Elaborazione di Segnali Multimediali
a.a. 2017/2018

La quantizzazione

L.Verdoliva

L'argomento di questa esercitazione è la quantizzazione uniforme di immagini, in particolare vedremo che effetto ha sulle immagini questa operazione e valuteremo le prestazioni in termini di curve tasso-distorsione. Quindi faremo diversi esperimenti con la quantizzazione predittiva e confronteremo diverse strategie di codifica.

1 Quantizzazione uniforme

In questa esercitazione focalizzeremo l'attenzione alla classe dei quantizzatori uniformi, in cui il passo di quantizzazione Δ è costante. L'operazione che deve realizzare tale quantizzatore non è altro che un'approssimazione dei valori in ingresso con i livelli di restituzione definiti dal quantizzatore, che, essendo uniforme, sono rappresentati proprio dai valori centrali dell'intervallo.

Concettualmente possiamo distinguere la quantizzazione in due fasi: determinazione del codice (livelli di restituzione del quantizzatore) e codifica vera e propria (i dati sono approssimati con i livelli determinati). Il primo passo è molto semplice per un quantizzatore uniforme nell'ipotesi in cui N sia il numero di livelli di restituzione si possono determinare facilmente nel seguente modo:

```
xmax = max(x(:));  
xmin = min(x(:));  
Delta = (xmax - xmin)/N;  
y = [xmin+Delta/2:Delta:xmax]; % livelli di restituzione
```

Noti i livelli di restituzione nella fase di quantizzazione vera e propria bisogna prima individuare in quale intervallo di quantizzazione cade ogni valore, e poi selezionare il corrispondente livello di quantizzazione. Un modo possibile per implementare questa operazione è quello di realizzare un test logico per stabilire quali valori cadono all'interno di ogni intervallo di quantizzazione, così come mostrato di seguito:

```
t = [min(x(:))-1 (y(1:N-1)+y(2:N))/2 max(x(:))+1]; % soglie di decisione  
xq = zeros(size(x));  
for i = 1:N,  
    xq = xq+((t(i)<=x)&(x<t(i+1)))*y(i);  
end;
```

La matrice $((t(i)<=x)&(x<t(i+1)))$ vale 1 nelle posizioni tali che $x(m,n)$ è compreso nell' i -esimo intervallo, 0 nelle altre posizioni. Moltiplicando per il livello di restituzione dell'intervallo e sommando il risultato a xq per ognuna delle celle, ci si assicura che, alla fine del ciclo, xq contenga proprio i valori quantizzati dell'im-

immagine. Notate come il codice che realizza la quantizzazione sia generale, nel senso che una volta assegnati i livelli può realizzare anche una quantizzazione non uniforme. Trasformate questo script in una funzione dal prototipo: `function xq = quant_uniform(x,R)` che quantizza i dati uniformemente su $N = 2^R$ livelli di restituzione.

Un modo molto semplice per realizzare la quantizzazione uniforme è attraverso l'uso della funzione `round` che effettua l'arrotondamento di valori reali:

```
y = Delta.*(round(x./Delta));
```

Confrontate questa soluzione con quella vista in precedenza.

2 Quantizzazione predittiva

Le prestazioni dello schema di codifica possono essere notevolmente migliorate se si fa uso della predizione nel processo di quantizzazione. Provate allora a scrivere il codice che esegue i seguenti passi:

1. Visualizzazione dell'immagine originale x con relativo istogramma (che può essere approssimato da una distribuzione uniforme);
2. Determinazione dell'immagine $\tilde{x}$ che contiene i valori di predizione (si assuma, per semplicità, che il valore predetto sia pari al pixel precedente lungo le righe);
3. calcolo della differenza tra l'immagine originale e quella predetta $y = x - \tilde{x}$ e sua visualizzazione con relativo istogramma (che può essere approssimato da una distribuzione Laplace);
4. quantizzazione dell'immagine delle differenze y mediante la funzione `quant_laplace` (disponibile sul sito) che produce $\hat{y}$;
5. ricostruzione dell'immagine quantizzata $\hat{x}$.

Realizziamo i passi delineati facendo riferimento alla figura 9 (appunti sulla compressione), cioè generiamo l'immagine delle differenze, quindi quantizziamo l'errore e poi ricostruiamo il segnale supponendo di conoscere perfettamente i valori predetti:

```
% Codifica
[Nr Nc] = size(x);
e = x(:,2:Nc)-x(:,1:Nc-1);
eq = quant_laplace(e,R);
eq = [x(:,1) eq];
% Decodifica
xr = eq(:,2:Nc) + x(:,1:Nc-1);
```

In realtà questa soluzione prevede la conoscenza in fase di decodifica dei dati originali, ciò che si conosce in realtà sono le stime prodotte dai valori quantizzati (figura 12, appunti). Modificate allora la fase di decodifica in tal senso e poi allineate il codificatore al decodificatore, in modo che anche in trasmissione la predizione sia realizzata a partire dai valori quantizzati.

Confrontate questa strategia in termini di PSNR con quella che realizza la quantizzazione uniforme sui dati originali, a parità di risorse utilizzate.

2.1 Esercizi proposti

1. *Codifica e Decodifica.* E' opportuno osservare che, se il quantizzatore deve essere inserito all'interno di un algoritmo di codifica, bisogna fornire in uscita non tanto x_q quanto gli indici corrispondenti ai livelli, quindi modificate il codice in modo che il prototipo sia: `function I=cod(x,R)`, dove I è la matrice degli indici. Scrivete poi un'altra funzione: `function xq=decod(I,R,xmin,xmax)` che restituisce l'immagine quantizzata.
2. *Curve tasso-distorsione.* Per valutare le prestazioni di un algoritmo di compressione lossy è utile tracciare la curva tasso-distorsione, in cui al variare del *tasso* di codifica $R = 1, 2, \dots, 7$ si considera la *distorsione* corrispondente, per esempio il rapporto segnale-rumore. Determinate tali curve per il quantizzatore uniforme su alcune delle immagini di prova (N.B. le immagini .y sono 512×512 rappresentate su 8 bit).
3. *Confronto strategie di predizione.* Nell'immagine Rice.tif si vuole predire il generico pixel $x(m, n)$ a partire da quelli lungo le righe e colonne precedenti, e si provano diverse strategie di predizione. In tutte le strategie, il pixel in alto a sinistra è assegnato, $\hat{x}(1, 1) = x(1, 1)$, quelli sulla prima riga sono predetti a partire da quelli che li precedono sulla riga stessa $\hat{x}(1, n) = x(1, n - 1)$, e quelli sulla prima colonna dai precedenti sulla colonna. Per gli altri pixel le strategie sono le seguenti:
 - (a) predizione a partire dal pixel precedente lungo la riga;
 - (b) predizione a partire dal pixel precedente lungo la colonna;
 - (c) predizione pari alla media dei pixel precedenti lungo riga, colonna, e diagonale;
 - (d) predizione lungo la riga se $\Delta_R < \Delta_C$ e lungo la colonna altrimenti, dove $\Delta_R = x(m - 1, n) - x(m - 1, n - 1)$ e $\Delta_C = x(m, n - 1) - x(m - 1, n - 1)$;

Confrontare le diverse strategie calcolando e visualizzando l'errore di predizione $y(m, n) = \hat{x}(m, n) - x(m, n)$ e valutandone il valore quadratico medio.