

Laboratorio di Teoria dell'Informazione

Codici rateless

Obiettivo: realizzazione di encoder e decoder per codici rateless e valutazione dell'overhead di codifica al variare della dimensione del blocco.

Parte 1. Encoder

Si scriva una funzione che, presi in ingresso k interi (32 bit senza segno), produca un numero arbitrario n di codifiche di tipo rateless. Ogni codifica è rappresentata dalla combinazione lineare di d interi scelti casualmente. Il grado d deve essere generato con un'opportuna distribuzione di probabilità. L'output della funzione sarà una lista di pacchetti codificati (ciascun pacchetto codificato è costituito da 32 bit senza segno) accompagnati dagli indici dei d simboli originali utilizzati per comporre ciascuna codifica.

Esempio di encoder.

1. Legge k interi $x[i]$, $i=0, \dots, k-1$
2. Produce n pacchetti + informazione di controllo come segue:
 - genera in modo casuale il grado di codifica d utilizzando una distribuzione di probabilità
 - sceglie d indici compresi tra 0 e $k-1$ (senza ripetizioni)
 - crea il pacchetto codificato $y[j]$, $j=0, \dots, n-1$ come XOR fra d $x[i]$ scelti al passo precedente

Al termine dell' operazione di codifica devono essere stati prodotti n pacchetti codificati accompagnati da altrettante liste degli indici usati per comporli (colonne della matrice generatrice del codice).

Per la realizzazione della funzione che genera il grado d viene fornito il codice C che estrae interi secondo la distribuzione RSD. Si veda il codice sorgente nell'archivio `rsd.tar.gz` contenente i seguenti file:

- `random.cpp`, `random.h` (generatore casuale per la rsd)
- `test.cpp` (esempio di uso delle funzioni `rsd`, `./test -h` per help)
- `Makefile` (make all per compilare il test)

In alternativa viene fornito uno script matlab per la generazione della medesima distribuzione

- `rsd.m`

Si implementi inoltre qualche altra distribuzione nota, ad esempio uniforme fra due interi o esponenziale.

Parte 2. Decoder

Si scriva una funzione che, dati n simboli codificati e la lista degli indici associati a ciascuno di essi, ricostruisca il maggior numero dei simboli originali $x[i]$. Si utilizzi la procedura di decodifica nota come *Belief Propagation* che, nel caso dei codici rateless, coincide con la soluzione del sistema lineare associato attraverso sostituzione all'indietro a partire dalle equazioni di grado $d=1$.

Esempio decoder

1. Ricerca un simbolo codificato $y[j]$ di grado $d=1$ (ovvero la lista degli indici contiene il riferimento a un solo simbolo originale $x[i]$). Se non esistono simboli di grado 1 la decodifica TERMINA senza successo.
2. Assegna $x[i]=y[j]$ ed elimina il simbolo $y[j]$ dalla lista dei pacchetti da decodificare
3. Ricerca tra i pacchetti codificati tutti quelli che contengono $x[i]$ nella lista dei simboli combinati con xor. Per ognuno dei pacchetti $y[k]$ trovati:
 - $y[k]=y[k] \text{ XOR } x[i]$
 - cancella indice i dalla lista dei simboli combinati corrispondente al pacchetto k
1. Se tutti gli $x[i]$ sono stati decodificati TERMINA, in caso contrario torna a 1.

Parte 3. Esperimenti.

1. Impostare la RSD (con parametri $c=0.05$ e $\delta=0.05$) come distribuzione del grado. Per i valori di $k=50, 100, 200, 500$ verificare se e' possibile portare a successo la decodifica con $n=2k$
2. Per alcuni valori di k valutare l'overhead medio di codifica procedendo come segue.
 - Per valori di $n=k, 1.1k, 1.2k \dots 2k$
 - Effettuare T codifiche e valutare la probabilità di decodificare con successo ($p(n)=\text{numero decodifiche con successo}/T$)Osservare l'andamento di $p(n)$ in funzione di n e fare dei confronti al variare di k .
3. Ripetere l'esperimento 1 utilizzando al posto della distribuzione RSD un meccanismo completamente casuale per costruire le equazioni di codifica (lanciare una moneta per stabilire la se ogni simbolo originale deve essere sommato o meno). Si individui il motivo per cui il decoder LT non riesce a determinare la soluzione del sistema lineare.