Esame del corso di
Tecniche Avanzate per il Trattamento delle Immagini

Data: 2 Febbraio 2009

Es.1.[5 Pt] Considerando una soglia di visibilità dell'1%, si determini il numero massimo di livelli di grigio distinguibili per un dispositivo di visualizzazione basato su un tubo a raggi catodici con contrasto 70:1

Es.2. [2pt] Indicare,utilizzando la notazione matematica adeguata, cosa si intende per "filtrare" un'immagine a toni di grigio con la seguente matrice:

0	-1	0
-1	5	-1
0	-1	0

[1 pt]A quale operazione matematica può essere assimilata?
[1 pt]Quale sarà l'effetto sull'immagine finale?
[2 pt]Indicare qualitativamente l'effetto che si produce sulla trasformata di Fourier Bidimensionale dell'immagine iniziale.

Es.3. Estraendo alcune feature dalle immagini di 5 oggetti si ottengono 5 vettori (uno per ciascun oggetto) in uno spazio a 3 dimensioni R^3.

$x_1 = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}, x_2 = \begin{bmatrix} -1 \\ 4 \\ 5 \end{bmatrix}, x_3 = \begin{bmatrix} 6 \\ 7 \\ 8 \end{bmatrix}, x_4 = \begin{bmatrix} 3 \\ 2 \\ 2 \end{bmatrix}, x_5 = \begin{bmatrix} -2 \\ -3 \\ 4 \end{bmatrix}$ effettuando l'analisi delle componenti principali (PCA) si ottengono i seguenti auto valori 91, 17 e 11 e gli autovettori associati sono rispettivamente:

$v_1 = \begin{bmatrix} -0.6 \\ -0.7 \\ -0.3 \end{bmatrix}, v_2 = \begin{bmatrix} 0.5 \\ 0 \\ -0.9 \end{bmatrix}, v_3 = \begin{bmatrix} 0.6 \\ -0.7 \\ 0.3 \end{bmatrix}$ (N.B. Tali valori sono approssimati alla prima cifra decimale)

[2 Pt]Indicare l'errore quadratico medio (RMS) che si compie trascurando le componenti dovute al 3° autovalore.
[3Pt] Determinare le 3 componenti del primo vettore x_1 nel nuovo spazio.
[1Pt] Indicare di quali proprietà godono gli autovettori.
[2Pt] Cosa sarebbe accaduto se un vettore fosse stato linearmente dipendente dagli altri? E se addirittura 3 vettori fossero stati linearmente dipendenti dagli altri due?

Es.4 [**6 Pt**] Si descrivano due procedure per ridurre la dimensionalità di dati nello spazio delle feature da 2D a 1D. Una delle due procedure da descrivere permetta di operare una riduzione della dimensionlità delle feature in modo non supervisionato; la seconda procedura, invece, permetta di operare una riduzione della dimensionalità delle feature in modo supervisionato (con conoscenza a priori delle classi di appartenenza dei dati). Siano date in ingresso N=2 classi di punti (g_p_1 e g_p_2)generati in modo random con distribuzione gaussiana (stessa varianza e diversa media):

```
g_p_1=[0.2*randn(100,2)+repmat([0.5 0],100,1); 0.2*randn(100,2)+repmat([0 0.5],100,1)];
g_p_2=[0.2*randn(100,2)+repmat([-0.5 0],100,1); 0.2*randn(100,2)+repmat([0 -0.5],100,1)];
```

Si implementino i passi necessari mediante codice Matlab e si descrivano le differenze nel codice tra le due procedure.

Es.5 [**6 Pt**] Data un'immagine truecolor memorizzata in un file immagine.bmp ed avente i dati rappresentati ad 8bit per ogni piano di colore si implementino i seguenti punti mediante codice Matlab:

Leggere, caricare nel workspace e visualizzare l'immagine.

Convertire l'immagine di partenza (truecolor) ad una rappresentazione a 256 toni di grigio e, successivamente, da una rappresentazione a 256 toni di grigio ad una a soli due valori (binaria) mediante un'operazione che riduca gli effetti "false contours" e "color flatness".

Applicare all'immagine con dati rappresentati a 1 bit l'operazione di erosione con elemento strutturante quadrato (3x3 pixel).

Si converta l'immagine a 256 toni di grigio ad una rappresentazione a valori decimali a valori compresi tra 0 e 1, poi si applichi del rumore additivo gaussiano con media nulla e deviazione standard 0.2.

Sottocampionare l'immagine a 256 toni di grigio di un fattore 2, e poi sovracampionarla di un fattore 4 applicando un'interpolazione bilineare.

Si ruoti l'immagine ottenuta al punto precedente di un angolo di 25 gradi mantenendo costanti le dimensioni dell'immagine in output rispetto all'immagine in input. Come metodo di interpolazione dei valori dei pixel risultanti, si usi una procedura che assegni al valore di un certo pixel nell'immagine in output il valore del pixel più vicino nell'immagine in input.

Elenco nomi funzioni Matlab:

```
diag
dither
eig
im2double
imerode
imread
imresize
imrotate
imshow
mean
randn
repmat
rgb2gray
size
sqrt
strel
```

Soluzioni

1. Seguendo la legge di Weber: $\frac{70}{1} = (1+0.01)^n \rightarrow n = \log_{1.01} 70 = \frac{\log 70}{\log 1.01} = 427$
2. L'operazione effettua un'enfasi sull'immagine basata sul Laplaciano. Il peso dato a tale operatore è uguale al peso dell'immagine iniziale:
$$1 \cdot \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} + 1 \cdot \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$
L'effetto sulle frequenze è quello di enfasi delle componenti in alta frequenza con un andamento quadratico (la derivata seconda di una funzione si traduce nella moltiplicazione per una funzione quadratica nelle frequenze.)
3. L'errore quadratico medio è pari al terzo autovalore (11).
Le componenti della proiezione del primo vettore sulla base definita dagli autovettori si ottiene effettuando il prodotto scalare tra il vettore ed i 3 autovettori.
Per quanto riguarda le proprietà degli autovettori consultare il testo.
Se un vettore fosse stato linearmente dipendente dagli altri non sarebbe cambiato nulla nella determinazione degli autovalori/autovettori, se invece 3 vettori fossero risultati linearmente dipendenti dagli altri avrebbe voluto dire che i cinque punti appartenevano ad uno stesso piano e quindi un autovalore sarebbe risultato nullo.
4. Le procedure richieste corrispondono alla Principal Component Analysis (PCA) e alla Linear Discriminant Analysis (LDA).
Con l'algoritmo PCA si calcola la trasformazione da applicare per ridurre la dimensionalità delle feature in modo non supervisionato (no conoscenza a priori delle classi di appartenenza dei dati). Invece con l'algoritmo LDA si calcola la trasformazione da applicare per ridurre la dimensionalità delle feature in modo supervisionato (con conoscenza a priori delle classi di appartenenza dei dati).

   Codice Matlab dell'algoritmo PCA:
   - Si calcola la matrice di covarianza della matrice dei dati complessivi

```
g_p = [g_p_1; g_p_2]';
g_p_cent = mean(g_p, 2);
g_p_zm = g_p - repmat(gn_p_cent, 1, size(g_p,2));
Cov = g_p_zm*(g_p_zm');
```

   - Si calcola la matrice di trasformazione composta dall'autovettore associato all'autovalore maggiore della matrice di covarianza `Cov`

```
[V_pca D_pca] = eig(Cov);
P_pca = diag(1./sqrt(D_pca(end,end)))*V_pca(:,end)';
```

   - Si trasformano i dati delle due classi

```
g_r_1_pca = P_pca*(g_p_1 -repmat(g_p_cent,1,size(g_p_1,2)));
g_r_2_pca = P_pca*(g_p_2 -repmat(g_p_cent,1,size(g_p_2,2)));
```

   Codice Matlab dell'algoritmo LDA:
   - Dati i dati in input, si calcolano i centroidi e le matrici Sw e Sb (con prob. a priori classi 0.5)

```
mu_1 = mean(g_p_1, 1);
mu_2 = mean(g_p_2, 1);
mu = 0.5*mu_1 + 0.5*mu_2;
Cov_1 =
 (g_p_1-repmat(mu_1,size(g_p_1,1),1))'                                *(g_p_1-
  repmat(mu_1,size(g_p_1,1),1));
Cov_2 =
```

```
 (g_p_2-repmat(mu_2,size(g_p_2,1),1))'                                *(g_p_2-
  repmat(mu_2,size(g_p_2,1),1));
S_w = 0.5*Cov_1 + 0.5*Cov_2;
S_b = (mu_1 - mu)'*(mu_1 - mu) + (mu_2 - mu)'*(mu_2 - mu);
```

- Si calcola C, i suoi autovettori ed autovalori

```
C = S_b/S_w;
[V_lda D_lda] = eig(C);
```

- Si calcola la matrice di trasformazione composta dall'autovettore associato all'autovalore maggiore

```
P_lda = diag(1./sqrt(D_lda(end,end)))*V_lda(:,end)';
```

- Si procede alla trasformazione dei dati delle singole classi

```
g_r_1_lda = P_lda*(g_p_1 - repmat(mu', 1, size(g_p_1,2)));
g_r_2_lda = P_lda*(g_p_2 - repmat(mu', 1, size(g_p_2,2)));
```

5. Eserzio 5:

```
img_rgb = imread('immagine.bmp');
figure;
imshow(img_rgb);
```

(oppure:

```
figure;
      imshow('immagine.bmp');
      img_rgb = getimage;)
```

-
```
img_gray = rgb2gray(img_rgb);
img_bin = dither(img_gray);
```

-
```
B = strel('square', 3);
img_bin_er_B = imerode(img_bin,B);
```

-
```
img_gray_double = im2double(img_gray);
f_n = 0.2*randn(size(img_gray_double));
f_O = img_gray_double + f_n;
f_O(f_O > 1) = 1;
f_O(f_O < 0) = 0;
```

-
```
img_gray_reduced = imresize(img_gray, 0.5);
img_gray_enlarged = imresize(img_gray_reduced, 4, 'bilinear');
```

-
```
img_gray_rotated = imrotate(img_gray_enlarged, 25, 'crop');
```