Implementazione di un sistema di riconoscimento facciale basato sul metodo di Viola e Jones

Lorenzo Cioni
Apprendimento Automatico
lore.cioni@gmail.com

Abstract

Il riconoscimento facciale gioca un ruolo sempre più determinante in molti settori applicativi, come ad esempio sistemi di sorveglianza, software di editing o applicazioni per l'acquisizione di immagini digitali nei vari dispositivi. In molti casi poi, a causa di alcune specifiche, potrebbe essere necessario dover effettuare dei riconoscimenti in tempo reale, come ad esempio in un video. L'obiettivo di questo elaborato è di implementare un sistema di riconoscimento facciale basato sul metodo proposto da P. Viola e M.J. Jones.

1. Introduzione

Il riconoscimento facciale è stato oggetto di ricerca dai primi anni novanta come parte del ramo del *pattern recognition* e della *object detection*. Il riconoscimento di oggetti e di facce presentano circa gli stessi problemi, ma in particolare, nel secondo caso, i problemi più determinanti sono dati dalla differenza di illuminazione, la scala e la rotazione del volto.

A questi problemi classici del riconoscimento di oggetti se ne aggiungono altri come la gran varietà di espressioni del volto umano o alla presenza di elementi che ne potrebbero compromettere il riconoscimento, come ad esempio una particolare capigliatura o semplicemente un paio di occhiali.

L'obiettivo di questo elaborato è implementare il metodo di riconoscimento facciale proposto da Viola e Jones [1] e valutarne le criticità e performance.

2. Il metodo di Viola e Jones

Il metodo di riconoscimento facciale proposto da Viola e Jones si basa su dei semplici classificatori lineari che operano su delle feature specifiche dell'immagine. Le feature utilizzate sono di tipo binario, molto buone per caratterizzare parti dell'immagine in cui si evidenzia un maggior contrasto: nel caso dei volti queste zone sono tipicamente il naso, gli occhi, etc.

Per consentire di calcolare rapidamente questo tipo di features, Viola e Jones hanno introdotto una nuova rappresentazione dell'immagine chiamata *immagine integrale*. Una volta calcolate queste si procede alla selezione di un loro sottoinsieme utilizzando una versione modificata dell'algoritmo AdaBoost.

Infine, per la classificazione, viene utilizzata una *cascata di classificatori*. I classificatori sono suddivisi in più livelli rendendo possibile il riconoscimento in tempo reale.

3. Features

Uno dei componenti essenziali del metodo di Viola e Jones è l'utilizzo di un sistema di feature molto semplice ed un modo altrettanto veloce per calcolarle.

Viene inizialmente definita una *detection window* di dimensione prefissata (generalmente 24 x 24) e si fa scorrere quest'ultima lungo tutta l'immagine da analizzare alla ricerca di facce. In ciascuna posizione vengono analizzate un'insieme di feature locali, in questo caso le Haar-like.

3.1. Le features Haar-like

Le features Haar-like consistono in 2 o più rettangoli bianchi o neri adiacenti all'interno della *detection window*. A partire da una configurazione di rettangoli viene calcolata la differenza delle somme dei pixel sottostanti una regione bianca e quelli sottostanti ad una regione nera. In questo modo è possibile identificare zone in cui vi è un contrasto nella direzione indicata dalla posizione dei rettangoli. Una zona omogenea infatti darà valori circa prossimi a 0.

Sia X un'immagine. Chiamiamo W l'insieme dei pixel sottostanti una regione *bianca* e B l'insieme dei pixel sottostanti una regione *nera*. Il valore della feature è ottenuto tramite

$$f = \sum_{i \in W} x_i - \sum_{j \in B} x_j$$

Le features utilizzate nel metodo di Viola e Jones sono di 5 tipi diversi, come mostrato in Figura 1.

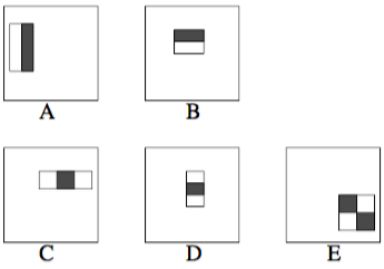

Figura 1. I 5 tipi diversi di Haar-like features utilizzati

Il numero totale di features di tutti e 5 i tipi scalati e traslati all'interno di una *detection window* 24 x 24 è di 162,336. Come nell'implementazione di OpenCV sono state però considerate solo le features con un area superiore a 32 pixels: questo per evitare di andare a considerare delle regioni troppo piccole e irrilevanti ai fini del riconoscimento. Con queste particolari restrizioni il numero totale di features si riduce a 108,490.

3.2. Immagini integrali

Al fine di poter riconoscere i volti in tempo reale è necessario poter valutare queste features in poco tempo. Per fare questo Viola e Jones hanno introdotto una nuova rappresentazione dell'immagine chiamata *immagine integrale*.

L'immagine integrale è costruita utilizzando una matrice della stessa dimensione dell'immagine originale con associata a ciascun elemento la somma dei valori dei pixel che si trovano sopra e a sinistra del pixel corrispondente.

Sia i l'immagine originale e I la sua rappresentazione integrale. Un generico elemento della matrice integrale si trova con:

$$I(x,y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

E' possibile giungere alla costruzione di questa matrice iterativamente utilizzando le seguenti due formule:

$$s(x,y) = s(x,y-1) + i(x,y)$$

$$I(x,y) = I(x-1,y) + s(x,y)$$

A questo punto, data la regione rettangolare D, per calcolare la somma dei pixel all'interno della stessa sarà sufficiente valutare

$$S = I(x_4,y_4) + I(x_1,y_1) - I(x_2,y_2) - I(x_3,y_3)$$

dunque è un calcolo estremamente veloce.

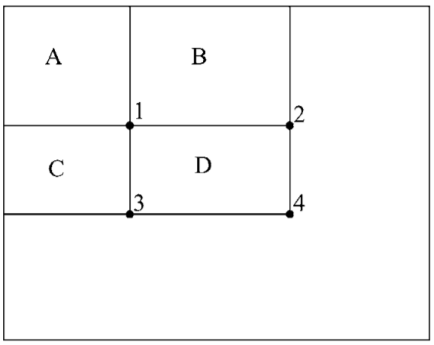

Figura 2. Calcolo della somma dei pixel in una regione rettangolare

4. Boosting

Una volta estratte le features all'interno di ciascuna detection window si procede con la sua classificazione. Per questa fase vengono usati una serie di semplici classificatori binari basati su una singola feature chiamati *decision stumps*.

Come visto in precedenza, anche in una finestra relativamente piccola come una detection window 24 x 24, il numero di feature estratte per ciascuna di esse è molto elevato. Nonostante dunque siano molto veloci da calcolare, il tempo necessario sarebbe troppo per applicazioni in tempo reale.

Si procede dunque alla riduzione dell'insieme delle feature estratte: Viola e Jones utilizzano per questo motivo una versione modificata di AdaBoost per combinare insieme classificatori basati sulle migliori feature selezionate.

4.1. Decision stumps

I *decision stumps* sono dei semplici classificatori binari lineari basati su un singolo valore di una feature.

Un generico classificatore h è definito con una *feature* f, una *soglia* θ ed una *polarità* (*positiva* o *negativa*). La classificazione è data da:

$$h(x,f,\theta,p) = \begin{cases} 1 & \text{se } pf(x) \leq p\theta \\ -1 & \text{altrimenti} \end{cases}$$

Questi classificatori sono chiamati *weak classifiers*.

4.2. AdaBoost

Gli algoritmi di Boosting combinano insieme una serie di *weak classifiers* per andare a formare uno *strong classifier*. Il metodo standard è l'algoritmo di *AdaBoost* (*Adaptive Boosting*), un algoritmo iterativo che seleziona le features migliori in base ad un errore nei singoli classificatori.

Ad un generico passo dell'algoritmo, la selezione della feature migliore dipende dai risultati della precedente iterazione, dando maggiore enfasi agli esempi che causano un più elevato *error rate*. Questo è possibile assegnando a ciascun esempio del training set un peso normalizzato (inizialmente uniforme). Ad ogni iterazione dell'algoritmo il peso viene aggiornato per fare in modo di concentrarsi sugli esempi mal classificati nella precedente iterazione. Il risultato di una singola iterazione è la costruzione di un *weak learner* h_t. L'errore ϵ_t del classificatore così definito è dato dalla somma dei pesi degli esempi classificati erroneamente.

Definiamo inizialmente il numero di iterazioni T. L'algoritmo di AdaBoost modificato da Viola e Jones è definito come Algoritmo 1.

Dato $(x_1, y_1), \ldots, (x_{n+m}, y_{n+m})$, con $x \in X$ e $y \in \{-1, 1\}$, n esempi *positivi* e m esempi *negativi*:

Algoritmo 1 AdaBoost (Viola e Jones)

- Inizializza il peso degli esempi positivi a $\frac{1}{2n}$ e quello degli esempi negativi a $\frac{1}{2m}$

for $t = 1, \ldots, T$ **do**

- Normalizza i pesi così che la loro somma sia unitaria

$$w_i = \frac{w_i}{\sum_{j=1}^{n+m} w_j}$$

- Scegli il miglior *weak learner* h_t tale che

$$h_h = \arg\min_{h_j \in H} \epsilon_j \sum_{i=1}^{n+m} w_i[y_i \neq h_j(x_i)]$$

- Aggiorna i pesi degli esempi

$$w_i = w_i \beta_t^{1-c_i}$$

dove $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$ e c_i vale 1 se l'esempio i-esimo è classificato correttamente, altrimenti vale 0

- Calcola $\alpha_t = \log(\frac{1}{\beta_t}$

end for

- Lo *strong classifier* finale è dato da

$$h(x) = \begin{cases} 1 & \text{se } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ -1 & \text{altrimenti} \end{cases}$$

La parte $\frac{1}{2} \sum_{t=1}^{T} \alpha_t$ è detta *soglia* del classificatore, ma si ottengono migliori risultati andando ad aggiustarla successivamente.

4.3. Considerazioni

Viola e Jones affermano che un classificatore basato su 200 feature estratte giunge ad ottimi risultati in termini di *detection*, ma con un alto costo computazionale che non permette un suo utilizzo in tempo reale, questo a causa del gran numero di feature da calcolare in ciascuna posizione della finestra.

Per ovviare a questo problema è stato proposto un nuovo approccio chiamato *attentional cascade*, in cui una serie di *strong classifiers* sono disposti in cascata in modo tale che siano via via più precisi.

5. Classificazione

5.1. La cascata di classificatori

5.2. Post-processing

6. Risultati

6.1. Creazione del dataset

6.2. Esperimenti

7. Conclusioni

Riferimenti bibliografici

[1] P. Viola and M. J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, May 2004.