

## Esercizio 2.1 Cifri a blocchi e ridondanza

COSCHIAU  
MARCO

pag 1

[a] Sia  $p$  una distribuzione di probabilità sull'alfabeto  $A$ .  $\sigma$  è una sequenza di  $m$  lettere

$$\sigma = x_1, x_2, \dots, x_m$$

Dato un blocco  $\sigma$  di  $m$  lettere, quindi  $\sigma \in A^m$ ;  $f(a) \triangleq \#$  occorrente di  $a$  in  $\sigma$

Un blocco  $\sigma$  è tipico se  $\forall a \in A$ ,  $f(a) \approx m \cdot p(a)$

- Trovare formula per  $p(\sigma)$

$\sigma$  è una sequenza di lettere  $\rightarrow \sigma = x_1, x_2, \dots, x_m \rightarrow p(\sigma) = p(x_1, \dots, x_m)$

Dato che ogni lettera di  $\sigma$  è indipendente dalle altre

$$p(\sigma) = p(x_1) \cdot \dots \cdot p(x_m)$$

Riordinando gli elementi  $x_i$  in modo che le prob. delle singole lettere siano vicine si ottiene

$$p(\sigma) = \prod_{a \in A} p(a)^{f(a)}$$

esempio

$$\sigma = a, d, d, b, c, d$$

$$\begin{aligned} \Rightarrow p(\sigma) &= p(a) \cdot p(d) \cdot p(d) \cdot p(b) \cdot p(c) \cdot p(d) \\ &= p(d) \cdot p(d) \cdot p(d) \cdot p(a) \cdot p(b) \cdot p(c) \\ &= p(d)^3 \cdot p(a) \cdot p(b) \cdot p(c) \end{aligned}$$

Nel caso dei blocchi tipici

$$f(a) \approx m \cdot p(a)$$

e pertanto

$$p(\sigma) = \prod_{a \in A} p(a)^{f(a)} \approx \prod_{a \in A} p(a)^{mp(a)}$$

per i blocchi  $p(\sigma)$  dipende solo da  $p(a)$  e dal parametro fissato

[b] Nei  $\sigma$  tipici è possibile approssimare  $p(\sigma) \approx 2^{-mH(p)}$  dove

$$H(p) = -\sum_{a \in A} p(a) \cdot \log(p(a))$$

è detta ENTRPIA, è misura il disordine di una distribuzione. Più gli elementi della distribuzione sono equiprobabili, più l'entropia è alta  $\rightarrow \log|A|$  è il valore massimo

- Dimostrare l'approssimazione (consiglio usare i logaritmi e poi l'elusamento a potenze)

Nel caso dei blocchi  $\sigma$  tipici

$$P(\sigma) \approx \prod_{\alpha \in A} P(\alpha)^{m_{P(\alpha)}}$$

Se si applica il logaritmo ad entrambi i membri si ha

$$\log(P(\sigma)) \approx \log\left(\prod_{\alpha \in A} P(\alpha)^{m_{P(\alpha)}}\right)$$

Per le proprietà dei logaritmi, il logaritmo di un prodotto è pari alla somma dei logaritmi. Nel caso del logaritmo di una produzione, il risultato è pari alla sommatoria dei logaritmi

$$\log(P(\sigma)) \approx \sum_{\alpha \in A} \log(P(\alpha)^{m_{P(\alpha)}})$$

Sfruttando la regola dell'esponente

$$\log(P(\sigma)) \approx \sum_{\alpha \in A} m_{P(\alpha)} \log(P(\alpha)) = m \underbrace{\sum_{\alpha \in A} P(\alpha) \log(P(\alpha))}_{-H(P)}$$

Sostituendo la definizione di entropia ed elevando al quadrato i due membri

$$2^{\log(P(\sigma))} \approx 2^{-mH(P)}$$

ottengo

$$P(\sigma) \approx 2^{-mH(P)}$$

[C] Fissato un blocco  $\sigma_0$ , quanti blocchi, in media, devo guardare prima di trovare  $\sigma_0$ ?

esempio: lancio di un dado

la probabilità che si ottenga una delle facce, per esempio la #4 è pari ad  $1/6$ . Per il th. di Bernoulli, sappiamo che il numero di lanci medi che devono essere fatti è pari a

$$n = \frac{1}{P(A)} = \frac{1}{1/6} = 6 \text{ lanci}$$

Possiamo fare lo stesso ragionamento alle singole lettere, che possiamo vedere come blocchi di dimensione 1. Per esempio, nel caso dei testi scritti in lingua inglese, la lettera "e" è la più frequente  $\rightarrow p(e) \approx 0,127$

Il numero medio di blocchi da leggere prima di trovare una "e" è pari a

$$n = \frac{1}{p(e)} = \frac{1}{0,127} \approx 8$$

proj 2  
LOSCHIAU  
Marco

Nel caso di blocchi di dimensione  $m$ , la probabilità di un blocco tipico  $\sigma$  è pari a

$$P(\sigma) \approx 2^{-mH(p)}$$

dove  $m = \dim. \text{blocco}$ ;  $H(p)$  entropia della distribuzione  $p$

Dato una distribuzione di prob.  $p$ , l'entropia è data da:

$$H(p) = - \sum_{a \in A} p(a) \log(p(a))$$

Più gli elementi della distribuzione sono equiprobabili, più il valore dell'entropia è alto.

Nel caso delle lingue naturali, la distribuzione delle singole lettere è

a	8,16%
b	1,28%
c	12,70%
d	0,07%

L'entropia è pari a

$$H(p) = 4,17$$

Pertanto si ottiene

$$P(\sigma) \approx 2^{-mH(p)} \Rightarrow n = \frac{1}{P(\sigma)} = 2^{mH(p)} = 2^{13 \cdot 4,17} \approx 2^{54}$$

Se le lettere fossero tutte equiprobabili, quindi  $\forall a \in A$ ,  $p(a) = 1/26$ , il valore dell'entropia sarebbe pari a

$$\log |A| = \log(26) \approx 4,7004$$

[d] Nel caso di blocchi  $\sigma$  generici (anche non tipici) di dimensione  $m$ :

$$\forall \sigma \in A^m \Rightarrow P(\sigma) = 2^{-m(H(q) + D(q||p))}$$

dove

- $q(a) \triangleq \frac{f(a)}{m}$  // nei blocchi  $\sigma$  tipici si ha  $q(a) \approx p(a)$

- $D(q||p) \triangleq \sum_{a \in A} q(a) \log(q(a)/p(a))$  e' detta DIVERGENZA KL, e

rappresenta la diversità delle distribuzioni  $p$  e  $q$ . Se  $p$  e  $q$  sono

molto simili, la divergenza è molto bassa, fino a diventare nulla nel caso in cui  $p$  e  $q$  sono uguali.

- Dimostrazione

la probabilità di un blocco  $\sigma$  generico è data da:

$$p(\sigma) = \prod_{\omega \in A} p(\omega)^{f(\omega)}$$

Ricorrendo alle proprietà dei logaritmi, possiamo trasformare la produttoria in sommatoria. Si ottiene così

$$\log(p(\sigma)) = \sum_{\omega \in A} \log(p(\omega)^{f(\omega)})$$

Per definizione  $q(\omega) \triangleq \frac{f(\omega)}{m} \Rightarrow f(\omega) = mq(\omega)$  e dunque

$$\begin{aligned} \log(p(\sigma)) &= \sum_{\omega \in A} \log(p(\omega)^{mq(\omega)}) = \sum_{\omega \in A} mq(\omega) \log(p(\omega)) = \\ &= m \sum_{\omega \in A} q(\omega) \log\left(\frac{p(\omega)}{q(\omega)} q(\omega)\right) = \\ &= m \sum_{\omega \in A} q(\omega) \left( \log\left(\frac{p(\omega)}{q(\omega)}\right) + \log(q(\omega)) \right) = \\ &= m \sum_{\omega \in A} q(\omega) \log\left(\frac{p(\omega)}{q(\omega)}\right) + q(\omega) \log(q(\omega)) \\ &\quad \downarrow \quad \downarrow \\ &- D(q||p) \quad - H(q) \end{aligned}$$

Pertanto si ottiene

$$p(\sigma) = 2^{-m(H(q) + D(q||p))}$$

### [e] Scacchis di Eddington

- La scacchis batte i tasti a caso

■ Distribuzione  $q$  dell'opera di Shakespeare: lingua inglese  $\Rightarrow H(q) = 4,77$

■ Distribuzione  $p$  della scacchis:  $\forall \omega \in \mathbb{Z}_{26}, p(\omega) = \frac{1}{26} \Rightarrow H(p) = \log|A| = 4,70$

Si ha una divergenza nulla tra  $q$  e  $p$ , pari a

$$D(q||p) = \sum_{\omega \in A} q(\omega) \log\left(\frac{q(\omega)}{p(\omega)}\right)$$

Possiamo prendere l'opera di Shakespeare come un blocco

pag 3 WOSCHIAU  
MARCO

tipico di bughetta  $m = 10^6$

$$q(\omega) \approx p(\omega) \Rightarrow D(q||p) = \sum_{\omega \in A} p(\omega) \log \left( \frac{p(\omega)}{1/26} \right) = 0,364$$

Sì ottiene

$$n = \frac{1}{p(\omega)} \approx 2^{m(H(q) + D(q||p))} = 2^{10^6 (4,70 + 0,364)} = 7,91 \cdot 10^{1524415}$$

• La scimmia è ammirestata:

■ In questo caso si ha uno sferefante trascurabile:

$$n = \frac{1}{p(\omega)} \approx 2^{mH(p)} = 2^{10^6 \cdot 4,71} = 1,9 \cdot 10^{1417851}$$

## Esercizio 3.1 - Analisi delle Frequenze -

COSCHI AND  
MARC Paj 4

Lo script utilizzato è `text-frequency-analysis.py` ed è stato utilizzato come testo il 1° capitolo di *Moby Dick*.

### 1. Iстogramma delle frequenze

- Una volta letto il testo dal file `.txt`, si applica la funzione `adjust-text(text)` la quale va ad adattare il testo convertendo tutto in maiuscolo, togliendo gli spazi ed i caratteri speciali.
- Il testo adattato viene passato alla funzione `frequency-analysis(text, m, p)` la quale calcola le occorrenze di ciascuna lettera (se  $m=1$ ). La funzione ritorna un dizionario con `keys = lettere`, `values = occorrenze`. Se  $p = \text{'plot'}$  stampa l'istogramma.

### 2. Distribuzione

- Si utilizza la `m-grams-distributionas(text, m, p)`. All'interno utilizza la `frequency-analysis(text, m, p)` in modo da ottenere le frequenze delle lettere. Se  $p = \text{'plot'}$  stampa l'istogramma.

### 3. Indice di lucidezza

- Si utilizza la funzione ~~index~~ `indexOfCoincidence(text, m)`. Nella puh viene calcolata la formula per l'indice di coincidenza. Per ottenere la frequenza del blocco delle  $m$ -lettere si usa `frequency-analysis(text, m, p)`.

### 4. Entropia

- Si utilizza la funzione `entropy(text, m)` la puh calcola l'entropia mediante la formula. Per ottenere la frequenza del blocco delle  $m$ -lettere si richiede la `frequency-analysis(text, m, p)`.

## Esercizio 3.2 Cifrario di Hill

### • CIFRATURA

1. leggo il testo .txt che sarà il mio plaintext. Tramite la `adjustText(text)` si va a modificare il testo, togliendoci spazi, caratteri speciali ecc..
2. Genero una chiave tramite la funzione `getKey(m)`, la quale ritornerà una matrice  $(m, m)$ . Nel mio caso ho scelto  $m=4$ , ~~mentre per generare le due~~ le due viene generata in modo da essere una matrice invertibile.
3. Chiamo la funzione `encryption(plaintext, K)` la quale:
  - converte il plaintext da stringa a lista di numeri tramite la funzione `stringToNumber(plaintext)`
  - spezza il plaintext in blocchi di dimensione  $m$  tramite la `get-N-blocks(plaintext, m)`
  - Si calcola il ciphertext tramite la formula  $C = K \cdot P \pmod{26}$
  - ritorna la stringa di ciphertext

### • DECFRATURA

1. Si chiama la `decryption(ciphertext, K)`
2. Applica il punto 3 della CIFRATURA, sostituendo al posto del plaintext il ciphertext e utilizzando la formula  $P = K^{-1} \cdot C \pmod{26}$ 

✓  
Si calcola  
prima l'inverso  
 $\pmod{26}$  di  $P$

### • ATTACCO KNOW-PLAINTEXT

1. Si chiama la funzione `attack(plaintext, ciphertext, m)` la quale ritorna la chiave
2. Si calcolano le matrici  $P^*$  e  $C^*$  di dimensione  $m \times m$
3. La chiave  $K$  è data da  $K = C^* \cdot \underbrace{P^{*-1}}_{\text{qui si calcola l'inver-} \atop \text{so mod 26 di } P}$ 

$\Rightarrow$  qui si calcola l'inver-  
so mod 26 di  $P$

## Esercizio 2.3 Un criptogramma di Vigenere

Pag 5 LOSCHIAU MARCO

- Si utilizza lo script vigenere-attack.py

Per prima cosa andiamo a determinare la lunghezza delle chiavi con il test di Kasiski utilizzando la funzione Kasiski(ciphertext).

Con la funzione si vuole stabilire quale è la lunghezza delle chiavi del ciphertext e si va a cercare parole con frequenza massima. Una volta trovate si calcola la distanza tra di esse, e si fa il MCD delle distanze. Nel nostro caso  $\text{MCD}(\text{di}) = 8$

A questo punto creiamo delle sottostringhe di dimensione ~~len(text)~~  $\frac{\text{len}(\text{text})}{m}$  nelle quali sono prese le lettere 8 a 8.

Questo viene fatto tramite la funzione getSubstrings(ciphertext, key\_len)

Trovata la funzione findKey(substrings) andiamo a ricavare la chiave lettera per lettera.

~~def findKey(substrings):~~

Viene chiamata la funzione findKeyElement(substring) per ogni sottostringa (in totale saranno 8).

Sapponendo che una substring sia il risultato di una shift encryption di parametro  $K$  si cerca il valore  $K$  che massimizza il prodotto scalare tra il vettore frequenze della stringa risultato di una shift encryption  $K$  e il vettore di frequenze delle lingue inglese.

Ritorna la chiave

In fine traccia la chiave trovata al punto precedente sare' possibile decifrare il ciphertext con la funzione  
decipher(~~ciphertext~~, key)