0.2, 0.2, 0.2, 0.4]$. Calcoliamo $H(R|S)$:

$$H(R|S) = \sum_{i=1}^{4} p(a_i) H(R|a_i)$$

$$= \sum_{i=1}^{4} p(s_i) \sum_{j=1}^{5} p(b_j|a_i) \cdot \log_2 \frac{1}{p(b_j|a_i)}$$

E' giusto? No! Non abbiamo conteggiato che:

1. La somma dei $p(a_i)$ sia $= 1$.
2. La somma delle righe della matrice è uguale a 1.

La prima riga della matrice data non fa 1!

Esercizio 10. Stesso esercizio di prima ma...

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

con $p = [0.2, 0.3, 0.1, 0.4]$. Calcolare $H(R|S)$. Stessa formula di prima, troviamo il risultato dei componenti!

$$H(R|a_1) = \sum_{j=1}^{5} p(b_j|a_1) \log_2 \frac{1}{p(b_j|a_1)} = 2.246$$

In modo analogo calcoliamo gli altri valori

$$H(R|a_2) = 1.96095$$

$$H(R|a_3) = 1.77$$

$$H(R|a_4) = 2.046$$

Quindi

$$H(R|S) = \sum_{i=1}^{4} p(a_i) H(R|a_i)$$

$$= (0.2 \cdot 2.246) + (0.3 \cdot 1.96) + (0.1 \cdot 1.77) + (0.4 \cdot 2.046) = 2.033$$

Esercizio 11. Esercizio lasciato al lettore (prof. potrebbe chiedere un'idea all'esame). Posso ottenere lo stesso risultato in un altro modo? (usando le formule viste prima). Due strade consigliate:

- ❒ Usare chain rule
- ❒ Usare l'uguaglianza

$$H(S,R) = H(S|R) + H(R) = H(R|S) + H(S)$$

per trovare e calcolare $H(R|S)$, quindi

$$H(R|S) = H(S|R) + H(R) - H(S)$$

Capitolo 9

Lezione IX

9.1 Informazione mutua

Introduciamo il concetto di **informazione mutua**. Per informazione mutua si intende un parametro (o misurazione) che fa riferimento a 2 variabili casuali; Ci dice quanta informazione viene rilasciata da una rispetto all'altra. L'infrmazione mutua è formalmente definita come

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

Fatto **Non negatività informazione mutua**

L'informazione mutua è non negativa, ovvero

$$I(X,Y) \geq 0$$

Dimostrazione 11. Applichiamo la definizione di probabilità congiunta

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

$$= \sum_{x\in\mathbb{X}}\sum_{y\in\mathbb{Y}} p(x,y)\log\frac{1}{p(x)} + \sum_{x\in\mathbb{X}}\sum_{y\in\mathbb{Y}} p(x,y)\log p(x|y)$$

$$H(X) - H(X|Y) \geq 0$$

Esercizio 12. Quanto vale l'informazione mutua tra X e Y se sono indipendenti?

$$H(X|Y) = H(X)$$

quindi

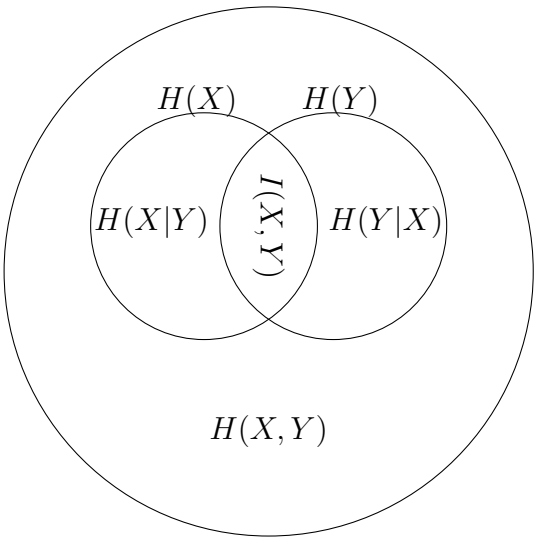
$$I(X,Y) = H(X) - H(X) = 0$$

Esercizio 13. Quanto vale l'informazione mutua tra X e Y se $X = g(Y)$? In questo caso $H(X|Y) = 0$, quindi

$$I(X,Y) = H(X)$$

L'informazione mutua la possiamo esprimere in modi diversi:

$$\begin{cases} H(Y) = H(Y|X) + I(X,Y) \\ H(X) = H(X|Y) + I(X,Y) \\ H(X,Y) = H(X) + H(Y) - I(X,Y) \\ H(X,Y) = H(X|Y) + H(Y|X) + I(X,Y) \end{cases}$$



Vale anche che

$$I(X,Y|Z) = \sum p(x,y|z) \log \frac{p(x,y|z)}{p(y|z)p(x|z)}$$

9.2 Data processing inequality

Introduciamo il seguente teorema, che non dimostriamo,

Teorema Data processing inequality

Siano X,Y e Z variabili casuali s ominio finito tali che $p(x,y,z)$ soddisfa $p(x,y|z) = p(x|y)p(z|y) \forall x,y,z$ (cioè x e z sono indipendenti dato y), allora l'informazione mutua tra x e y è $\geq$ dell'informazione mutua tra x e z ovvero

$$I(X,Y) \geq I(X,Z)$$

Corollario

Vale la seguente disequazione:

$$I(X,Y) \geq I(X,Y|Z)$$

Esempio 3. Consideriamo due variabili casuali X e Y bernoulliane indipendenti di parametro $\frac{1}{2}$ e definiamo $Z = X + Y$. Chiaramente X, Z non sono indipendenti dato Y. Osserviamo che $I(X,Y) = 0$ (perché sono indipendenti) mentre

$$I(X,Y|Z) = H(X|Z) - H(X|Y,Z)$$

Sappiamo che $H(X|Y,Z) = 0$ quindi

$$= p(Z=0)H(X|Z=0) + p(Z=1)H(X|Z=1) + p(Z=2)H(X|Z=2)$$

Dove $p(Z=0)H(X|Z=0) = 0$ e $p(Z=2)H(X|Z=2) = 0$ allora possiamo scrivere

$$= p(Z = 1)H(X|Z = 1)$$

L'entropia di X è massima se $Z = 1$ e allora $H(X|Z = 1) = 1$,

$$= p(Z = 1) = \frac{1}{2}$$

Abbiamo quindi costruito un esempio in cui $I(X, Y) \leq I(X, Y|Z)$ dove non è vero che X e Z sono indipendenti dato Y.

9.3 Disuguaglianza di Fano

Un altro teorema, che non dimostriamo, è il seguente:

Teorema Disuguaglianza di Fano

Siano x, y variabili casuali su domini X e Y finito. Sia $g : Y \to X$ la funzione di decodifica e p_e la probabilità di errore $p_e = p(g(y) \neq x)$, allora

$$p_e \geq \frac{H(x|y) - 1}{\log_2 |X|}$$

Con questo teorema riusciamo a legare il rumore con l'entropia relativa.

Capitolo 10

Lezione X

10.1 Canale

Dobbiamo codificare il messaggio sul canale. Definiamo il canale con la tripla

$$C < \mathbb{X}, \mathbb{Y}, p(y|x) >$$

dove

- $\mathbb{X}$ è l'insieme dei simboli di input.
- $\mathbb{Y}$ è l'insieme dei simboli di output.
- $p(y|x)$ è la probabilità di ottenere y dato x. Notare che y e x sono la realizzazione delle varibaili casuali Y e X e formalmente sarebbe più giusto scrivere $p(Y = y|X = x)$.

Non è detto che $\mathbb{X} = \mathbb{Y}$. Useremo solamente canali discreti e senza memoria, ovvero canali dove il bit ricevuto dipende solo dal bit appena inviato. Se un canale viene usato n volte, qual è la probabilità che inviato

$$x^n = \{x_1, \ldots, x_n\}$$

riceviamo

$$y^n = \{y_1, \ldots, y_n\}$$

? Possiamo scrivere questa probabilità come

$$p(y^n|x^n)$$

Poiché i simboli sono indipendenti tra loro allora

$$p(y_n|y^{n-1}, x^n)p(y_{n-1}|y^{n-2}x^n)\dots p(y_1|x^n)$$

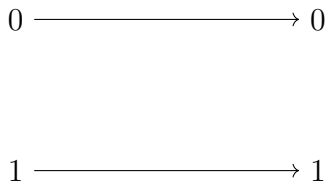
Ricordando che il canale che consideriamo è senza memoria

$$p(y_n|x_n)p(y_{n-1}|x_{n-1})\dots p(y_1|x_1) = \prod_{i=1}^{n} p(y_i|x_i)$$

Cioè consideriamo solo l'ultimo simbolo inviato.
Vediamo ora alcuni esempi di canali.

10.1.1 Canale binario senza rumore

Vediamo il canale binario senza rumore. Possiamo dare due possibili rappresentazioni equivalenti, la prima è quella grafica

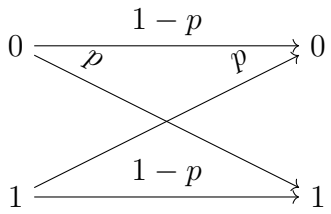


La seconda rappresentazione è quella matriciale

$$\begin{array}{c} X\backslash Y \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{array}$$

10.1.2 Canale binario simmetrico

Diamo prima la rappresentazione grafica



E poi la rappresentazione matriciale

$$\begin{array}{c} X\backslash Y \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix} \end{array}$$

10.2 Capacità del canale

Definzione 7. La **capacità** del canale indica quanta informazione possiamo mandare sul canale.

Formalmente è definita come

$$C = \max_{p(x)} I(x, y)$$

dove con $\max_{p(x)}$ prendiamo in considerazione tutte le possibili distribuzioni di $p(x)$, ovvero di probabilità di generare i simboli sorgenti. Calcoliamo le capacità per i canali precedenti:

- Canale binario senza rumore. Riscriviamo la capacità esprimendo l'informazione mutua

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

  Siccome dato Y non abbiamo incertezza su X, allora $H(X|Y) = 0$, quindi

  $$= \max_{p(x)} H(X)$$

  Scegliendo $p(0) = \frac{1}{2}$ e $p(1) = \frac{1}{2}$ massimizziamo l'entropia che vale 1 in questo caso. Allora la capacità del canale è proprio 1.

- Canale binario simmetrico. Cominciamo con l'osservare che

  $$I(X,Y) = H(Y) - H(Y|X) = H(Y) - H(Y|X=0)p(X=0) - H(Y|X=1)p(X=1)$$

  Notiamo che

$$H(Y|X=0) = -p(y=0|x=0)\log_2 p(y=0|x=0) - p(y=1|x=0)\log_2 p(y=1|x=0)$$

$$= -(1-p)\log_2(1-p) - p\log_2 p = H(p)$$

Con $H(p)$ l'entropia di una bernoulliana di parametro p. Analogamente possiamo dimostrare che $H(Y|X=1) = H(p)$. Quindi la capacità del canale si riduce a

$$C = \max_{p(x)} H(Y) - H(p)$$

Non ci resta altro da fare che trovare il massimo valore di $H(Y)$. Cominciamo con analizzare $P(Y=1)$

$$P(Y=1) = P(Y=1|X=0)P(X=0) + P(Y=1|X=1)P(X=1)$$

$$= pP(X=0) + (1-p)P(X=1)$$

Notiamo che quando $P(X=1) = \frac{1}{2}$ abbiamo che $P(Y=1) = \frac{1}{2}$. Per questa scelta di $p(x)$ abbiamo che $H(Y) = 1$, ed è il massimo valore che può assumere. Possiamo quindi concludere che

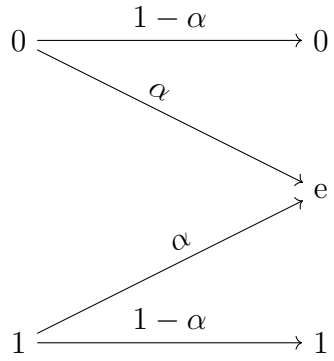
$$C = 1 - H(p)$$

Capitolo 11

Lezione XI

11.1 Canale binario a cancellazione

Introduciamo un altro canale, il canale binario a cancellazione (in inglese BEC, binary erased channel).

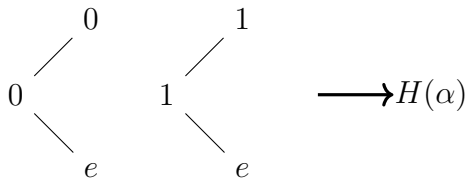


Dove "e" indica l'errore. Con probabilità $1-\alpha$ ricevimanto il simbolo giusto invece con probabilità α si verifica un errore. Calcoliamo la capacità di questo canale. Cominciamo con l'osservare che

$$H(Y|X=0) = H(Y|X=1) = H(\alpha)$$

e quindi

$$H(\alpha) = H(Y|X)$$



Ciò implica che

$$I(X,Y) = H(Y) - H(\alpha)$$

Dobbiamo quindi calcolarci il massimo $H(Y)$. Cominciamo con introdurre la variabile casuale bernoulliana Z:

$$Z = \begin{cases} 1 & \text{se } \mathbb{Y} = e \\ 0 & \text{altrimenti} \end{cases}$$

Quindi abbiamo 1 se c'è errore, 0 altrimenti. Notiamo che

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

siccome $H(Z|Y) = 0$

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

Vale anche che

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

Dunque

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

Osserviamo che

$$p(z=1) = p(z=1|X=0)p(X=0) + p(z=1|X=1)p(X=1)$$

$$= \alpha p(X=0) + \alpha p(X=1) = \alpha$$

Ne consegue che $p(z=0) = 1-\alpha$ e quindi $H(\alpha) = H(Z)$. Ci resta solamente da calcolare $H(Y|Z)$. Innanzitutto sappiamo che $p(Y=1|Z=0) = p(X=1)$ e quindi $H(Y|Z=0) = H(X)$. Possiamo allora scrivere

$$H(Y|Z) = H(Y|Z=0)p(z=0) + H(Y|Z=1)p(z=1)$$

e sapendo che $H(Y|Z=1)=0$ e che $p(z=0)=1-\alpha$

$$= H(X)(1-\alpha)$$

Concludendo

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

11.2 Codice di Fano

Presentiamo l'algoritmo per la costruzione di un codice di Fano

1. Prendiamo le probabilità e le ordiniamo in maniera decrescente.
2. Dividiamo la lista delle probabilità in due gruppi tali che

$$\sum_{1° gruppo} prob. \tilde{=} \sum_{2° gruppo} prob.$$

   ovvero due gruppi che hanno più o meno la stessa probabilità.
3. Assegniamo 0 agli eventi del 1° gruppo e 1 a quelli del 2° gruppo.
4. Applichiamo ricorsivamente i punti ①,② e ③ finché abbiamo ancora simboli da assegnare.

Esempio 4. Dati i simboli $\{A, B, C, D, E\}$ e le probabilità a loro associate (rispettivamente) $\{0.35, 0.25, 0.15, 0.15, 0.1\}$. Applichiamo l'algoritmo passo per passo:

1. Dividiamo in due gruppi con probabilità più o meno uguali. Scegliamo (ci possono essere più possibilità) di costruire $\{A, B\}$ e $\{C, D, E\}$.
2. Assegniamo 0 ad $\{A, B\}$ e 1 a $\{C, D, E\}$.

3. Prendiamo il gruppo $\{A, B\}$ e ripetiamo la procedura.
   (a) Dividiamo nell'unico modo possibile, quindi $\{A\}$ e $\{B\}$.
   (b) Assegniamo 0 ad A e 1 a B.
4. Prendiamo il gruppo $\{C, D, E\}$ e ripetiamo la procedura.
   (a) Dividiamo in due gruppi, $\{C\}$ e $\{D, E\}$.
   (b) Assegniamo 0 a C e 1 a D, E.
   (c) Prendiamo il gruppo $\{D, E\}$ e ripetiamo la procedura.
      i. Dividiamo nell'unico modo possibile, quindi $\{D\}$ e $\{E\}$.
      ii. Assegniamo 0 ad D e 1 a E.

Abbiamo quindi ottenuto il seguente codice

- $A \to 00$
- $B \to 01$
- $C \to 10$
- $D \to 110$
- $E \to 111$

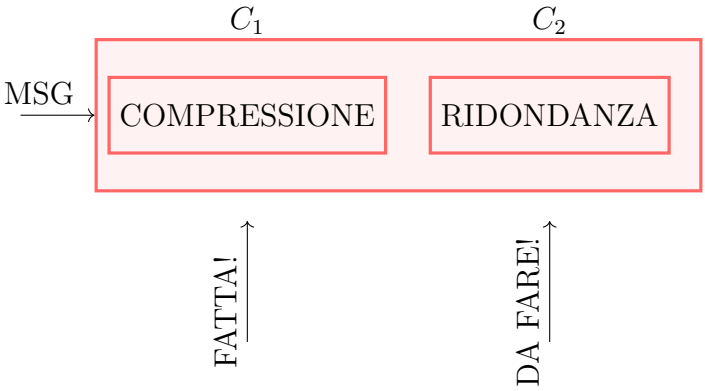
Notiamo dall'esempio che i codici di Shannon-Fano non sono ottimali. Il problema è che partiamo dalla radice e già assegniamo i prefissi, quindi al contrario di quello che fa Huffman.

Capitolo 12

Lezione XII

12.1 Introduzione

Il problema che c'eravamo posti all'inizio era suddiviso in due parti



La parte che ci resta da fare C_2 è divisa in 2 sottoparti:

1. rilevazione errore
2. correzione errore

Ci sono due tipi di errori con cui dobbiamo confrontarci

- Errori singoli

$$1010x01010x$$

- Errori burst

$$1xxx01xxxx01$$

Definzione 8. Si dice **rumore bianco** il rumore che ha effetto su tutti i bit allo stesso modo.

12.2 Esercizi di probabilità

Siano:

- $n \to$ numero di bit
- $p \to$ probabilità di errore
- $(1-p)^n \to$ probabilità di avere n bit giusti

Risolvere i seguenti esercizi:

Esercizio 14. Qual è la probabilità di avere esattamente 1 errore?

$$np(1-p)^{n-1}$$

Esercizio 15. Qual è la probabilità di avere esattamente l errori?

$$\binom{n}{l} p^l (1-p)^{n-l}$$

Esercizio 16. Qual è la probabilità di avere al **massimo** l errori?

$$\sum_{i=1}^{l} \binom{n}{i} p^i (1-p)^{n-i}$$

Esercizio 17. Qual è la probabilità di avere un numero pari di errori?

$$\sum_{i=0}^{\lceil \frac{n}{2} \rceil} \binom{n}{2i} p^{2i} (1-p)^{n-2i}$$

Ovvero usiamo una sommatoria da 0 a $\frac{n}{2}$ e consideriamo i numeri pari moltiplicando per 2 l'indice all'interno dell'espressione. Così, se n fosse 10, la sommatoria andrebbe a 0 a 5 e noi considereremmo solo i casi in cui gli errori sono $0, 2, 4, 6, 8, 10$, proprio come volevamo.

Questo ultimo esercizio è utile in certe situazioni.

12.3 Codici di correzione

Consideriamo alcuni codici noti che permettono di rilevare errori in fase di trasmissione e in alcuni casi di corregerli.

12.3.1 Single parity check code

Permette di identificare **un solo** errore all'interno della stringa.

Esempio 5. Data la stringa 0101010 calcoliamo il bit di parità in questo modo

$$\sum_{i=1}^{7} x_i \mod 2$$

In questo caso vale 1 e quindi mandiamo la stringa 01010101. Se un bit viene sbagliato il ricevente se ne accorge.

Per avere dei codici che permettano di rilevare, o addirittura correggere, degli errori bisogna aggiungere dei bit. Il numero di bit aggiunti è chiamata **ridondanza**. Formalmente è definita come

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

Dove con "bit di informazione" intendiamo quei bit che avrei dovuto realmente spedire. Nel caso del single parity check code la ridondanza è

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

e $\frac{1}{n-1}$ viene detta **ridondanza aggiunta**. Dobbiamo cercare un compromesso tra esattezza (affidabilità) e lunghezza del messaggio (efficienza).

12.3.2 Codice ASCII

Il codice ASCII fu pensato inizialmente per 7 bit. In questa versione viene aggiunto un ottavo bit a quelli di informazione, chiamato bit di parità. Il codice ASCII è usato per errori di tipo burst con il side effect che errori multipli possono auto-cancellarsi. Dato un messaggio si costrisce in questo modo il codice ASCII:

1. Prendiamo il messaggio
2. Lo convertiamo in binario
3. Aggiungiamo bit di parità per ogni parola
4. Per ogni colonna calcoliamo la checksum e aggiungiamo la parola ottenuta al messaggio.

Esempio 6. Dato la stringa $Hello\text{ß}NCTU$ dove ß rappresenta lo spazio.

PAROLA	BIT DI PARITA'	PAROLA IN BINARIO
110_8 = H =	0	1001000_2
145_8 = E =	0	1100101_2
154_8 = L =	0	1101100_2
154_8 = L =	0	1101100_2
157_8 = O =	0	1101111_2
040_8 = ß =	1	0100000_2
116_8 = N =	0	1001110_2
103_8 = C =	1	1000011_2
124_8 = T =	1	1010100_2
125_8 = U =	0	1010101_2
CHECKSUM	$\rightarrow$	1101110_2

12.3.3 Codici pesati

Come gli altri codici, i codici pesati aggiungono una checksum al messaggio. Questa viene calcolata in un modo particolare, che prende in considerazione la posizione dei simboli all'interno del messaggio, per questo sono detti "pesati". Per trovare la checksum procediamo nel seguente modo

MSG	$\sum$	$\sum\sum$
ω	ω	ω
x	$\omega + x$	$2\omega + x$
y	$\omega + x + y$	$2\omega + 2x + y$
z	$\omega + x + y + z$	$2\omega + 2x + 2y + z$
checksum ?	$\omega + x + y + z$	$2\omega + 2x + 2y + z + (\omega + x + y + z)$

Dove per la checksum non abbiamo aggiunto nulla, dato che non ne sappiamo il valore. Una volta ottenuto $2\omega + 2x + 2y + z + (\omega + x + y + z)$ possiamo ricavarla. Detta n la grandezza dell'alfabeto (cioè il numero di simboli) e r il resto della divisione tra $2\omega + 2x + 2y + z + (\omega + x + y + z)$ e n, la checksum è quel numero tale che

$$r + \text{checksum} \equiv_2 0$$

Esempio 7. Data la stringa $3B$ß8 trovare la checksum. Prima di tutto notiamo che il numbero di simboli è 37 ($\{0, \ldots, 9, A, \ldots, z, ß\}$ ha cardinalità 37). Calcoliamo il valore che interessa come spiegato prima

	MSG	$\sum$	$\sum\sum$
3	3	3	3
B	11	14	17
ß	36	50	67
8	8	58	125
checksum	??	58	183

Dividamo 183 per 37 per trovare il resto.

$$\frac{183}{37} = 4 \text{ resto } 35$$

Qual è quel numero che sommato a 35 è uguale a 0 in modulo 37? E' 2! Infatti

$$35 + 2 = 37 \equiv_{37} 0$$

Come controlliamo che abbiamo fatto giusto?

Parola		Posizione	Risultato
3	$\times$	5	15
11	$\times$	4	44
36	$\times$	3	108
8	$\times$	2	16
2	$\times$	1	2
TOTALE			185

Per essere corretto deve valere che

$$185 \equiv_{37} 0$$

Siccome vale allora abbiamo fatto i calcoli correttamente.

Esercizio 18. Controllare che se riceviamo $3B82$ il messaggio è sbagliato.

Parola		Posizione	Risultato
3	×	4	12
11	×	4	33
8	×	3	16
2	×	2	2
TOTALE			63

$63 \equiv_{37} 26$ quindi è sbagliato, dato che dovrebbe essere 0.

Esercizio 19. Controllare se il messaggio $3B$ß28 è sbagliato o meno.

Parola		Posizione	Risultato
3	×	5	15
36	×	4	44
5	×	3	111
2	×	2	4
8	×	1	8
TOTALE			182

182 modulo 37 fa 34 quindi è sbagliato.

12.3.4 Codici (M,N)

Nelle lezioni precedenti abbiamo definito il canale con la tripla

$$< \mathbb{X}, \mathbb{Y}, p(y|x) >$$

Diamo adesso la definizione di canale su cui vengono spediti n messaggi

$$< \mathbb{X}^n, \mathbb{Y}^n, p(y^n|x^n) >$$

Siccome siamo su canali senza memoria sappiamo che

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

. Un codice (M, N) è tale che

- M è la lunghezza del messaggio spedito sul canale (i simboli sono numerati da 1 a M).
- N è il numero di volte che viene utilizzato il canale. Notare che a ogni utilizzo del canale possiamo scriver o il bit 0 o il bit 1. Quindi utilizzandolo N volte scriviamo N bit, cioè un messaggio al massimo di dimensione 2^N.

Introduciamo due funzioni:

- Funzione di codifica

$$x^q : \{1, \ldots, M\} \to \mathbb{X}^q$$

- Funzione di decodifica

$$g : \mathbb{Y}^q \to \{1, \ldots, M\}$$

Definiamo la probabilità di errore sull'i-esimo simbolo come

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

La probabilità massima di errore è

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

Invece la probabilità media di errore è

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

Vale la seguente disequazione

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

Il **tasso di trasmissione** di un codice di tipo (m, n) è dato da

$$R = \frac{\log_2 M}{n}$$

dove la base del logaritmo è 2 perché siamo in binario. La lunghezza massima del messaggio che possiamo spedire è $M = 2^N$, questo è possibile solamente senza rumore. Il tasso di trasmissione in questo caso è $R = 1$.

12.4 II Teorema di Shannon

Siamo finalmente pronti a enunciare il secondo teorema di Shannon.

Teorema II Teorema di Shannon

Sia $< \mathbb{X}, \mathbb{Y}, p(y|x) >$ un canale con capacità c. $\forall R < c \exists k_1, k_2, \ldots$ sequenze di codici dove k_n è di tipo $(2^{nR_n}, n)$ tale che

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

e

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

Notiamo che R_n indica il rumore, se è $= 1$ allora non c'è rumore, più si avvicina a 0 più ce n'è. Quello che ci dice questo teorema è che, a) più il messaggio è grande più il rumore, qualsiasi esso sia (ma sempre minore di c) diventa "trascurabile", e b) che la probabilità massima di errore tende a 0 con il crescere della lunghezza del messaggio.

Capitolo 13

Lezione XIII

13.1 Codici di rilevazione errori

In questa lezione vediamo alcuni esempi di codici reali.

13.1.1 ISBN-10

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

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

Dove x indica il 10. $ISBN-10$ è un codice pesato.

Esempio 8. Il codice

$$0-471-24195-4$$

è un codice $ISBN-10$, ed è tale che:

- 0 è la nazione, nello specifico 0 indica i paesi anglofoni.
- 471 è l'id-publisher.
- 24195 è il book-number.
- 4 è l'error detection.

Esercizio 20. Dato il codice

$$0 - 52 - 18 - 4868 - 7$$

controllare se è corretto. Prima di tutto notiamo che è scritto in modo leggermente diverso da prima. In realtà sono identici e $18 - 4868$ identica il publisher id in questo caso. Per controllare se è corretto procediamo come segue:

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

Per essere corretto $209 \equiv_1 10$, e lo è, infatti $\frac{209}{11} = 19$ resto 0.

Si lascia al lettore il seguente esercizio:

Esercizio 21. Verificare che il codice $0 - 471 - 24195 - 4$ sia corretto. (Risposta: E' corretto, $176 \equiv_{11} 0$)

13.1.2 UPC

Il codice UPC (Universal Product Code) è il comune codice a barre. Non è un codice pesato ma di parita ed è a 12 cifre.

Esempio 9. Un codice UPC è fatto nel seguente modo

$$\underbrace{036000}_{\substack{\text{ID} \\ \text{PRODUTTORE}}} \quad \underbrace{29145}_{\substack{\text{ID} \\ \text{PRODOTTO}}} \quad \underbrace{2}_{\text{CHECKSUM}}$$

Come capiamo se un codice è corretto? Sommiamo le cifre dispari e le moltiplichiamo per 3 e poi sommiamo le cifre pari (iniziamo a contare da 1). Il risultato deve essere 0 in modulo 12. Proviamo con il codice nell'esempio:

$$\begin{aligned} &3(6+2+1+5)+(3+9+4+2)= \\ &=3(14)+18=42+18=60\equiv_{12}0 \end{aligned}$$

13.2 Codici di rilevazione e correzione

Fino a ora abbiamo visto solamente codici che possono rilevare gli errori ma non correggerli. Ci concentriamo adesso sui codici che possono anche correggerli. Supponiamo di avere una stringa da spedire

$$(x_1, x_2, x_3)$$

Aggiungiamo dei bit di controllo (x_4, x_5, x_6) che sono definiti come

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

Quindi quello che spediremo sarà

$$\underbrace{(x_1, x_2, x_3}_{\substack{\text{BIT} \\ \text{DI INFORMAZIONE}}}, \overbrace{x_4, x_5, x_6)}^{\substack{\text{BIT DI} \\ \text{CONTROLLO}}}$$

Dal lato ricevente dovrà valere che

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

SE SBAGLIO	VENGONO ERRATI
x_1	y_4, y_5
x_2	y_4, y_6
x_3	y_5, y_6
x_4	y_4
x_5	y_5
x_6	y_6

Questo codice riesce quindi a correggere 1 errore, ma non di più. Riesce a rilevare il doppio errore ma non riesce a correggerlo.

SE SBAGLIO	VENGONO ERRATI
x_1, x_2	y_5, y_6
x_1, x_3	y_4, y_6
x_1, x_4	y_5
x_1, x_5	y_4
x_1, x_6	y_4, y_5, y_6

Proviamo ad ampliare il nostro codice:

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

Il nostro codice peggiore, facciamo retromarcia.

13.2.1 Codice a ripetizione tripla

$$\begin{cases} x_2 = x_3 = 0 & \text{se } x_1 = 0 \\ x_2 = x_3 = 1 & \text{se } x_1 = 1 \end{cases}$$

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

RICEVO	SCELGO
001	0
010	0
011	1
100	0
101	1
110	1

Esercizio 22. Sia $p_e = p$ la probabilità di errore, qual è la probabilità di scegliere 0?

$$p(\text{di scegliere } 0) = \frac{1}{2}$$

ed è uguale alla probabilità di scegliere 1.

Esercizio 23. Qual è la probabilità di inviare 000 e di ricevere 001?

$$p(w\ 000, r\ 001) = p(w\ 000)p(r\ 001|w\ 000) = \frac{1}{2}p(1-p)^2$$

Esercizio 24. Qual è la probabilità di inviare 111 e di ricevere 001?

$$p(w\ 111, r\ 001) = p(w\ 111)p(r\ 001|w\ 111) = \frac{1}{2}p^2(1-p)$$

Esercizio 25. Qual è la probabilità di ricevere 001?

$$p(r\ 001) = p(w\ 000, r\ 001) + p(w\ 111, r\ 001) = \frac{1}{2}p^2(1-p) + \frac{1}{2}p(1-p)^2$$

$$= \frac{1}{2}p(1-p)(p + (1-p)) = \frac{1}{2}p(1-p)$$

Un altro codice a rilevazione doppia e correzione singola è

$$\begin{cases} x_1 + x_2 = 0 \\ x_1 + x_3 = 0 \end{cases}$$

Possiamo scriverla in forma matriciale

$$M = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad x^T = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

13.2.2 Codici di tipo (n,k)

Sia C un codice di tipo (n, k), ovvero che mappa k bit (chiamiamola s) in una stringa binaria di n bit. C è un codice linare se esiste una matrice $G_{k\times n}$ e $H_{(n-k)\times n}$ tale che

$$x = (x_1, \ldots, x_n) = (s_1, \ldots, s_k) \cdot G$$

e vale anche che $Hx^T = 0^T$, dove x è la parola del codice e s il messaggio binario associato.

Definzione 9.

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

viene detto **rate**, ovvero quanti bit inviati per ogni bit di informazione.

Codice di Hamming

Il codice di Hamming è un codice di tipo $(7, 4)$, quindi con rate $R = \frac{7}{4} = 1.75$. Ha le stesse caratteristiche del codice a rilevazione tripla, ma utilizza meno bit.

$$\underbrace{(p_1, p_2, p_3}_{\text{BIT DI CONTROLLO}}, \overbrace{s_1, s_2, s_3, s_4)}^{\text{BIT DI INFORMAZIONE}}$$

Dove p_1, p_2, p_3 sono calcolati come

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

In forma matriciale

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

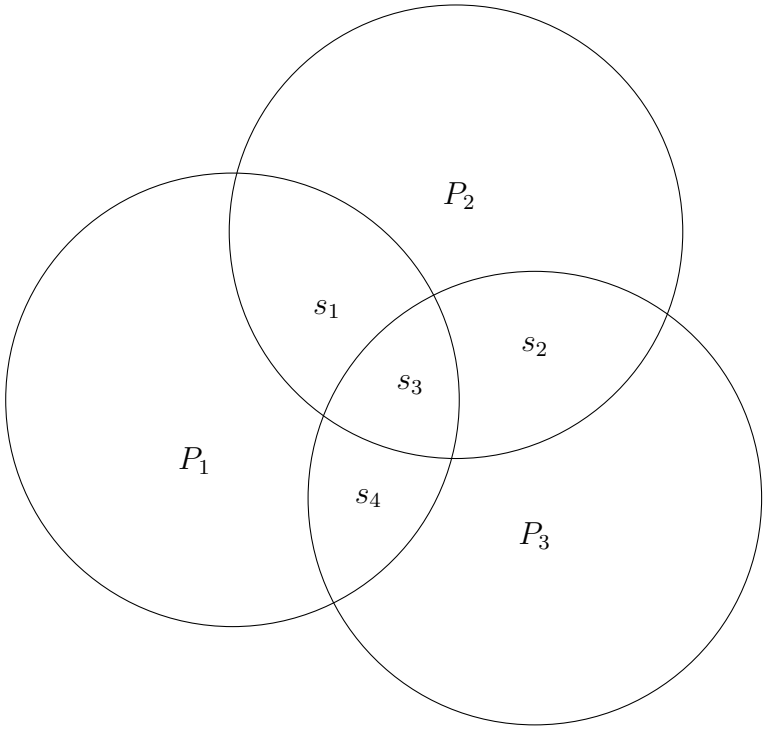
Esprimiamo la matrice generatrice come:

$$\mathbf{G} = \begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \\ s_4 \end{array} \begin{array}{c} \begin{array}{ccccccc} p_1 & p_2 & p_3 & & & & \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Esempio 10. Spediamo il messaggio 0111100 e riceviamo $p_1 = 0, p_2 = 1, p_3 = 1$ (quindi abbiamo ricevuto 011100011), rilevare se c'è stato un errore e correggerlo.
Prendiamo la matrice H e la moltiplichiamo per il vettore messaggio

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Per essere corretto il vettore risultante (chiamato **sindrome**) dovrebbe essere composto da soli 0. Se nella posizione i c'è un 1 allora vuol dire che p_i è errato. In questo caso sono sbagliati p_1 e p_2. Come facciamo a sapere qual è il bit errato? Usiamo il diagramma di **McEliece**.



Nel nostro caso abbiamo il bit errato era s_1. Il messaggio ricevuto è della somma del messaggio inviato con l'errore (ignorando i bit p_1, p_2, p_3)

$$y = x + e = (0110100) + (0001000)$$

Inoltre vale

$$Hy = Hx^T + He^T$$

dove Hx^T vale 0 nel caso il messaggio sia corretto.

Esercizio 26. Verificare se il messaggio ricevuto $y = (1110100)$ sia errato.

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Usando McEliece ci accorgiamo che p_1 è il bit errato. Lo si può correggere trasformandolo in 0.

Capitolo 14

Lezione XIV

14.1 Campi di Galois

In questa sezione discutiamo i campi di Galois, che ci serviranno nel contesto dei codici ciclici. Solitamente noi lavoriamo su un 1 byte, cioè 8 bit. Avendo 8 bit possiamo esprimere al massimo 256 numeri diversi (da 0 a 255). Abbiamo però un problema. I 256 possibili valori formano un anello, e su un anello non è garantito che un elemento abbia l'inverso. La domanda che sorge spontanea è, che cosa ce ne frega dell'inverso? Per rispondere a questa domanda vediamo prima di tutto cos'è l'inverso. L'inverso di un numero a è quel valore tale che

$$a \cdot \frac{1}{a} \equiv_{256} 1$$

Nel contesto dei codici cicli ci servirà l'operazione di divisione. Il problema è che se un numero non ha inverso, non può dividere un altro numero.

Esempio 11. Consideriamo l'anello $\mathbb{Z}_8$ e l'elemento 2. 2 non ha inverso in $\mathbb{Z}_8$, infatti

$$4 \cdot 1 \equiv_8 4$$

$$4 \cdot 2 \equiv_8 0$$

$$4 \cdot 3 \equiv_8 4$$

$$4 \cdot 4 \equiv_8 0$$

$$4 \cdot 5 \equiv_8 4$$

$$4 \cdot 6 \equiv_8 0$$

$$4 \cdot 7 \equiv_8 4$$

Allora se prendiamo l'elemento 1 del campo e lo proviamo a dividere per 4, la divisione non si può fare. Infatti non esiste x tale che $4 \cdot x = 1$.

La mancanza dell'inverso non preclude la divisione, infatti se gli elementi hanno un fattore comune la divisione potrebbe essere possibile anche senza inverso.

Esempio 12. Consideriamo l'anello $\mathbb{Z}_6$ e l'elemento 2. 2 non ha inverso in $\mathbb{Z}_6$, infatti

$$2 \cdot 1 \equiv_6 2$$

$$2 \cdot 2 \equiv_6 4$$

$$2 \cdot 3 \equiv_6 0$$

$$2 \cdot 4 \equiv_6 2$$

$$2 \cdot 5 \equiv_6 4$$

Allora se prendiamo l'elemento 4 del campo e lo proviamo a dividere per 4, la divisione si può fare. Infatti esiste x tale che $2 \cdot x = 4$, ed è proprio 2. Questo è possibile perché hanno un fattore comune. Invece, per esempio, 5 non è divisibile, infatti non esiste alcun x che soddisfi $2 \cdot x = 5$.

Perché serve l'inverso per poter dividere un numero? Si consideri l'operazione di divisione sotto-forma di moltiplicazione

$$b \div a = b \cdot a^{-1}$$

se l'inverso non esiste, non possiamo svolgere l'operazione. Se però esiste un fattore comune allora $b = d \cdot b'$ e $a = d \cdot a'$, ne consegue che

$$\frac{b}{a} = \frac{d \cdot b'}{d \cdot a'} = \frac{b'}{a'}$$

E' ovvio che se b' e a' non possano essere semplificati ulteriormente e se a' non ha inverso allora l'operazione non è possibile.
Lavorare su un anello è problematico. Se però 256 fosse primo, allora ci troveremmo in presenza di un campo, il quale garantisce l'inverso per tutti i suoi elementi.

Esercizio 27. Determinare se Z_{10} è un campo o meno. Soluzione: Z_{10} non è un campo, infatti 2 non ha inverso in Z_{10}.

Esercizio 28. Determinare se Z_{15} è un campo o meno. Soluzione: In Z_{15} 2 ha inverso, infatti $2^{-1} = 8$. Problema: l'elemento 3 (che in Z_{10} aveva inverso) non ha inverso in Z_{15}. Quindi Z_{15} non è un campo.

Quindi, in generale, $Z_{(n)}$ con n non primo, ovvero composto, è un anello. Invece, Z_p con p primo è un campo.

Esercizio 29. Determinare se Z_7 è un campo o meno. Soluzione Z_7 è un campo, infatti:

$$2 \cdot 4 \equiv_7 1$$
$$3 \cdot 5 \equiv_7 1$$
$$4 \cdot 2 \equiv_7 1$$
$$5 \cdot 3 \equiv_7 1$$
$$6 \cdot 6 \equiv_7 1$$

Generalmente in un anello Z_n un elemento a ha inverso se e solo se a è co-primo con n, ovvero $MCD(a, n) = 1$.

14.1.1 Generatore

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

Esempio 13. Proviamo a vedere se 4 è generatore di $\mathbb{Z}_7$

$$4^0 \equiv_7 1$$
$$4^1 \equiv_7 4$$
$$4^2 \equiv_7 2$$
$$4^3 \equiv_7 1$$
$$4^4 \equiv_7 4$$
$$4^5 \equiv_7 2$$
$$4^6 \equiv_7 1$$

4 non è un generatore per $\mathbb{Z}_7$

Notiamo che appena troviamo un 1 i valori si ripetono. Se il numero è un generatore trovo 1 solamente all'inizio e alla fine. Se trovo un 1 in mezzo so già che non è un generatore. Infatti, vale il seguente teorema:

Teorema Piccolo teorema di Fermat

Vale la seguente uguaglianza

$$\alpha^{p-1} \equiv_p 1$$

Per essere più veloce con i conti, non c'è bisogno di calcolare, per esempio 4^5, ma basta prendere il risultato di 4^4 e moltiplicarlo per 4. Nell'esempio sopra, 4^4 è uguale a 4, $4 \cdot 4 \equiv_7 2$. Molto più veloce di fare $4^5 = 1024 \equiv_7 2$.

Esercizio 30. Verificare se 3 è un generatore per $\mathbb{Z}_7$

$$3^0 \equiv_7 1$$
$$3^1 \equiv_7 3$$
$$3^2 \equiv_7 2$$
$$3^3 \equiv_7 6$$
$$3^4 \equiv_7 4$$
$$3^5 \equiv_7 5$$
$$3^6 \equiv_7 1$$

3 è un generatore per $\mathbb{Z}_7$.

Esercizio 31. Verificare se 2 è un generatore per $\mathbb{Z}_7$

$$2^0 \equiv_7 1$$
$$2^1 \equiv_7 2$$
$$2^2 \equiv_7 4$$
$$2^3 \equiv_7 1$$

2 non è un generatore per $\mathbb{Z}_7$.

Esercizio 32. Verificare se 5 è un generatore per $\mathbb{Z}_7$

$$5^0 \equiv_7 1$$
$$5^1 \equiv_7 3$$
$$5^2 \equiv_7 2$$
$$5^3 \equiv_7 6$$
$$5^4 \equiv_7 4$$
$$5^5 \equiv_7 5$$
$$5^6 \equiv_7 1$$

5 è un generatore per $\mathbb{Z}_7$.

Come abbiamo visto, possono esistere più generatori all'interno di un campo. Possiamo mappare i due generatori tra di loro. In questo caso il campo è detto **isomorfo**.

14.1.2 Definizione campi di Galois

I campi di Galois ci permettono di avere un campo anche se il numero non è primo. Un campo di Galois è formalmente

$$GF(p^n)$$

dove p è un numero primo e n un numero naturale. Alcuni campi particolari sono $GF(11^1)$ e $GF(7^1)$. Nel nostro caso

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

$$GF(2) \xrightarrow{\text{ESTENSIONE}} GF(2^8)$$

Gli elementi di un campo di Galois sono dei polinomi di grado $n-1$ e con coefficienti in $\mathbb{Z}^p$. Nel nostro caso hanno grado al massimo 7 e coefficienti in $\{0,1\}$.

Esempio 14. Il seguente polinomio rappresenta un byte

$$x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x^1 + x^0$$

che rappresenta il byte di valore

1	1	1	1	1	1	1	1

Esempio 15. Il seguente polinomio

$$x^6 + x^5 + x^3 + x^1$$

il byte di valore

0	1	1	0	1	0	1	0

Facciamo un po' di calcoli per prenderci la mano.

Esempio 16. Prendiamo due polinomi $a(x), b(x) \in GF(2^8)$:

$$a(x) = x^6 + x + 1$$
$$b(x) = x^4 + x^2 + x$$

Innanzitutto notiamo che la loro somma darà sempre un elemento del campo.

$$a(x) + b(x) = x^6 + x^4 + x^2 + 1$$

Invece la moltiplicazione non è detto

$$a(x) \cdot b(x) = x^10 + x^5 + x^4 + x^8 + x^3 + x^2 + x^7 + x^2 + x$$
$$= x^10 + x^8 + x^7 + x^5 + x^4 + x^3 + x$$

Notiamo che x^{10} e x^8 sono fuori dal campo, dobbiamo riuscire a ridurli, cioè dobbiamo prenderne il modulo.

$$\begin{array}{l|l} c(x) & m(x) \\ \hline r(x) & q(x) \end{array}$$

Quindi $a(x)b(x) \equiv_{m(x)} r(x)$. Vogliamo che il valore ottenuto sia all'interno di un campo e non di un anello. Dobbiamo scegliere $m(x)$ irriducibile. Usiamo per esempio $m(x) = x^8 + x^4 + x^3 + x + 1$ (usato con AES). Non è l'unico polinomio che possiamo utilizzare, ce ne sono diversi (per $GF(2^8)$ ce ne sono circa 30). Se utilizzando due polinomi diversi troviamo due resti diversi, $r(x)$ e $r'(x)$ allora ci troviamo in un campo isomorfo, e potremmo mapparli tra di loro. Concludiamo l'esercizio di prima:

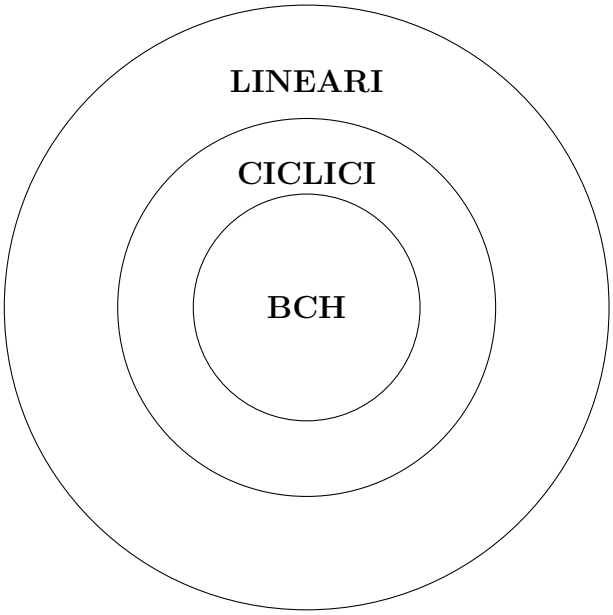
$x^{10} + x^8 + x^7 + x^5 + x^4 + x^3 + x$	$x^8 + x^4 + x^3 + x + 1$
$x^{10} + x^6 + x^5 + x^3 + x^2$	$x^2 + 1$
$x^8 + x^7 + x^6 + x^4 + x^2 + x$	
$x^8 + x^4 + x^3 + x + 1$	
r(x) $= x^7 + x^6 + x^3 + x^2 + 1$	

Quindi

$$a(x) \cdot b(x) \equiv_{m(x)} x^7 + x^6 + x^3 + x^2 + 1$$

1	1	0	0	1	1	0	1

14.2 Codici ciclici



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

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

e il polinomio generatore è

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

I codici ciclici possono essere

- Sistematici: Posizioni prestabilite per bit di controllo e di informazione. Parola di codice $(v(x)$ rappresentata come

  $$v(x) = m(x) \cdot g(x)$$

  dove $m(x)$ è il messaggio da spedire.

- Non sistematici: Posizione bit di controllo non prefissata. Parola di codice rappresentata come

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

Vediamo un esempio

Esempio 17. Dato il codice $(7,4)$ con $g(x) = 1+x+x^3$ e $m(x) = 1+x^2+x^3$ trovare la parola di codice corrispondente.

- ❒ CASO SISTEMATICO:

$$\begin{aligned} v(x) &= m(x)g(x) \\ &= (1+x^2+x^3)(1+x+x^3) \\ &= x^6+x^5+x^4+x^3+x^2+x+1 \end{aligned}$$

- ❒ CASO NON SISTEMATICO:

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

Cominciamo con il calcolare $x^{n-k}m(x)$

$$x^3(x^3+x^2+1) = x^6+x^5+x^3$$

Dividiamo per $g(x) = 1+x+x^3$. Facendo i conti (lasciati da fare al lettore) troviamo che

$$v(x) = x^3+x^5+x^6+1$$

14.2.1 Matrice generatrice

Per trovare le parole di codice possiamo anche usare la matrice generatrice. Ecco come,

1. Scriviamo la matrice generatrice

$$G = \begin{bmatrix} x^{k-1} & \cdot & g(x) \\ x^{k-2} & \cdot & g(x) \\ \vdots & \vdots & \vdots \\ x^{k-k} & \cdot & g(x) \end{bmatrix}$$

2. Se vogliamo ottenere le parole del codice nella forma sistematica, applichiamo combinazione lineare di righe per ottenere

$$G = \left[\begin{array}{c|c} I & \quad \end{array}\right]$$

(dove I è la matrice identità).

3. Trovo le parole di codice

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

dove c è la parola di codice e d è il messaggio (come s nei codici a ripetizione tripla).

Definiamo il polinomio di parità come

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

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

Esercizio 33. Dato il codice $(7, 3)$ con $g(x) = x^4 + x^2 + x + 1$, trovare tutte le possibili parole di codice.

Capitolo 15

Lezione XV

15.1 Correzione codice ciclico

Sia c un codice ciclico $(7, 4)$. Prima di tutto vogliamo trovare il polinomio generatore. Sappiamo che il grado di $g(x)$ è

$$\text{grado}[g(x)] = 7 - 4 = 3$$

Inoltre $g(x)$ deve dividere $x^n - 1 = x^7 - 1$ (oppure $x^7 + 1$ dato che siamo in binario e sono uguali). Sapendo che

$$x^7 - 1 = (x + 1)(x^3 + x + 1)(x^3 + x^2 + 1)$$

Abbiamo due possibili candidati, ovvero x^3+x+1 e x^3+x^2+1. Ne scegliamo uno a caso, per esempio $x^3 + x^2 + 1$. A questo punto possiamo definire le parole di codice $p(x)$ come

$$p(x) = g(x) \cdot a(x)$$

Dove $a(x)$ è il messaggio non codificato. Assumiamo di volere trasmettere $x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$, e che ci sia del rumore e x^4 venga sbagliato. Come lo correggiamo?

15.1.1 Algoritmo di correzione

Cominciamo con il dare la seguente definizione

Definzione 10. Il **peso di Hamming** è il numero di 1 che una parola ha.

Per esempio la parola 01000100 ha peso di hamming 2 (scriviamo $\omega(01000100) = 2$ per indicarlo, in generale $\omega(s(x))$). Prima di definire l'algoritmo di correzione, diciamo che il resto $r(x)$ della divisione tra $p(x)$ e $g(x)$ è anche indicato con $s(x)$, ovvero la **sindrome**. Quindi, data una parola $p(x)$ per correggerla utilizziamo il seguente algoritmo:

1. Calcoliamo $s(x)$.
2. Se $\omega(s(x)) \leq t$, dove t è il numero di errori, allora possiamo correggere la parola. Altrimenti continuiamo con l'algoritmo.
3. Se $\omega(s(x)) > t$ allora prendiamo $k = 0$ e lo incrementiamo di 1. Poi facciamo shift ciclico a destra (che poi nei fatti il prof. lo fa a sinistra?) della parola. Otteniamo $p'(x)$.
4. Calcoliamo $s'(x)$.
5. Ripartiamo dal punto (2).

Vale il seguente teorema

Teorema Termine algoritmo

Dopo un certo numero di passi il codice ciclico è tale che $\omega(s(x)) \leq t$.

Una volta terminato l'algoritmo, per correggere la parola basta fare

$$\begin{aligned} p(x) &= y(x) + ShiftSinistra_k(s_f(x)) \\ &= y(x) + ShiftDestra_{n-k}(s_f(x)) \end{aligned}$$

dove $y(x)$ è la parola ricevuta, k il contatore e s_f l'ultima sindrome calcolata.

Esempio 18. Dato $g(x) = x^3 + x^2 + 1$ e la parola ricevuta $y(x) = x^6 + x^5 + x^3 + x^2 + x + 1$ applicare l'algoritmo per correggerla, sapendoo che vogliamo correggere 1 errore ($t = 1$). Cominciamo con la prima divisione.

$x^6+x^5+x^3+x^2+x+1$	x^3+x^2+1
$x^6+x^5+x^3$	x^3
$s(x)=x^2+x+1$	

La sindrome ha peso $\omega(s(x))=3$. t è 1 e quindi $3>1$, procediamo con l'algoritmo. Poniamo $k=0$ e la incrementiamo di 1, $k=1$. Shiftiamo la parola e otteniamo

$$\begin{aligned} p'(x) &= p(x)\cdot x - x^7 + 1 \\ &= x^6+x^4+x^3+x^2+x+1 \end{aligned}$$

dove sommiamo $-x^7+1$ perché è uno shift ciclico (leviamo l'elemento x^7 che è fuori dal campo e sommiamo 1 per farlo riapparire dall'altra parte). Continuiamo con la seconda divisione

$x^6+x^4+x^3+x^2+x+1$	x^3+x^2+1
$x^6+x^5+x^3$	x^3+x^2
$x^5+x^4+x^2+x+1$	
$x^5+x^4+x^2$	
$s'(x)=x+1$	

Il peso è $\omega(s'(x))=2$, quindi $t<\omega(s'(x))$. k diventa $k=2$ e

$$p''(x)=x^5+x^4+x^2+x+1$$

Dividiamo $p''(x)$:

$x^5+x^4+x^3+x^2+x+1$	x^3+x^2+1
$x^5+x^4+x^2$	x^2+1
x^3+x+1	
x^3+x^2+1	
$s''(x)=x^2+x$	

Il peso è $\omega(s'(x)) = 2$, quindi $t < \omega(s'(x))$. k diventa $k = 3$ e

$$p'''(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x$$

Dividiamo $p'''(x)$:

$x^6 + x^5 + x^4 + x^3 + x^2 + x$	$x^3 + x^2 + 1$
$x^6 + x^5 + x^3$	$x^3 + x + 1$
$x^4 + x^2 + x$	
$x^4 + x^3 + x$	
$x^3 + x^2$	
$x^3 + x^2 + 1$	
$s'''(x) = 1$	

Il peso è $\omega(s'(x)) = 1$, quindi $t = \omega(s'(x))$. Terminiamo l'algoritmo. Possiamo ora correggere la parola. Abbiamo $k = 3$, facciamo $n - k$ shift a destra (che sono a sinistra in realtà), shiftiamo $p(x)$ di 4 posizioni.

$$p(x) + 1 \cdot x^4 = x^6 + x^5 + x^4 + x^3 + x^2 + 1$$

che è il messaggio che è stato inviato.

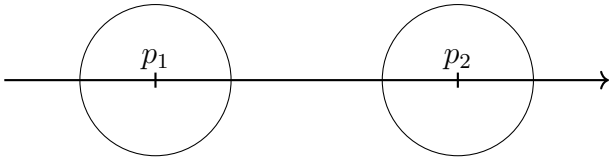
Esercizio 34. Si svolga lo stesso esercizio con $g(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$, $y(x) = x^7 - 1$ e $t = 3$.

15.2 Codici BCH

Se questa parte vi sembra confusionaria è perché lo è. Il professore l'ha spiegata in maniera sbrigativa all'ultima lezione.

I codici BCH vengono costruiti in base a quanti errori vogliamo correggere. Se vogliamo correggere t errori, le parole devono essere distanti almeno $2t+1$ tra di loro.

SFERE DI COLLISIONE



Se le sfere di collisione di due parole collidono e non sono almeno di raggio t non riusciamo a correggerle.

15.2.1 Ampliamento algebrico

Dato un polinomio che non ha radice nel campo gliene imponiamo una noi. Quindi prendiamo un polinomio irriducibile e supponiamo che abbia come soluzione una certa radice. Le radici devono essere tali che

$$\beta_1, \beta_2, \ldots, \beta_m \equiv_n 1$$

Ma quante radici ci servono? Dipende dal numero di errori che vogliamo correggere e dal campo in cui mi trovo. Ad esempio se siamo in $GF(11)$ e vogliamo correggere 2 errori, il numero M di radici che ci servono è il più piccolo numero tale che

$$2^M \equiv_{11} 1$$

In questo caso è 10. A questo punto troviamo gli elementi del campo e costruiamo $g(x)$ come il minimo comune multiplo tra gli elementi del campo.

Esempio 19. Ampliamo il campo di Galois (2), con coefficienti in $\mathbb{Z}_2$. Scegliamo un polinomio irriducibile di grado 3, così otteniamo $GF(2^3)$ (che interessa a noi dato che lavoriamo sui byte),

$$p(x) = x^3 + x + 1$$

Definiamo gli elementi del campo , ovvero tutti i polinomi di grado minore di 3:

$$\{0, 1, \beta, \beta + 1, \beta^2 + \beta^2 + 1, \beta^2 + \beta, \beta^2 + \beta + 1\}$$

In questo modo abbiamo effettuato l'espansione, passando da 2 elementi del campo $GF(2)$ (ovvero 0 e 1) a 8 elementi in $GF(2^3)$. Troviamo ora le radici del campo espanso. Imponiamo che β sia radice ponendo $p(\beta) = 0$:

$$p(\beta) = \beta^3 + \beta + 1 = 0$$
$$\beta^3 = \beta + 1$$

Siccome grado deve essere < 3 scriviamo

$$\beta \cdot \beta^2 = \beta + 1$$

A questo punto vogliamo trovare qual è l'ordine del campo. Quindi cerchiamo una potenza tale che $\beta^i = 1$.

$$\beta^4 = \beta^3 \cdot \beta = (\beta + 1) \cdot \beta = \beta^2 + \beta$$
$$\beta^5 = \beta^2 + \beta + 1$$
$$\beta^6 = \beta^2 + 1$$
$$\beta^7 = 1$$

Abbiamo trovato l'ordine dal campo, ovvero 7, dato che β elevato alla 7 restituisce 1. Da qui in poi le potenze si ripetono, quindi $\beta^8 = \beta$, $\beta^9 = \beta^2$, $\beta^{10} = \beta + 1$, etc... . In questo modo abbiamo costruito una mappatura (la colonna più a destra è la rappresentazione in byte)

0	0	000
1	β^7	001
β	β^1	010
$\beta + 1$	β^3	011
β^2	β^2	100
$\beta^2 + 1$	β^6	101
$\beta^2 + \beta$	β^4	110
$\beta^2 + \beta + 1$	β^5	111

La mappatura ci permette calcoli più veloci, ad esempio:

$$(\beta^2 + 1)(\beta^2 + \beta + 1) =$$
$$= \beta^6 \cdot \beta^5 = \beta^7 \cdot \beta^4$$
$$= \beta^4 = \beta^2 + \beta$$

Teorema Costruzione codici BCH

$\forall n \in \mathbb{N}$, dato r numero primo e $GF(r^{\phi(n)})$ (dove ϕ è la funzione di Eulero), siano $\beta^1, \ldots, \beta^n$ radici per costruire il codice ciclico. Allora $\omega(t)$ è l'insieme dei polinomi

$$\omega(w) = \omega_0 + \omega_1 x + \cdots + \omega_{n-1} x^{n-1}$$

tali che hanno radici $\omega(\beta^1) = \omega(\beta^2) = \cdots = \omega(\beta^n) = 0$. Queste formano esattamente un codice ciclico che corregge n errori.