Laboratorio Sperimentale di Matematica Computazionale (Parte I) Lezione 3

Gianna Del Corso <gianna.delcorso@unipi.it>

8 Marzo 2017

1 Classificazione di cifre scritte a mano

In questa lezione prendiamo in considerazione un particolare problema di classificazione, quello del riconoscimento automatico di caratteri numerici scritti a mano. Analizzeremo con un approccio tipico del "Machine learning", due metodi. Il primo è l'approccio banale che classifica i caratteri sulla base della distanza da un "rappresentante" medio. L'altro è' basato sulle proprietà della SVD di concentrare le informazioni più caratterizzanti un insieme di dati nei primi vettori e valori singolari di un'opportuna matrice.

Nell'approccio machine learning alla classificazione, viene generalmente fornito un insieme di dati (che costituisce il cosı detto Training Set) classificati in modo manuale. È su questi dati che l'algoritmo dovrà "apprendere" le caratteristiche peculiari delle varie categorie. La correttezza dell'algoritmo viene poi testata eseguendo questo stesso metodo sul Test Set per classificare i nuovi oggetti. Nello specifico delle cifre scritte a mano, supporremo di avere un Training set di immagini che sono state classificate a mano ed un Test Set di immagini che vogliamo classificare in base alla loro "somiglianza" appresa dai dati contenuti nel Training Set.

1.1 Descrizione del data set

I dati sono contenuti nel file `dataset_digits` che può essere letto con il comando `load('dataset_digits')`. Vengono caricate 4 matrici

- La matrice `Training` ha dimensione 256×1707. Ogni colonna di `Training` contiene l'immagine di una cifra numerica di 256 pixel messa per colonna. Con il comando `ima2(Training(:, j)` si visualizza l'immagine 16×16 della cifra memorizzata nella colonna j di `Training`.

- Il vettore DigitsTraining ha dimensione 1707 e contiene l'associazione tra le immagini di `Training` e le cifre che queste rappresentano. In particolare se `DigitsTraining(j)=k` significa che `Training(:, j)` rappresenta la cifra k.
- La matrice `TestSet` contiene 2007 immagini da classificare, ogniuna di dimensione 16×16. Come per le immagini del Training Set ogni immagine è rappresentata come un vettore con 256 componenti e memorizzata in una colonna di `TestSet`.
- Il vettore `DigitsTest` ha 2007 componenti e può essere usato per valutare la bontà del nostro classificatore perché contiene la classificazione manuale anche delle immagini dell'insieme test.

Il file `ima2(v)` permette di visualizzare il vettore v, che si assume di 256 elementi, come un'immagine 16×16 con 10 livelli di grigio.

1.2 Un semplice classificatore di cifre numeriche scritte a mano

Si scriva una funzione

```
function [class]=mean_classifier(TrainingSet, DigitsTraining, TestSet)
% TrainingSet: una matrice mxn contente m immagini vettorizzate su n componenti
% DigitsTraining: vettore di n componenti che riporta la cifra rappresentata
% di ogni immagine in TraininSet
% TestSet: matrice di immagini da classificare
% class: vettore che conterra' la classificazione delle cifre nel TestSet
```

La classificazione deve avvenire secondo il seguente schema:

- Per ognuna delle 10 cifre $c = 0, \ldots, 9$ si calcoli l'immagine media m_c tra le immagini di `TrainingSet`. Si utilizzi il vettore `DigitsTraining` per capire a quale cifra è associata ogni colonna di `TrainingSet`.
- Si memorizzino le immagini medie come colonne di una matrice M di dimensione $m \times 10$ e le si visualizzino utilizzando i comandi `subplot(2, 5, c), ima2(M(:, c))`. Poichè gli indici in Matlab partono dal valore 1, si assuma che $M(:, c)$ contenga l'immagine media relativamente alla cifra $c - 1$.
- Per ogni immagine in `TestSet`, si classifichi questa immagine come "c" se questa immagine test risulta più vicina all'immagine media m_c in norma 2.

Sul dataset messo a disposizione si fornisca la percentuale di cifre catalogate correttamente confrontando i risultati ottenuti nel vettore `class` con quelli forniti in `DigitsTest`.

1.3 Un classificatore basato sulla SVD

Indichiamo con A la matrice con colonne in $\mathbf{R}^m, m = 256$ contenente tutte le immagini nel Training Set *di uno stesso tipo*, ad esempio tutte le immagini della cifra 5. Si può

vedere che in questo caso particolare rank(A) $= 88 \ll 256$, cioè sebbene le immagini possano essere interpretate com vettori di $\mathbf{R}^{256}$, in realtà i vettori che rappresentano la cifra 5 spannano un sottospazio di dimensione molto minore. Questo fa sperare che l'intersezione tra sottospazi relativi a cifre differenti sia piccola. Utlizzando la SVD possiamo riscrivere A come

$$A = \sum_{k=1}^{m} \sigma_k \mathbf{u}_k \mathbf{v}_k^T,$$

quindi ogni colonna di A (quindi ogni immagine di un 5 nel Training Set) si può scrivere come combinazione ortogonale delle colonne di U, cioè

$$\mathbf{a}_i = \sum_{k=1}^{m} (\sigma_k v_{ki}) \mathbf{u}_k,$$

La direzione dei primi $\mathbf{u}_k$ contribuisce maggiormente alla formazione dell'immagine rappresentata dal vettore $\mathbf{a}_i$ poichè i valori singolari sono ordinati in modo non crescente.

Un'immagine test rappresentanta da un vettore $\mathbf{z}$ potrà essere classificata come un 5 se questa si esprime bene come combinazione lineare dei vettori $\mathbf{u}_k$ che generano lo spazio immagine del 5, ovvero se

$$\min_{\alpha} \|\mathbf{z} - \sum_{k=1}^{m} \alpha_i \mathbf{u}_k\|_2$$

è piccolo rispetto al valore che otterremmo usando le matrici dei vettori singolari sinisti corrispodenti agli spazi immagine delle altre cifre.

Si scriva una funzione

```
function [class]=svd_classifier(TrainingSet, DigitsTraining, TestSet, h)
% TrainingSet: una matrice nxm contente m immagini vettorizzate su n componenti
% DigitsTraining: vettore di m componenti che riporta la cifra rappresentata
% di ogni immagine in TraininSet
% TestSet: matrice di immagini da classificare
% h :intero
% class: vettore contente la classificazione delle cifre nel TestSet
```

La classificazione deve avvenire secondo il seguente schema:

- Per ogni cifra c, $c = 0, \ldots, 9$, si costruisca una matrice A_c di dimensione $n \times N_c$ le cui colonne sono le immagini di una stessa cifra c nel training set. N_c è quindi il numero di occorrenze della cifra c nel training set. Quindi $m = \sum_{c=0}^{9} N_c$, numero di colonne di `TrainingSet`.
- Per ognuna delle 10 matrici ottenute A_c se ne calcoli la SVD, con il comando `[U_c,S_c,V_c]=svd(A_c)`.
- Si visualizzino con i comandi `subplot(2, 5, c), ima2(U_c(:, 1));` i primi vettori singolari sinistri. Che immagini si ottengono, perchè?

- Per ogni immagine del Test Set rappresentata dal vettore $\mathbf{z}$ e per ogni cifra c si considerino le prime `h` colonne di ogni matrice U_c, e si risolva il seguente problema

$$\gamma_c = \min_{\alpha} \|\mathbf{z} - U_c(:,k)\,\alpha\|_2,$$

la cui soluzione è data da $\alpha = U_c(:,1:h)^T\mathbf{z}$ ottenendo il residuo

$$\gamma_c = \| \left(I - U_c(:,1:h)U_c(:,1:h)^T\right)\mathbf{z}\|_2.$$

L'immagine rappresentata da $\mathbf{z}$ è classificata come la cifra c se γ_c è il minimo tra i 10 valori ottenuti.

Si usino dei valori di h nell'intervallo $[5, 20]$.

Suggerimento: Si noti che le matrici A_c servono solo per calcolarne la SVD e che si può utilizzare una matrice `BigU` per memorizzare a blocchi di h colonne le matrici $U_c(:,1:h)$

Si analizzino poi i seguenti aspetti specifici.

- Si rappresenti in una tabella o in un grafico la percentuale di immagini test classificate correttamente in funzione del numero dei vettori h messi nella base.
- Si analizzi se tutti i numeri hanno la stessa difficoltà ad essere classificati. Per quelli più difficili da riconoscere si guardi se questo dipende dalla calligrafia o dalla difficoltà intrinseca ad identificare quella cifra.
- Si analizzino i vettori singolari delle differenti classi e li si visualizzino come immagini. Può essere una buona idea utilizzare un numero diverso di vettori singolari per rappresentare diverse classi? Si analizzi specificatamente se per alcune classi si ha beneficio usando un numero ridotto di vettori.