Progetto - Food Classification

Social Media Management 2016/2017

Biondi - Greco - Maggio

INTRODUZIONE
Machine learning & Bag of visual Word

Machine Learning (ML), o Apprendimento Automatico, si occupa di sviluppare algoritmi in grado di riconoscere pattern complessi tramite l'analisi di dati. Questo settore di ricerca ha assunto grande rilievo, grazie all'elevata efficienza che esibisce nella soluzione di problemi complessi come il riconoscimento di immagini. Il valore di queste tecniche è legato alla potenza di calcolo dei moderni computer e alla ricchezza di informazioni disponibili in forma digitale (social network, Internet of Things, eBooks).

Bag of visual word (BOVW) è un metodo utilizzato nell'Information Retrieval per rappresentare le immagini tramite tramite un insieme di feature. In Computer Vision un modello BOVW è un vettore sparso di occorrenze del vocabolario di caratteristiche locali dell'immagine.

Per questo progetto abbiamo deciso di affrontare una tematica di classificazione. Per cui abbiamo scelto di effettuare un'analisi sull'argomento relativo alle immagini del cibo contro quelle del "non-cibo".

Food - Etichetta 0

No-Food - Etichetta 1

python

Acquisizione collezione di immagini

Abbiamo acquisito il dataset di immagini di cibo VS non-cibo già suddiviso nelle rispettive categorie. Il dataset inizialmente presentava circa 13.000 immagini, dunque lo abbiamo ristretto per motivi computazionali a 4000 immagini.

http://iplab.dmi.unict.it/madima2015/

CPU: Intel Core i7-4700HQ CPU @ 3.4GHz
GPU: GeForce 840M
RAM: 16gB

Number of images in the training set: 2800
Number of images in the test set: 1200
Total number of images: 4000
Extract Features: 477.38 sec
Describe Training set: 705.04 sec
Describe Test set: 301.25 sec

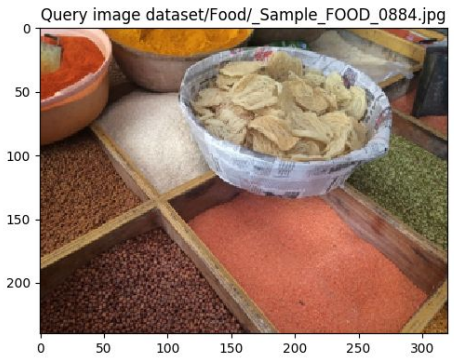
python

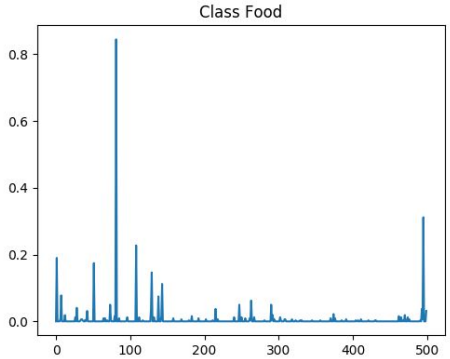
Estrazione delle feature

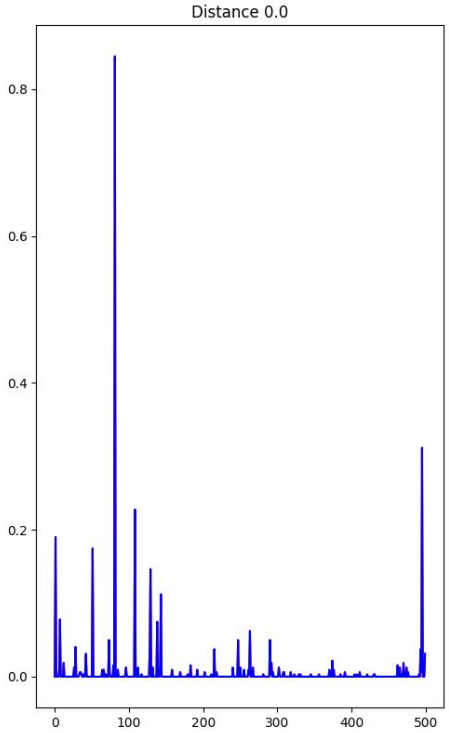
Passi per la costruzione del modello BOVW.

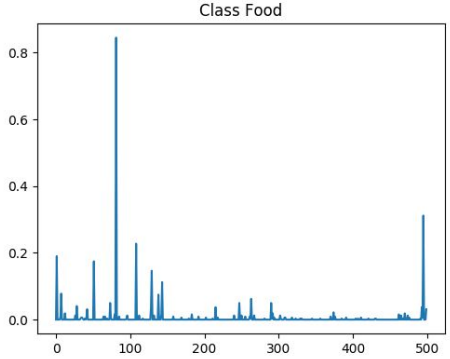
- Abbiamo estratto le feature splittando il dataset al 70 % per training e al 30% per test set.
- Abbiamo utilizzato la funzione DAISY con step pari a 8 per l'estrazione delle feature ed ottenere i descrittori locali.
- Per costruire il nostro dizionario di parole visuali, abbiamo messo insieme tutte le feature estratte dal training set e le abbiamo clusterizzate con l'algoritmo KMeans (scikitlearn) scegliendo come numero di cluster 500.
- Successivamente abbiamo utilizzato KNN per classificare ed estrarre dati come accuracy e confusion matrix.

Rappresentazione delle immagini e classificazione



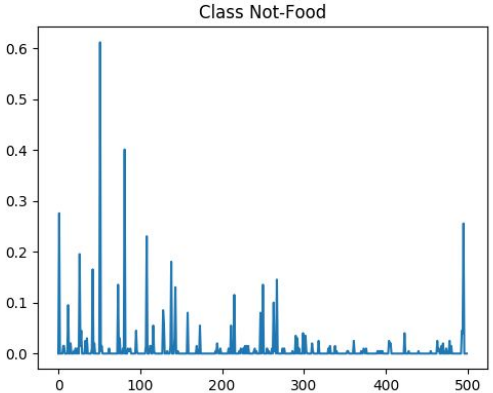


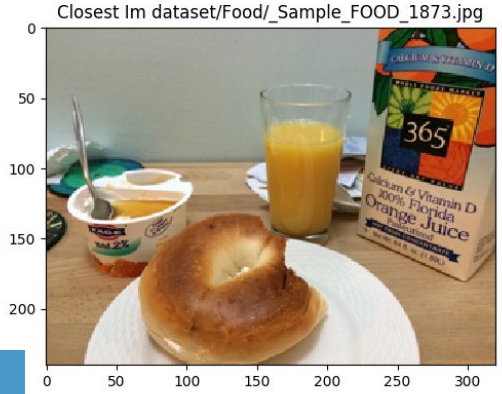


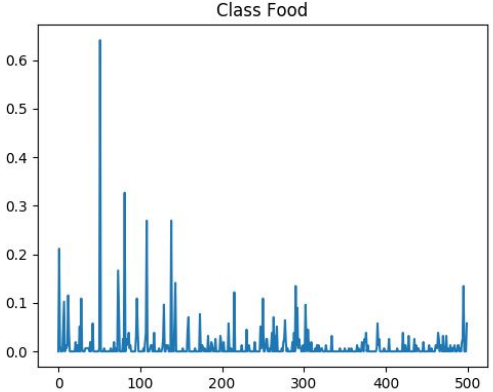


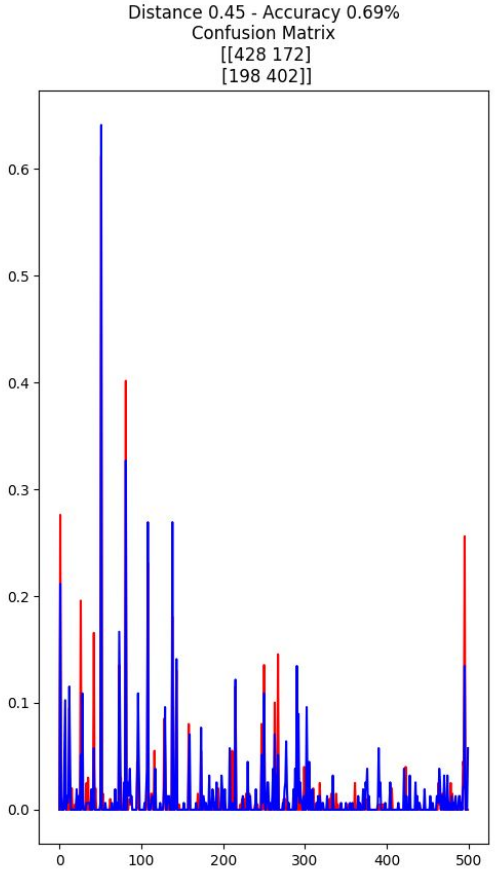
python


Query image dataset/Non-Food/_Sample_NOFOOD_1810_.jpg


Class Not-Food


Closest Im dataset/Food/_Sample_FOOD_1873.jpg


Class Food


Distance 0.45 - Accuracy 0.69%
Confusion Matrix
[[428 172]
[198 402]]

python

Valutazione dei risultati
REPORT

1-NN, accuracy: *0.69%*, Confusion Matrix:

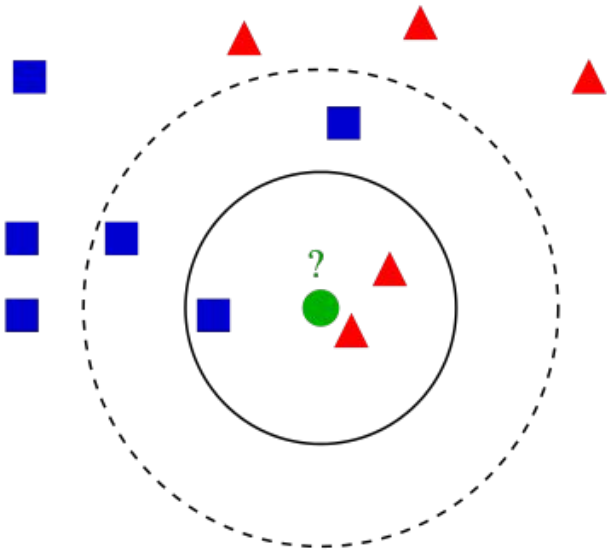
428	172
198	402

5-NN, accuracy: *0.73%*, Confusion Matrix:

447	153
165	435

50-NN, accuracy: *0.74%*, Confusion Matrix:

471	129
187	413



In figura è rappresentato un esempio di classificazione mediante kNN. Il punto sotto osservazione è il pallino verde. Le classi sono due:
Quella dei triangolini rossi;
Quella dei quadratini blu.
Se *k = 3* (cioè vengono considerati i 3 oggetti più vicini), allora il pallino verde viene inserito nella stessa classe dei triangolini rossi perché sono presenti 2 triangolini e 1 quadratino. Se *k = 5* allora viene inserito nella stessa classe dei quadratini blu perché sono presenti 3 quadratini e 2 triangolini.

Matrice di confusione

Nell'ambito dell'intelligenza artificiale la matrice di confusione, detta anche tabella errata classificazione, restituisce una rappresentazione dell'accuratezza di classificazione statistica.
Ogni colonna della matrice rappresenta i valori predetti, mentre ogni riga rappresenta i valori reali. L'elemento sulla riga i e sulle colonna j è il numero di casi in cui il classificatore ha classificato la classe "vera" j come classe i.

Nell'esempio si può notare che dei 7 gatti reali, il sistema ne ha classificati 2 come cani. Allo stesso modo si può notare come dei 12 conigli veri, solamente 1 è stato classificato erroneamente. Gli oggetti che sono stati classificati correttamente sono indicati sulla diagonale della matrice, per questo è immediato osservare dalla matrice se il classificatore ha commesso o no degli errori.

Esempio di matrice di confusione

		Predetti			***Somma***
		Gatto	**Cane**	**Coniglio**	
Reali	**Gatto**	5	2	0	7
	Cane	3	3	2	8
	Coniglio	0	1	11	12
Somma		8	6	13	27

Accuratezza

Nella teoria degli errori, l'accuratezza è il grado di corrispondenza del dato teorico, desumibile da una serie di lavori misurati, con il dato reali o di riferimento, ovvero la differenza tra valor medio campionario e valore vero o di riferimento. Indica la vicinanza del valore trovato a quello reale.
Il più semplice metodo di per valutare le performance di un algoritmo di classificazione, consiste nel contare quanti valori predetti y_i coincidono con i relativi valori di ground truth y_i. La frazione (o percentuale) dei valori correttamente predetta viene detta "accuracy" e può essere calcolata come segue

$$a = \frac{\sum_{i=1}^{N} \mathbf{1}(y_i = \hat{y}_i)}{N}$$

dove N è il numero totale di campioni nel test se e 1 (condizione) denota la funzione indicatore che restituisce 1 se la condizione è verificata e 0 altrimenti.

Codice per lo sviluppo

```
def nearestimage_and_representation(paths_training, index_training, X_training, y_training, ...):
        plt.figure()

        query_image = sio.imread(paths_test[index_test])
        plt.subplot2grid((2,3), (0, 0))
        if(y_test[index_test] == 0): typeofclass = "Food"
        else: typeofclass = "Not-Food"
        plt.title("Query image {0}".format(paths_test[index_test]) )
        plt.imshow(query_image)

        plt.subplot2grid((2,3), (0, 1))
        plt.title("Class {0}".format(typeofclass) )
        plt.plot(X_test[index_test])

        closest_im = sio.imread(paths_training[index_training])
        plt.subplot2grid((2,3), (1, 0))
        if(y_training[index_training] == 0): typeofclass = "Food"
        else: typeofclass = "Not-Food"
        plt.title("Closest Im {0}".format(paths_training[index_training]) )
        plt.imshow(closest_im)

        plt.subplot2grid((2,3), (1, 1))
        plt.title("Class {0}".format(typeofclass) )
        plt.plot(X_training[index_training])

        plt.subplot2grid((2,3), (0, 2), rowspan=2)
        plt.title("Distance {0} - Accuracy {1}%\nConfusion Matrix\n {2}".format(distance, accuracy, matrix))
        plt.plot(X_test[index_test], color='red')
        plt.plot(X_training[index_training], color='blue')

        plt.show()
```

```
nn5 = KNN(5)
nn5.fit(X_training, y_training)
predicted_labels = nn5.predict(X_test)
a = accuracy_score(y_test, predicted_labels)
M = confusion_matrix(y_test, predicted_labels)

print "\n5-NN, accuracy: %0.2f, Confusion Matrix:\n" %a
print "Accuracy: %0.2f" %nn5.score(X_test, y_test)
print M
```

Social Media Management 2016/2017

Pietro Biondi, Federica Greco, Alessandro Maggio

Referenze

- http://iplab.dmi.unict.it/madima2015/ (Paper)
- http://iplab.dmi.unict.it/madima2015/UNICT-FlickrFood.rar (dataset Food)
- http://iplab.dmi.unict.it/madima2015/UNICT-FlickrNon-Food.rar (dataset Not-Food)