

Face Miner

Data Mining applicato alla Face Detection

Paolo Galeone, nessuno@nerdz.eu

Indice

1	Introduzione	2
2	Face Detection	3
2.1	Viola-Jones framework	3
2.1.1	Selezione delle Haar Features	3
2.1.2	Creazione dell'immagine integrale	4
2.1.3	Training dell'algoritmo di Machine Learning: AdaBoost	4
2.1.4	Classificazione a cascata	5
3	Face Miner	6
3.1	Fase di training	6
3.1.1	Preprocessing	7
3.1.2	Ricerca dei feature pattern positivi e negativi	8
3.1.3	MAFIA: MAXimal Frequent Itemset Algorithm	8
3.1.4	Uso di MAFIA in Face Miner	10
3.1.5	Costruzione del face detector	12
3.1.6	Variance classifier	12
3.1.7	Apprendimento delle soglie	13
3.1.8	Features classifier	14
3.1.9	SVM classifier	16
3.1.10	PCA: Principal Component Analysis	17
3.2	Fase di detection	18
3.2.1	Criterio di risposta singola	19
4	Testing ed analisi delle performance	20
4.1	Face Miner	20
4.2	Viola & Jones	20
5	Conclusioni	21

1 Introduzione

In questo documento viene presentato il lavoro svolto per la realizzazione di Face Miner (nel seguito FM): un framework che utilizza tecniche di data mining per l'estrazione automatica di features che verranno utilizzate per costruire classificatori in grado di risolvere il problema della face detection. Le features, in italiano caratteristiche, sono degli elementi in grado di caratterizzare un oggetto o una classe di oggetti. Un esempio di features utilizzate nella classificazione automatica di documenti, ad esempio, sono le parole contenute all'interno del documento. La frequenza, congiuntamente con la posizione e l'importanza attribuita, permettono di classificare il documento in una determinata categoria. La face detection, è il problema dell'individuazione dei volti umani all'interno delle immagini. FM è basato sul paper "A Data Mining approach to Face Detection"[1]. A causa della scarsa riproducibilità del contenuto del paper, FM è una variante di quanto da loro realizzato ed è frutto dell'analisi delle loro scelte e dell'uso di strumenti e metodologie differenti laddove le loro non fossero applicabili oppure dove vi fossero alternative migliori.

Nel seguito saranno presentate le loro scelte e per ognuna di queste sarà presentata la variante utilizzata con relativa motivazione.

Il documento è organizzato come segue: un'introduzione al problema della face detection, mostrando qual è e come funziona lo standard de facto in quest'ambito. Successivamente, verrà illustrata dettagliatamente ogni fase realizzativa di FM, seguendo la struttura della pubblicazione su cui è basato. Per ogni sezione verrà presentata prima l'idea su cui gli autori si sono basati, la loro implementazione e l'eventuale variante implementata in FM. Verrà posta particolare enfasi nella descrizione degli algoritmi di data mining e machine learning utilizzati, mostrando quando questi differiscano da ogni altra implementazione per la risoluzione del medesimo problema.

2 Face Detection

La Face Detection (Riconoscimento facciale) è un caso specifico del problema di object-class detection.

Il compito della object-class detection è quello di trovare la posizione e la dimensione di tutti gli oggetti che appartengono ad una determinata classe. Nel caso della face detection, la classe è quella dei volti.

Un essere umano può effettuare il task di face detection in maniera naturale a differenza di una macchina che necessita di precise informazioni e vincoli da rispettare.

Tra i vari algoritmi presentati in letteratura, il più noto ed utilizzato algoritmo è il Viola-Jones framework.

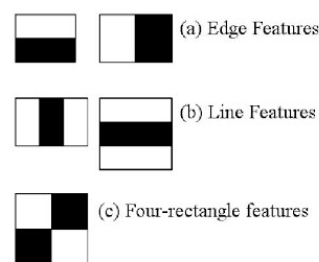


Figura 1: Haar-like features

2.1 Viola-Jones framework

L'algoritmo di Viola-Jones è il primo algoritmo con performance tali da poter essere utilizzato in applicazioni di face detection in real time.

È un algoritmo basato sull'estrazione di determinate caratteristiche (le HAAR like features) e l'uso di tecniche di apprendimento automatico.

Il framework è organizzato come segue:



Figura 2: Selezione delle Haar features

2.1.1 Selezione delle Haar Features

Tutti i volti umani, condividono alcune proprietà simili. Queste regolarità possono essere identificate mediante l'uso delle Haar Features. Le Haar Features non sono altro che kernel (matrici di convoluzione). Come ogni altra matrice di convoluzione, questa mostra la sua utilità nel momento in cui viene eseguita la convoluzione dell'immagine (o di una regione della stessa) con il kernel, applicando un approccio sliding window. L'approccio sliding window consiste nello scandire l'intera immagine, spostandosi di un determinato step verso una direzione, facendo scorrere una "finestra" (una regione rettangolare $n \times n$) su tutta l'immagine. Il risultato del passo di convoluzione è l'estrazione di determinate caratteristiche dall'immagine. L'uso della convoluzione di kernel con l'immagine è alla base di tutte le tecniche di filtering applicate alle immagini (ad esempio la riduzione del rumore è il risultato della convoluzione dell'immagine con un particolare kernel). Le Haar Features originariamente sviluppate da Viola & Jones sono quelle mostrate in figura 1. La parte bianca rappresenta un 1, la parte nera un -1. Il risultato della convoluzione è la sottrazione della somma delle intensità presenti nella zona dell'immagine coperta dalla parte nera, dalla somma delle intensità presenti nella parte bianca. Tutte le possibili dimensioni e posizioni di ogni kernel devono essere calcolate per estrarre le features necessarie. (Utilizzando una window di dimensione 24×24 si ottengono più di 160000 features). Come precedentemente detto, per il calcolo di ogni feature è necessario trovare la somma dei pixel sottostanti il rettangolo bianco e nero. Per risolvere questo problema, Viola & Jones introdussero il concetto di **immagine integrale**, che permette di semplificare notevolmente la complessità del calcolo e rendere di conseguenza l'algoritmo più veloce.

2.1.2 Creazione dell'immagine integrale

Sia I una generica immagine in scala di grigi, come mostrato in figura 3.

Il valore del pixel (x, y) dell'immagine integrale \mathbb{I} rappresenta la somma dei valori di ogni pixel dell'immagine sorgente contenuti all'interno del rettangolo $(0, 0) - (x, y)$:

Il framework crea l'immagine integrale e la utilizza per velocizzare il calcolo delle Haar features. Infatti, grazie all'uso dell'immagine integrale è possibile ad un costo computazionale costante di 4 somme, ottenere la sommatoria di una qualunque sottoparte rettangolare dell'immagine

I .

$$\sum_{x_0 < x \leq x_1} \sum_{y_0 < y \leq y_1} i(x, y) = \mathbb{I}(D) + \mathbb{I}(A) - \mathbb{I}(B) - \mathbb{I}(C)$$

$$\mathbb{I}(x, y) = \sum_{v=0}^y \sum_{u=0}^x I(x, y)$$

Anche utilizzando l'immagine integrale per velocizzare il calcolo, il numero di features calcolate rimane grande. Inoltre, molte di queste non sono rilevanti. Basta guardare l'immagine 2. La prima riga mostra due features "buone". La prima feature, sembra focalizzarsi sulla proprietà che la regione degli occhi è frequentemente più scura della della regione del naso e delle guance. La seconda si basa sulla proprietà che gli occhi sono più scuri del ponte del naso. La stessa window, però, applicata altrove (anche della stessa regione selezionata, come ad esempio sulle guance fornirà features totalmente irrilevanti).

Quindi è necessario scegliere le features migliori, scartando tutte le altre. A questo scopo è utilizzato l'algoritmo **Adaboost**.

2.1.3 Training dell'algoritmo di Machine Learning: AdaBoost

È necessario ottenere un training set, formato da immagini di volti e non volti, opportunamente classificate. Per ogni immagine presente nel training set, vengono calcolate le features come mostrato al punto precedente. Per ognuna di queste feature calcolate, vengono selezionate quelle che producono il minimo errore di classificazione, che significa che sono le features che meglio caratterizzano un volto ed un non volto.

Scendendo nel dettaglio, il processo di Adaptive Boosting (da cui il nome AdaBoost), consiste per l'appunto nell'esecuzione del Boosting adattivo. La tecnica del boosting è una tecnica di data mining (o di machine learning) utilizzata per ridurre i bias (la polarizzazione dei risultati di un algoritmo di classificazione) e la varianza dei dati classificati da un insieme di classificatori "weak", per creare un classificatore "strong". Un weak learner è un classificatore che è solo marginalmente correlato con la vera classificazione (può classificare esempi in maniera poco migliore di un classificatore casuale). Al contrario, uno strong learner è un classificatore che è arbitrariamente ben correlato con la vera classificazione. Lo scopo del boosting è quindi quello di produrre uno strong learner partendo da molti weak learner.

La versione Adattiva dell'algoritmo è quella utilizzata da Viola & Jones. Questa versione si basa sullo stesso principio del boosting (tanti weak classifier per formare uno strong classifier), ma è in grado di determinare automaticamente quali siano le features che maggiormente contribuiscono alla creazione dello strong learner e di conseguenza adattarsi, dando un peso maggiore a queste e

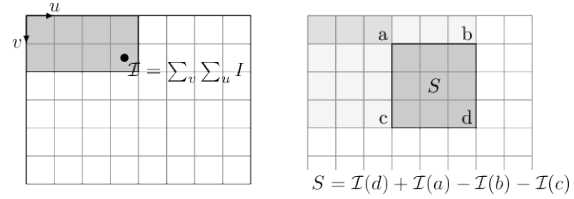


Figura 3: Creazione ed uso dell'immagine integrale

riducendo il peso a quelle inutili. Il classificatore risultante è la somma pesata dei weak classifier. Questi classificatori sono detti weak (deboli, scarsi) perché da soli non sono in grado di classificare un'immagine, ma assieme formano lo strong classifier in grado di farlo.

Viola & Jones hanno così ottenuto una riduzione della dimensionalità. Nella loro pubblicazione, asserirono di essere passati da 160000 features a 6000 per una window 24×24 . Il numero di features estratto, seppur di molto ridotto, è lo stesso enorme. Per risolvere questo problema gli autori hanno introdotto il concetto di **Cascata di classificatori**.

2.1.4 Classificazione a cascata

Il concetto di classificazione a cascata si basa sull'idea che è possibile scomporre il problema dell'identificazione di un pattern, eliminando precocemente ciò che quel pattern sicuramente non può essere. Si utilizza quindi una cascata di classificatori, in cui se l'immagine non viene classificata come un volto dal classificatore n_{i-1} viene scartata immediatamente e non passata al classificatore n_i . Al posto di estrarre tutte le 6000 features necessarie per ogni window, è stato suddiviso il problema in diversi stage (gli autori ne hanno utilizzati 38), ognuno dei quali calcola un numero ridotto di features ed applica la classificazione basandosi esclusivamente su quelle. Se la window attraversa con successo ogni classificatore, allora viene classificata come una faccia. Altrimenti viene scartata non appena un classificatore la classifica come non volto, facendo risparmiare il calcolo di tutte le features rimanenti e quindi aumentando le performance.

3 Face Miner

La realizzazione di Face Miner si basa sulla pubblicazione di Wen-Kwang Tsao et al. [1]. Il loro approccio si basa sull'utilizzo di tecniche di data mining per l'estrazione automatica di features in grado di identificare un volto umano in maniera efficiente. Il lavoro è suddiviso in due fasi:

- Fase di training
- Fase di detection

3.1 Fase di training

La fase di training è dove la maggior parte del lavoro di progetto e di sviluppo è stato effettuato. In questa fase, viene definita l'architettura del classificatore ed utilizzati i metodi di data mining per l'estrazione automatica delle features. Nel seguito saranno mostrate sia le scelte fatte dagli autori, che le variazioni effettuate nella realizzazione di FM.

La fase di training consiste in tre stage:

- **Preprocessing:** utilizzando l'operatore di edge detection di Sobel, verranno estratti i bordi (edge) per ogni immagine di training. Nel seguito l'immagine frutto dell'estrazione dei bordi verrà chiamata edge image.
- **Ricerca dei feature pattern positivi e negativi:** utilizzando l'algoritmo MAFIA [2] per ottenere i maximal frequent pattern (MFP) delle edge image ottenute al punto precedente. Tra i vari MFP, verranno selezionati quelli contenenti il maggior numero di caratteristiche facciali. Il pattern selezionato è detto positive feature pattern. Inoltre, a partire dalle non-edge image, che sono il complemento delle edge image, verranno estratti i negative feature pattern (NFP).
- **Costruzione del face detector:** verranno utilizzati i pattern estratti allo stage precedente per costruire un face detector, che consiste di tre classificatori in cascata. Il primo classificatore, sarà incaricato di effettuare pruning di tutte quelle window che non potranno essere dei volti. Esso si basa sul concetto di varianza ed è detto Variance classifier. Il secondo, utilizza le features estratte al punto precedente e per questo è detto Feature classifier. Il terzo classificatore utilizza una Support Vector Machine (SVM) per raffinare il risultato. Gli autori per velocizzare il processo di detection usando la SVM e per ridurre la dimensionalità, hanno utilizzato un k-d-tree. Nella realizzazione di FM invece la scelta è ricaduta su una tecnica di riduzione della dimensionalità detta Principal Component Analysis che sarà illustrata nel seguito.

Com'è evidente è necessario almeno un training set (per l'estrazione del MFP e per l'allenamento della SVM), contenente esempi positivi (volti) e negativi (non volti). Gli autori del paper hanno utilizzato un dataset privato. Motivo per il quale non è stato possibile replicare esattamente i risultati da loro ottenuti. FM è stato realizzato e testato utilizzando solo dataset pubblici, in maniera tale da poterne garantire la totale replicabilità.

La fase di preprocessing è strettamente legata alla seconda fase, in quanto il preprocessing viene effettuato solo per poter estrarre caratteristiche necessarie per la riuscita della seconda fase.

3.1.1 Preprocessing

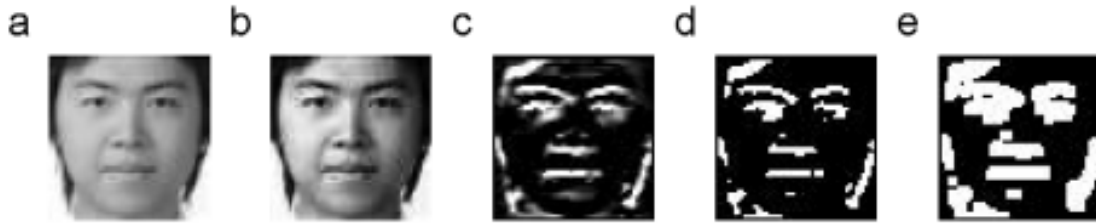


Figura 4: Gli step di preprocessing

Questo stage è praticamente identico a quello realizzato dagli autori. L'unica differenza riguarda la scelta di un parametro costante nell'operazione di thresholding (soglia) dell'immagine. Gli autori non hanno specificato il valore di questo parametro in quanto funzione dello specifico dataset da loro utilizzato (che ricordiamo non essere pubblicamente disponibile).

I passi di preprocessing sono visibili in figura 4.

In questa fase vengono estratti i componenti chiave del volto umano. Prima di tutto è necessario convertire in scala di grigi l'immagine 4(a).

In seguito, si migliora il contrasto dell'immagine, equalizzandone l'istogramma 4(b).

A questo punto si applica l'operatore di Sobel per effettuare edge detection. In particolare, dato che le caratteristiche del volto umano sono prevalentemente composte da linee orizzontali (si pensi ad esempio alle sopracciglia, occhi e labbra, che si sviluppano prevalentemente con tratti allungati orizzontalmente piuttosto che arrotondati o sviluppati in altre direzioni), viene applicato l'operatore di Sobel solo in questa direzione. L'operatore di Sobel è un operatore differenziale, che calcola il valore approssimato del gradiente di una funzione che rappresenta la luminosità dell'immagine. Applicando Sobel in ogni punto dell'immagine ciò che si ottiene è una nuova immagine in cui ogni pixel assume il valore del gradiente in tale punto. Il suo calcolo è effettuato mediante convoluzione dell'immagine con due kernel distinti, uno per direzione. Visto il nostro interesse per la sola direzione orizzontale, una approssimazione del magnitudo del gradiente viene calcolato come:

$$|G| = |G_x| + |G_y|$$

considerando nulla la componente lungo y .

Il risultato della determinazione dei bordi in direzione orizzontale è visibile in 4(c). È evidente dall'immagine che il risultato di dell'applicazione dell'operatore ha generato del rumore che è bene eliminare, in maniera tale da poter ottenere i bordi contenenti la caratteristiche facciali più nitidi e distinti possibile.

Per filtrare (eliminare) i pixel affetti da rumore utilizziamo l'equazione:

$$f(i) = \begin{cases} 255 & \text{se } i > \mu + c\sigma \\ 0 & \text{altrimenti} \end{cases}$$

Dove i è l'intensità del pixel in input, μ l'intensità media dell'immagine, c una costante e σ la deviazione standard dell'intensità dell'immagine.

Gli autori non hanno fornito alcun valore per la costante c , la quale però svolge un ruolo determinante nella discriminazione tra pixel, in quanto a seconda del suo valore, varia l'influenza della deviazione standard e di conseguenza alcuni pixel vengono classificati come sfondo ed altri come bordo. Gli autori probabilmente hanno deciso di non rendere pubblico il valore della costante, in quanto specifica per il dataset utilizzato.

Nella realizzazione di FN, in cui è stato utilizzato il dataset MIT CBCL Face Dataset #1 [5], il valore empiricamente determinato che offre una riduzione generale del rumore su tutte le immagini del dataset senza eliminare elementi significativi per la detection è:

$$c = 0.97$$

Il risultato dell'operazione di thresholding è visibile in 4(d).

L'ultimo step della fase di preprocessing è l'applicazione dell'operatore morfologico di dilatazione, utilizzando come elemento strutturante quello mostrato in figura 5. L'operatore morfologico di dilatazione è anche detto somma di Minkowsky ed è definito come:

$$A \oplus B = \{c \in E^2 : c = a + b, a \in A, b \in B\}$$

cioè l'insieme A dilatazione B è l'insieme di tutte le possibili somme tra elementi di A e di B (intesa come somma di coordinate). In questo caso, A rappresenta l'immagine dopo l'operazione di thresholding e B è l'elemento strutturante.

Il risultato dell'operazione di dilatazione è visibile in figura 4(e).

Terminata la descrizione della fase di preprocessing, si può passare alla descrizione della fase di ricerca dei feature pattern positivi e negativi. In tale fase, verrà utilizzata la fase di preprocessing come step preliminare prima dell'estrazione delle features.

0	1	0
1	1	1
0	1	0

Figura 5: Elemento strutturante utilizzato per l'operazione morfologica di dilatazione

3.1.2 Ricerca dei feature pattern positivi e negativi

In questo stage viene utilizzato l'algoritmo MAFIA [2] per estrarre i maximal frequent patterns dalle edge image ottenute dallo stage precedente.

MAFIA può efficientemente generare i maximal frequent pattern da un database. Il motivo per cui è stato scelto questo algoritmo è che molti pixel identificanti features del volto tendono a co-occorrere all'interno dei volti umani e quindi queste features formeranno un maximal frequent pattern. Dato che non necessitiamo di tutti i maximal frequent itemset estratti, selezioneremo solo quelli che contengono più caratteristiche facciali. Il maximal frequent pattern selezionato viene chiamato positive feature pattern.

Inoltre, viene applicato MAFIA anche sulle non-edge image, ognuna delle quali ricordiamo essere il complemento di una edge-image. In questo modo si ottiene il cosiddetto negative feature pattern.

3.1.3 MAFIA: MAXimal Frequent Itemset Algorithm

MAFIA viene utilizzato per trovare i maximal frequent pattern presenti nelle immagini presenti nel dataset di train, a seguito dello step di preprocessing su ognuna di queste per ottenere l'edge image.

Dimostriamo come funziona l'algoritmo mediante un esempio.

La figura 6 mostra un database di esempio contenente 3 edge image di dimensione 3×3 .

Definiamo la soglia di minimo supporto pari a 0.66.

Image ID	Image	Patterns
I1		$\{(0,0), (2,0), (1,1), (1,2)\}$
I2		$\{(0,0), (2,0), (1,1)\}$
I3		$\{(2,0), (1,1), (1,2)\}$

Figura 6: Database utilizzato da MAFIA

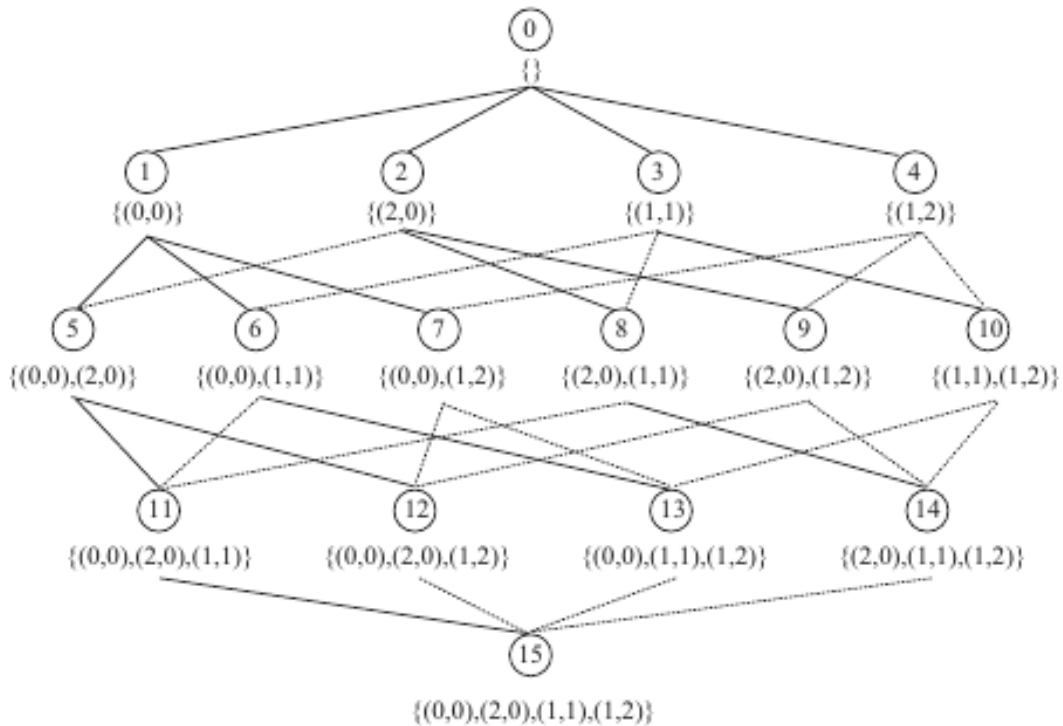


Figura 7: Reticolo sviluppato da mafia per ricercare i MFP

Il supporto è una misura utilizzata nell'ambito del mining di regole associative del tipo $X \rightarrow Y$, dove $X, Y \subset T$, con T database di transazioni, ognuna delle quali contenente determinati item.

Il supporto rappresenta la percentuale di transazioni che contengono $X \cup Y$.

Nell'ambito della scoperta di regole associative, il supporto viene utilizzato per definire regole che possono essere interessanti (devono avere un supporto maggiore di una determinata soglia per esserlo) e per effettuare pruning, eliminando tutti gli itemset che non superano la determinata soglia. Il supporto viene utilizzato in MAFIA per poter eliminare itemset che non soddisfano il vincolo sulla soglia di supporto.

MAFIA utilizza una tecnica reticolare per poter effettuare pruning di itemset che non soddisfano la soglia di supporto.

Nell'immagine 6 è possibile osservare 4 item distinti: $(0,0)$, $(2,0)$, $(1,1)$, $(1,2)$.

Il reticolo costruito per questi 4 item è quello mostrato in figura 7.

MAFIA effettua una ricerca depth-first. Quando giunge in un nodo, controlla il valore del supporto per il determinato nodo ed effettua pruning nel caso in cui questo non soddisfi la soglia di supporto oppure non sia possibile che il nodo sia un maximal frequent pattern.

L'algoritmo inizia dal nodo 0, per poi attraversare il nodo 1, 5, 11 e fermarsi al 15, dato che questo nodo è il nodo più profondo per la diramazione di sinistra.

MAFIA determina che il pattern $\{(0,0), (2,0), (1,1), (1,2)\}$ del nodo 15 non è frequente, siccome il suo supporto è pari a 0.33. A questo punto l'algoritmo effettua un passo di backtracking e torna al nodo 11. Conta il supporto per questo nodo ed ottiene 0.66. Quindi, l'itemset $\{(0,0), (2,0), (1,1)\}$ è frequente. Dato che il pattern del nodo 11 è frequente, i pattern dei suoi antenati (che sono i sotto

pattern di $\{(0,0), (2,0), (2,1)\}$ possono essere esclusi dalla ricerca (passo di pruning), in quanto, per la proprietà Apriori, essi avranno già un supporto maggiore o uguale a quello trovato. La proprietà Apriori ci assicura che:

$$A \subseteq B \Rightarrow s(B) \geq s(A)$$

Dove con $s(X)$ indichiamo il supporto dell'insieme X .

Di conseguenza, MAFIA elimina dalla ricerca i nodi: 6,5,1,2,3 e 0.

L'algoritmo continua con i nodi rimanenti, sempre procedendo in maniera depth-first. Continua quindi ad attraversare il reticolo, passando per i nodi 12,13 e 7. Comunque, il supporto di $\{(0,0), (2,0), (1,2)\}$ (nodo 12), $\{(0,0), (1,1), (1,2)\}$ (nodo 13) e $\{(0,0), (1,2)\}$ (nodo 7) è 0.33, quindi non sono frequenti. E di conseguenza non contribuiscono alla creazione del maximal frequent itemset.

Il nodo 14 viene attraversato. Il supporto di $\{(2,0), (1,1), (1,2)\}$ è 0.6, quindi è frequente. Di conseguenza, i nodi 9,10, e 4 vengono eliminati per la proprietà Apriori.

L'algoritmo termina restituendo i due maximal frequent pattern estratti, che sono: $\{(0,0), (2,0), (1,1)\}$ (nodo 11) e $\{(2,0), (1,1), (1,2)\}$ (nodo 14), come mostrato in figura 8.



Maximal frequent pattern 1		$\{(0,0), (2,0), (1,1)\}$ [0.66]
Maximal frequent pattern 2		$\{(2,0), (1,1), (1,2)\}$ [0.66]

Figura 8: I MFP trovati da MAFIA

3.1.4 Uso di MAFIA in Face Miner

L'algoritmo è in grado di estrarre il maximal frequent itemset di insiemi numerici, non di insiemi di coordinate (come mostrato nell'esempio precedente).

Per poter utilizzare MAFIA con insiemi di coordinate, è stato necessario effettuare un mapping 1 : 1 tra l'insieme delle coordinate e l'insieme dei numeri naturali.

Per far questo è stata utilizzata la funzione **coppia di Cantor**, che è così definita:

$$\pi(k_1, k_2) := \frac{1}{2}(k_1 + k_2)(k_1 + k_2 + 1) + k_2$$

Questa definizione può essere generalizzata in modo da ottenere la funzione **tupla di Cantor**:

$$\pi^{(n)} : \mathbb{N} \rightarrow \mathbb{N}, (k_1, \dots, k_{n-1}, k_n) := \pi(\pi^{(n-1)}(k_1, \dots, k_{n-1}), k_n)$$

La funzione coppia di Cantor crea una relazione biunivoca tra dominio e codominio.

Nel caso delle coordinate in 2 dimensioni (caso di interesse), si definisce

$$z = \frac{(x+y)(x+y+1)}{2} + y$$

Così facendo è stata creata la relazione biunivoca tra $z \in \mathbb{N}$ e $(x, y) \in \mathbb{N}^2$.

È possibile ricavare da z il valore di (x, y) nel seguente modo:

Sia

$$w = \left\lfloor \frac{\sqrt{8z+1}-1}{2} \right\rfloor$$

Si pone

$$t = \frac{w^2 + w}{2}$$

da cui si ricavano le coordinate originarie:

$$y = z - t$$

$$x = w - y$$

La funzione coppia di cantor viene applicata su ogni edge image, campionando i soli pixel bianchi (che identificano gli edge). Per ogni coordinata campionata, viene calcolato il numero naturale corrispondente.

L'insieme dei numeri naturali calcolati per ogni edge-image rappresenta una **transazione**. L'insieme delle transazioni rappresenta il database utilizzato da MAFIA per il calcolo del maximal frequent itemset.

La stessa operazione viene effettuata sulle non-edge image. Il risultato ottenuto sul dataset di test è mostrato in figura 9, nella quale è possibile vedere come il maximal frequent itemset positivo (positive feature pattern) abbia catturato le caratteristiche facciali degli occhi, sopracciglia e bocca.

Mentre quello negativo (negative feature pattern) ha catturato il naso ed i bordi del viso.

La scelta della soglia di minimo supporto per il calcolo di entrambi i feature pattern è critica ed è in grado di far variare di molto il risultato ed il tempo di calcolo.

In particolare, maggiore è la soglia di supporto, minori sono le caratteristiche catturate e minore è il tempo di esecuzione. Maggiore la soglia, maggiori saranno le caratteristiche catturate, ma non è detto che queste siano caratteristiche rappresentative.

Per cui è necessario scegliere accuratamente il supporto per entrambi i pattern estratti, in maniera tale da catturare le parti che si ritengono discriminanti all'interno del volto, senza tagliare troppo ma anche senza includerne di inutili.

Inoltre, è bene ricordare che a seconda del dataset utilizzato il risultato varia.

La scelta dei parametri per questa fase è stata determinata empiricamente e sono:

- Minimo supporto per il positive feature pattern: 0.7
- Minimo supporto per il negative feature pattern: 0.79

Gli autori, con riferimento al loro dataset, hanno utilizzato valori diversi che sono 0.95 e 0.98 rispettivamente.

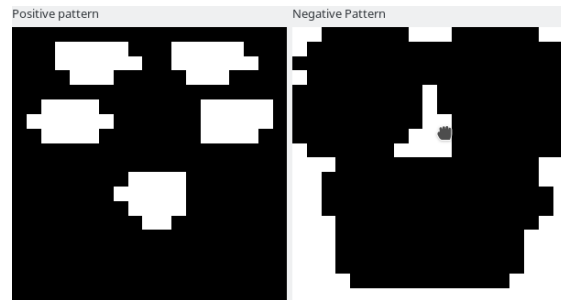


Figura 9: Positive e Negative feature pattern estratti da MAFIA usando il dataset MIT CIBCL

3.1.5 Costruzione del face detector

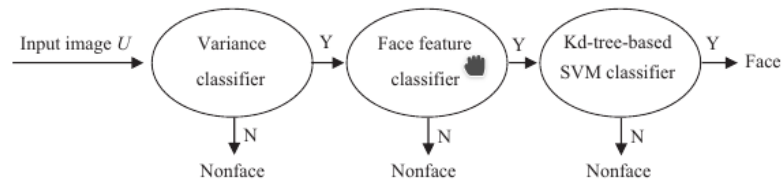


Figura 10: Classificazione a cascata adottata

Basandosi sui positive e negative feature pattern estratti, adotteremo strategia corse-to-fine e simple-to-complex per costruire il face detector.

Dall'analisi del framework di Viola & Jones fatta nel capitolo precedente, è facile notare come gli autori del paper a cui FM è ispirato si siano ispirati a loro volta.

Infatti, la strategia simple-to-complex è l'equivalente dell'utilizzo dei classificatori a cascata per le Haar features. Le prime, infatti, escluderanno le parti grossolane in cui sicuramente non vi sarà presente un viso. Andando via via ad aggiungere test fino ad ottenere una selezione fine.

Nel paper, gli autori hanno deciso di adottare anch'essi una struttura a cascata per realizzare il classificatore finale. I classificatori sono organizzati come mostrato in figura 10.

In particolare, sono stati definiti tre classificatori. Il primo, permette di eliminare in maniera rapida le parti più grossolane dell'immagine, basandosi sul calcolo della varianza per alcune regioni della window considerata. Questo classificatore è detto variance classifier.

Il secondo utilizza i feature pattern estratti tramite MAFIA per classificare la window in esame. Questo classificatore è detto features classifier.

Infine, il terzo classificatore utilizza una SVM per discriminare in maniera ancor più fine se la window in ingresso è o meno un viso.

Per i primi due classificatori, gli autori hanno generalmente indicato che è necessario l'apprendimento della soglia discriminante (il decision boundary) tra volti e non volti.

L'apprendimento delle soglie non è stato spiegato dagli autori. L'unica informazione che questi forniscono è un grafico che mostra la percentuale di volti che il classificatore deve lasciare.

La soglia fissata dagli autori è del 95%. Nell'implementazione di FM questo valore differisce.

L'implementazione di questi classificatori rispetto a quanto descritto nel paper, varia in maniera sostanziale. Nel seguito verranno mostrati i classificatori sia come proposti nel paper che come effettivamente implementati.

3.1.6 Variance classifier

Per aumentare l'efficienza del classificatore, viene adottato un criterio di esclusione immediata per eliminare le window da confrontare.

Questo criterio divide la window in 5 regioni, come mostrato in figura 11, dove i numeri accanto le regioni sono le distanze in termini di pixel.

Gli autori, lavorando con window 21×21 hanno adottato quella suddivisione. Nell'implementazione, invece, lavorando con un dataset con immagini 19×19 , sono state adottate misure lievemente differenti che però rispettano la stessa proporzione e suddivisione delle zone del volto.

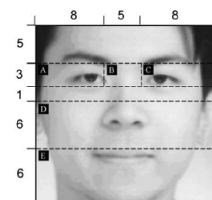


Figura 11: Le regioni in cui viene diviso un sample e su cui si basa il variance classifier

Le regioni sono:

- Occhio di sinistra (A) [2×7]
- Ponte del naso (B) [2×5]
- Occhio di destra (C) [2×7]
- Il naso (D) [5×19]
- La bocca (E) [6×19]

Il criterio si basa sul calcolo delle varianze delle regioni D ed E, rispettivamente.

La window verrà immediatamente scartata e classificata come non volto se una delle due varianze è minore di una soglia predefinita (che è necessario apprendere).

Se la window supera questo primo step, allora verranno calcolati i valori di:

- m_a : l'intensità media di quei pixel che sono più scuri dell'intensità media della regione A
- m_b : l'intensità media di quei pixel che sono più chiari dell'intensità media della regione B
- m_c : l'intensità media di quei pixel che sono più scuri dell'intensità media della regione C

La window verrà esclusa se:

$$m_b < k \cdot m_a \text{ o } m_b < k \cdot m_c$$

dove k è un fattore di controllo costante, che dev'essere appreso.

3.1.7 Apprendimento delle soglie

La fase di apprendimento delle soglie per il variance classifier in FM consiste nella valutazione dei valori di **precision** e **recall** al variare del valore di T e k .

- **Precision**: è la frazione di item recuperati che sono rilevanti.

$$Precision = \frac{TP}{TP + TN} \in [0, 1]$$

- **Recall**: è la frazione di item rilevanti che sono stati recuperati.

$$Recall = \frac{TP}{TP + FN} \in [0, 1]$$

I valori di TP, TN, FP, FN rappresentano rispettivamente:

- TP : i veri positivi. Volti \in test set e classificati come tali.
- TN : i veri negativi. Non volti \in test set e classificati come tali.
- FP : i falsi positivi. Non volti \in test set e classificati come volti.
- FN : i falsi negativi. Volti \in test set e classificati come non volti.

Il dataset MIT CBCL è già suddiviso in train e test set, in questo modo è stato possibile testare le performance del classificatore senza dover ricorrere alla cross-validation.

La cross validation è un processo di suddivisione del dataset in k sottoinsiemi, in maniera tale da poter valutare le performance dell'algoritmo di classificazione simulando un ambiente reale, cioè con variabilità dei campioni. Il processo detto di k -fold cross validation, consiste nella suddivisione in k

sottoinsiemi. $k - 1$ vengono utilizzati per allenare il classificatore e quello rimanente per testarne la qualità. Il processo viene ripetuto k volte e la valutazione complessiva viene fatta aggregando le k valutazioni (facendone la media).

L'apprendimento delle soglie per FM è stato un processo iterativo, in cui sono state valutati manualmente i valori di precision e recall al variare sia di T che di k .

Sono stati testati diversi valori, fino a giungere a:

$$k = 1.65$$

$$T = 182$$

Che generano la matrice di confusione formata dai valori:

$$TP = 253, TN = 15641$$

$$FP = 7359, FN = 219$$

Che implicano:

$$Precision = 0.03$$

$$Recall = 0.57$$

Questi valori vanno considerati nel loro contesto, che è il passaggio del solo variance classifier su un'immagine 19x19 un'unica volta.

In FM, utilizzando un approccio alla detection piramidale, ci si può permettere di avere valori di precisione bassi, in quanto un volto viene analizzato su più livelli e quindi se non viene identificato ad una scala esiste la possibilità che venga identificato in quella successiva.

Lo scopo principale di questo classificatore è quello di eliminare il prima possibile zone che sicuramente non sono volti. Il valore di TN indica che l'obiettivo è stato raggiunto, senza penalizzare eccessivamente la detection dei volti (basta guardare il valore di TP , sempre ricordando l'approccio piramidale).

3.1.8 Features classifier

Questo classificatore utilizza i positive e negative feature pattern estratti mediante MAFIA per selezionare solo quelle window che hanno il maggior numero di caratteristiche di un volto e scartare quelle che sicuramente non lo sono. Per semplificare il processo, viene calcolata la differenza tra le intensità dei pixel tra il pattern positivo e negativo, sia nella raw image che nella edge image.

La raw image è l'immagine grezza, quella che non ha subito alcun passo di preprocessing.

I valori da calcolare sono:

- C_1 : la somma delle intensità dei pixel del positive feature pattern nella raw image
- C_2 : la somma delle intensità dei pixel del negative feature pattern nella raw image
- C_3 : la somma delle intensità dei pixel del positive feature pattern nella edge image
- C_4 : la somma delle intensità dei pixel del negative feature pattern nella edge image

I test da superare affinché la window sia considerata una buona candidata per essere un volto sono:

- **Regola 1:** $C_1 - C_2 > T_1$
- **Regola 2:** $C_3 - C_4 > T_2$
- **Regole 3,4,5,6:** $T_{i,lower} < C_i < T_{i,upper}, i = 1, 2, 3, 4$

Le sei soglie necessarie per la classificazione devono essere apprese. Anche in questo caso gli autori non spiegano come ottenere le soglie, ma danno solo un'indicazione generica, dicendo che il classificatore deve classificare come volti il 95% dei volti presenti nel test set.

Ogni immagine che soddisfa le 6 regole sarà considerata una buona candidata e verrà quindi passata al classificatore successivo.

I valori delle soglie sono stati identificati mediante un lungo processo di tipo trial-and-error, manualmente.

L'idea di base è quella di eliminare gli outlier che potrebbero essere presenti all'interno del dataset e costruire la soglia come quel valore che meglio è in grado di suddividere i volti dai non volti.

Per farlo, sono stati costruiti dei vettori contenenti i valori di C_1, C_2, C_3, C_4 per ogni immagine. Questi vettori sono stati poi ordinati e sono stati rimossi i valori più alti e più bassi (dei probabili outlier). Onde evitare di considerare più volte elementi con lo stesso valore, sono stati eliminati i duplicati.

Il valore di ogni soglia è stato poi individuato testando manualmente quale valore rendeva migliori i valori delle metriche Precision e Recall. Di seguito sono mostrati i valori trovati:

$$T_1 = -5648, T_2 = 2512$$

$$T_{1,lower} = 4028, T_{1,upper} = 13584$$

$$T_{2,lower} = 5883, T_{2,upper} = 16376$$

$$T_{3,lower} = 5258, T_{3,upper} = 10333$$

$$T_{4,lower} = 1481, T_{4,upper} = 3924$$

Che ha prodotto il seguente benchmark:

True positive: 239

True negative: 21750

False positive: 1823

False negative: 233

Precision: 0.115907

Recall: 0.506356

Negative detection rate: 0.922666

La parte di maggior interesse è la negative detection rate, che è molto alta. Questo permette di utilizzare questo classificatore per effettuare un buon pruning dei non volti.

3.1.9 SVM classifier

Per aumentare l'accuratezza del face detector ed evitare di avere troppi falsi positivi, gli autori utilizzano una SVM ed un k-d-tree come ultimo stage del face detector. Questo è il classificatore che maggiormente differisce nell'implementazione rispetto a quanto descritto nel paper.

Le variazioni principali riguardano:

- Quali features utilizzare
- Come ridurre la dimensionalità dei vettori

Nel seguito le differenze verranno mostrate e motivate.

Iniziamo con il descrivere l'implementazione degli autori:

Per un problema di classificazione binaria, dove un dataset $D = \{(x_i, y_i)\}_{i=1}^l$ contiene l feature vector (vettori di caratteristiche) e $y_i \in \{-1, 1\}$, la SVM è in grado di classificare i vettori in due classi con errore minimo. Intuitivamente, la SVM cerca un iperpiano a massimo margine in grado di suddividere gli elementi delle due classi.

Il margine è definito come la somma delle distanze dall'iperpiano dei vettori più vicini di entrambe le classi. Un iperpiano ottimo è utilizzato per classificare il dataset in due classi come mostrato in figura 12, dove v_1, v_2, v_3 sono i vettori di supporto che si trovano al confine tra le classi.

In molti casi, può essere difficile trovare un iperpiano per classificare ogni vettore del dataset nella classe corretta. Ma possiamo sempre trovare un iperpiano che massimizzi il margine e che contemporaneamente minimizzi il numero di vettori classificati erroneamente.

Se il dataset non può essere classificato mediante un classificatore lineare (cioè utilizzando un iperpiano), possiamo proiettare i vettori del dataset in uno spazio a dimensionalità maggiore dove le componenti sono linearmente separabili: per fare ciò si utilizzano delle funzioni, dette kernel function (delle quali sigmoide e radiale sono un esempio).

Le features che gli autori hanno deciso di utilizzare per allenare ed utilizzare la SVM sono quelle che presentano le caratteristiche facciali più discriminanti, cioè la regione degli occhi e della bocca. Quindi, dal loro campione 21×21 hanno estratto:

- **Le intensità delle righe [8-11] e [11-14]:** quindi 21×8 caratteristiche.
- **I coefficienti della trasformata di Haar delle stesse righe:** quindi altre 21×8 caratteristiche.

Quindi la SVM utilizza vettori di $2(21 \cdot 8) = 336$ dimensioni. [19 x 2 from (0, 5)] [19 x 3 from (0, 11)] A differenza di quanto appena detto, nell'implementazione di FM vengono utilizzati **solo** i coefficienti della trasformata di Haar.

La scelta è giustificata dal fatto che la trasformata di Haar altro non è che una trasformazione delle intensità luminose dei pixel in un altro dominio. Inoltre essendo la trasformata di Haar invertibile, è possibile passare da un dominio all'altro e viceversa, di conseguenza entrambi i valori scelti dagli autori codificano la stessa informazione e quindi sono ridondanti.

Sono stati scelti i valori della trasformata e non le intensità dei pixel anche perché è stato empiricamente osservato che la SVM è in grado di determinare un iperpiano a margine maggiore lavorando con i valori della trasformata anziché con i valori delle intensità.

Inoltre, vista la dimensione ridotta delle istanze del dataset, il numero di features con cui la SVM di FM lavora è:

$$19 \times 5 = 95$$

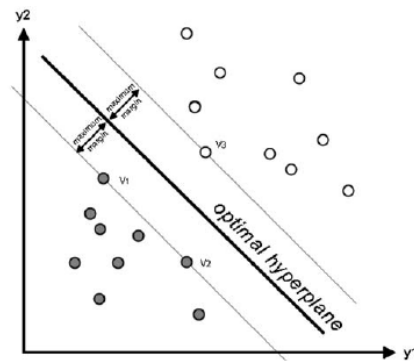


Figura 12: Una SVM cerca l'iperpiano a massimo margine

Le regioni selezionate sono:

- 19×2 nella regione degli occhi, a partire dalla quinta riga
- 19×3 nella regione della bocca, a partire dalla undicesima riga

Gli autori a questo punto decidono di utilizzare il k-d-tree per ridurre la dimensionalità dei vettori e quindi velocizzare il processo di detection della SVM.

Il kd-tree è la naturale generazione a k dimensioni dell'albero di ricerca binario. Gli autori decidono di utilizzare questo albero per effettuare delle proiezioni lungo alcune delle dimensioni, in maniera tale da avere nelle foglie solo alcuni features.

Infine utilizzano le feature presenti nelle foglie per allenare la SVM e per effettuare poi la detection.

In FM, invece, è stata utilizzata solo la SVM, senza utilizzare il k-d-tree, ma adottando un'altra tecnica per la riduzione della dimensionalità: la Principal Component Analysis (PCA).

3.1.10 PCA: Principal Component Analysis

L'analisi delle componenti principali è una tecnica di riduzione della dimensionalità di tipo algebrico e statistico, che permette di eliminare componenti a bassa varianza che conseguentemente sono meno discriminative rispetto alle rimanenti (con più alta varianza).

Algebricamente, la PCA ci permette di trovare la direzione lungo a quale i dati variano maggiormente. Infatti, il risultato di una PCA su dataset formato da punti in 2 dimensioni produce due vettori, detti eigenvector (o autovettori) che sono le componenti principali dell'intero dataset.

La dimensione di ogni autovettore è codificato nel corrispondente autovalore che indica quanto i dati varino lungo la componente principale.

L'origine nel nuovo sistema di riferimento identificato dai due autovettori è posizionato al centro di tutti i punti nel dataset.

Applicando la PCA ad un dataset n -dimensionale, otterremo quindi n autovettori n -dimensionali, n autovalori ed 1 punto n -dimensionale al centro.

La PCA permette di selezionare solo $k \leq n$ componenti principali, arbitrariamente. In quanto:

- Le componenti principali sono tra loro non correlate
- Le componenti principali sono ordinate in ragione della variabilità complessiva che esse possono sintetizzare, cioè ordinate in base al valore dell'autovalore associato

Quindi, scegliendo di preservare solo le prime k componenti, queste saranno quelle a variabilità maggiore rispetto alle $n - k$ rimanenti.

La scelta delle componenti va fatta in maniera empirica, a seconda dell'informazione che queste codificano e dell'impatto che hanno sulla classificazione.

In FM, si è osservato come il numero minimo di features in grado di offrire un buon detection rate è 29 che è meno di un terzo della dimensione originale, oltre che essere notevolmente inferiore alla dimensione originale degli autori.

Gli autori del paper, non hanno esplicitato il numero di features presenti nei vettori della SVM.

3.2 Fase di detection

Nella fase di detection viene utilizzato un approccio sliding window per la ricerca di volti all'interno delle immagini di test, congiuntamente alla rappresentazione multi-scala dell'immagine (approccio piramidale).

L'approccio sliding window consiste nel far scorrere una window, che è una regione rettangolare, sopra l'immagine e valutare la zona sottostante la finestra.

La window inoltre può scorrere di un determinato numero di pixel per volta in direzione orizzontale e verticale, questo numero è detto step.

In FM e nel paper, lo step in entrambe le direzioni è stato fissato a due pixel.

La rappresentazione multi-scala dell'immagine consiste in una rappresentazione su multiple scale dell'immagine originaria, in maniera tale da poter rilevare volti anche su scale differenti. Ogni livello è il risultato di un operazione di sotto-campionamento (downsampling) dell'immagine precedente, che ne riduce le dimensioni, rispettando le proporzioni (vedi figura 13).

Il downsampling consiste nella riduzione della dimensione dell'immagine del layer precedente di un fattore costante. In FM ed anche nel paper, è stato utilizzato il fattore di scala di 1.25.

La combinazione di queste due tecniche consiste nel fare scorrere la sliding window su ogni livello della piramide, in maniera tale identificare volti in scale diverse.

Gli autori del paper, a questo punto, sostengono che l'immagine dev'essere preprocessata e poi data in input al classificatore a cascata.

Questo però **non è possibile** in quanto il primo classificatore si basa sulla varianza. Varianza che è sempre nulla al termine dell'operazione di preprocessing, proprio perché le zone da analizzare sono frutto della segmentazione che ha reso le zone o bianche o nere.

Inoltre, anche il secondo ed il terzo classificatore, non possono utilizzare le immagini preprocessate, in quanto il secondo classificatore necessita sia delle edge image che della raw image per calcolare i valori delle costanti, mentre il terzo classificatore deve poter calcolare il valore della trasformata di Haar dell'intensità dei pixel. Se la trasformata fosse applicata ai pixel frutto del preprocessing, avrebbe solo due valori (trasformata dell'intensità di nero e trasformata dell'intensità di bianco).

L'approccio utilizzato in FM è quello di fornire in ingresso ad ogni classificatore l'immagine raw e lasciare ad ogni classificatore il compito di preprocessarla a seconda delle sue esigenze.

Gli autori, inoltre, utilizzano un metodo di scoring delle window che hanno passato i 3 step di classificazione, collezionando le window candidate in una pool e calcolando per ognuna il proprio score come $(C_1 + C_3) + (C_2 + C_4)$.

Dopo aver collezionato i candidati per ogni scala, uniscono quelli che appartengono alla stessa regione dell'immagine originale e che hanno rank maggiore. Considerano le regioni sovrapposte se le window, ri-scalate in dimensione originale, hanno la distanza tra i loro centri che è minore di un quinto della dimensione della finestra.

L'approccio utilizzato in FM invece è diverso e più performante.

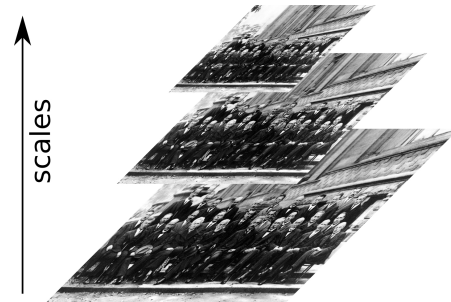


Figura 13: Approccio piramidale

3.2.1 Criterio di risposta singola

Il criterio di risposta singola consiste nell'identificare il prima possibile un volto e racchiuderlo in un'area quadrata.

La ricerca parte dalla punta della piramide, cioè facendo scorrere la sliding window nell'immagine più piccola possibile e ricercando volti a quella scala (che identificano poi dei volti in primo piano).

Non appena un volto, ad una qualsiasi scala, viene identificato, viene immediatamente racchiuso in un rettangolo che indica che la zona **non deve più** essere oggetto di ricerca per i layer successivi.

Quindi in ogni livello, si esclude la proiezione verso il basso dell'area racchiusa dal rettangolo che identifica il volto.

In questo modo, non vi è alcuna necessità né di cercare all'interno di zone in cui è già nota la presenza di un volto né tantomeno collezionare e dare un'ordine alle regioni trovate.

Il criterio quindi, fa saltare intere aree di ricerca, aumentando così la velocità della detection. La sliding window salterà direttamente alla zona immediatamente successiva alla proiezione.

Questo implica che FM avrà tempi di detection diversi a seconda di quanto prima (nel senso di livelli alti della piramide) riesce ad individuare un volto e di conseguenza saltare aree di dimensione sempre maggiore nei livelli successivi.

4 Testing ed analisi delle performance

Il test della qualità della detection e delle performance è stato condotto sullo Yale Face Database [9]. Il dataset è formato da 166 immagini di volti di 15 soggetti differenti, in diverse condizioni di illuminazione ed in posizione frontale alla telecamera. Ogni immagine ha dimensione 320 x 243.

L'analisi delle performance è stata realizzata confrontando le performance dell'algoritmo di Viola & Jones rispetto a face miner, utilizzando un'implementazione in C++ di entrambi gli algoritmi su un computer avente CPU Intel(R) Core(TM) i3-3217U CPU @ 1.80GHz e 4 GB di RAM.

I parametri dell'algoritmo di Viola & Jones sono gli stessi utilizzati in face miner: stesso fattore di scala di 1.25 e stessa grandezza della detection window 19 x 19.

4.1 Face Miner

Le performance misurate, costruendo manualmente la matrice di confusione, sono:

- **Precision:** $\frac{TP}{TP+TN} = \frac{124}{124+7} = 0.94$

- **Recall:** $\frac{TP}{TP+FN} = \frac{124}{124+37} = 0.77$

I valori di precision e recall sono abbastanza buoni. Tuttavia, la velocità di detection risulta molto lenta, in quanto in media, per analizzare un'immagine 320 x 243 px sono necessari: **0.528 secondi**. Un gran numero di falsi positivi sono stati individuati in zone che "assomigliano" ad un volto: cioè zone che hanno una differenza tra le tonalità di grigio nella parte alta e bassa che ricorda molto il positive pattern minato mediante MAFIA.

Inoltre, si osserva che tra i falsi negativi molti sono contenenti volti ruotati lievemente oppure in condizioni di luce non ideali.

Di conseguenza Face Miner non è robusto alle rotazioni ed alle variazioni di illuminazione.

4.2 Viola & Jones

Viola & Jones si dimostra non solo robusto, ma anche estremamente performante.

Il risultato del benchmark sullo stesso dataset ha fornito i seguenti valori di precision e recall:

- **Precision:** $\frac{TP}{TP+TN} = \frac{166}{166+0} = 1$

- **Recall:** $\frac{TP}{TP+FN} = \frac{166}{166+0} = 1$

I valori di precision e recall sono quelli del classificatore ideale.

Questi valori sono fortemente influenzati dal dataset utilizzato, in quanto contenente solo volti e non altri elementi distrattori. In condizioni reali, in cui nell'immagine sono presenti elementi che sono non-volti, le performance saranno (lievemente) inferiori.

In ogni modo, la qualità del classificatore di Viola & Jones è molto alta ed è per questo che è l'algoritmo di face detection più utilizzato.

Oltre alla qualità, la velocità media dell'algoritmo è **di molto** superiore a quella di face miner, infatti è di solo: 0.017 secondi.

5 Conclusioni

La non riproducibilità del paper utilizzato come base per la realizzazione di face miner non ha reso possibile verificare, effettuando una riproduzione esatta, il lavoro degli autori.

Tuttavia, la realizzazione effettuata in face miner ha dato risultati soddisfacenti per quanto riguarda la qualità della detection (ricordiamo la precision del 94%), ma molto meno per quanto riguarda la performance della detection.

Inoltre è inevitabile domandarsi se effettivamente le differenze implementative e l'utilizzo di altri dataset rispetto al paper originale, possano portare ad un miglioramento delle performance tale da rendere questo migliore rispetto a quello di Viola-Jones, come asserito dagli autori nel paper.

In ogni modo, l'idea di utilizzare un approccio di data mining per la determinazione automatica delle features da utilizzare per creare i classificatori si è dimostrata valida ed in grado di funzionare dopo aver effettuata l'adeguato tuning dei parametri.

Sitografia / Bibliografia

- [1] *A Data Mining approach to Face Detection*
<http://dl.acm.org/citation.cfm?id=1660713>
- [2] *MAFIA: MAXimal Frequent Itemset Algorithm*
<http://himalaya-tools.sourceforge.net/Mafia/>
- [3] *Sensitivity and Specificity*
https://en.wikipedia.org/wiki/Sensitivity_and_specificity
- [4] *Precision and Recall*
https://en.wikipedia.org/wiki/Precision_and_recall
- [5] *CBCL Face Database #1*
MIT Center For Biological and Computation Learning <http://www.ai.mit.edu/projects/cbcl>
- [6] *Immagine Integrale*
<http://www.ce.unipr.it/people/medici/geometry/node29.html>
- [7] *Face detection using Haar Cascades*
http://docs.opencv.org/master/d7/d8b/tutorial_py_face_detection.html
- [8] *Funzione coppia*
https://it.wikipedia.org/wiki/Funzione_coppia
- [9] *Yale face database* P. Belhumeur, J. Hespanha, D. Kriegman, Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection, IEEE Transactions on Pattern Analysis and Machine Intelligence, July 1997, pp. 711-720.