

Implementazione di una libreria in Python per la Action Unit detection tramite SVM e SVR

Luca Arrotta






























Upper Face Action Units					
AU 1	AU 2	AU 4	AU 5	AU 6	AU 7
					
Inner Brow Raiser	Outer Brow Raiser	Brow Lowerer	Upper Lid Raiser	Cheek Raiser	Lid Tightener
*AU 41	*AU 42	*AU 43	AU 44	AU 45	AU 46
					
Lid Droop	Slit	Eyes Closed	Squint	Blink	Wink
Lower Face Action Units					
AU 9	AU 10	AU 11	AU 12	AU 13	AU 14
					
Nose Wrinkler	Upper Lip Raiser	Nasolabial Deepener	Lip Corner Puller	Cheek Puffer	Dimpler
AU 15	AU 16	AU 17	AU 18	AU 20	AU 22
					
Lip Corner Depressor	Lower Lip Depressor	Chin Raiser	Lip Puckerer	Lip Stretcher	Lip Funneler
AU 23	AU 24	*AU 25	*AU 26	*AU 27	AU 28
					
Lip Tightener	Lip Pressor	Lips Part	Jaw Drop	Mouth Stretch	Lip Suck

Fig. 1: Tabella che presenta alcune delle possibili Action Unit.

Abstract—In questo progetto, basato sull'implementazione del software OpenFace, sono state addestrate diverse Support Vector Machine (SVM) e Support Vector Machine per la Regressione (SVR), per il rilevamento della presenza e dell'intensità delle Action Unit che compaiono sul volto di un soggetto all'interno di un'immagine. I modelli SVM e SVR sono stati addestrati utilizzando le immagini presenti in tre diversi dataset: CK+, UNBC e DISFA+. L'immagine di input, prima di essere passata ai modelli SVM e SVR, subisce una pipeline di operazioni, le quali hanno lo scopo di estrarre da essa un vettore di feature con cui caratterizzarla.

1 INTRODUZIONE

Il volto è il canale principale attraverso cui avviene la comunicazione non verbale. Le espressioni facciali ci forniscono informazioni riguardanti la personalità, le emozioni e le intenzioni di un essere umano. Per questo motivo, l'analisi automatica delle espressioni facciali può essere uno strumento molto utile nelle applicazioni che

L. Arrotta, Corso di Modelli di Computazione Affettiva, A/A 2018-2019, Università degli Studi di Milano, Via Celoria 18, Milano, Italy
E-mail: luca.arrotta@studenti.unimi.it

Emotion	Action units
Happiness	6+12
Sadness	1+4+15
Surprise	1+2+5B+26
Fear	1+2+4+5+7+20+26
Anger	4+5+7+23
Disgust	9+15+16
Contempt	12A+ 14A

Fig. 2: Combinazioni di Action Unit associate ad alcune emozioni di base.

riguardano l'interazione uomo-macchina (come ad esempio gli assistenti personali virtuali) e la diagnosi di patologie come la depressione.

Il Facial Action Coding System (FACS) [1] è un sistema di codifica delle espressioni facciali che presenta una lista di Action Unit (AU), ognuna delle quali si riferisce agli effetti visibili che produce l'attivazione di uno o più muscoli sul volto di una persona. In Figura 1 è riportata una tabella che presenta alcuni esempi di Action Unit. Dalla combinazione di Action Unit relative ai soli muscoli facciali è possibile riconoscere alcune emozioni di base (Figura 2). Ad ogni AU può essere associata anche una lettera (dalla A alla E), che indica l'intensità con cui è attivo un certo gruppo di muscoli.

Il contributo di questo progetto consiste nell'implementazione di una libreria in Python che permetta di rilevare la presenza (tramite SVM) e l'intensità (tramite SVR) delle AU che compaiono in un'immagine, basandosi sulle feature estratte da quest'ultima.

2 ANALISI DELLO STATO DELL'ARTE

OpenFace [2] è un toolkit che fornisce strumenti utili per l'analisi dei comportamenti facciali. Tra le altre cose, permette di rilevare i landmark, le feature HOG e le AU relative al volto del soggetto presente nel video passato in input al software.

I landmark sono dei punti di riferimento che possiamo rilevare all'interno di un volto (Figura 3). Rilevare i

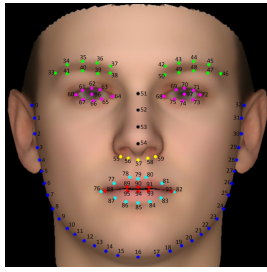


Fig. 3: Esempio di landmark presenti sul volto di un essere umano.



Fig. 4: Esempio di feature HOG relative ad un'immagine.

landmark di un volto, ci consente per esempio di sapere la posizione in cui si trovano gli occhi di una persona all'interno di un'immagine.

L'Histogram of Oriented Gradients (HOG) è un descrittore di feature, cioè una rappresentazione semplificata di un'immagine che permette di estrarre da essa solo le caratteristiche più rilevanti [3]. L'HOG descrive un'immagine tramite la distribuzione delle direzioni dei gradienti all'interno dell'immagine stessa (Figura 4). I gradienti sono importanti in un'immagine, perché la loro magnitudo cresce in corrispondenza di bordi e angoli. Rilevare bordi e angoli è molto utile per capire la forma degli oggetti presenti nell'immagine.

Per rilevare la presenza e l'intensità delle AU, OpenFace applica la seguente pipeline di operazioni alle immagini che riceve in input:

- Rilevamento dei Landmark dell'immagine
- Allineamento dell'immagine in un sistema di riferimento comune a tutte le immagini considerate
- Applicazione di una maschera per ottenere solo la porzione di volto delimitata dai landmark
- Estrazione delle feature HOG
- Riduzione della dimensionalità, tramite PCA, del vettore di feature HOG ottenuto
- Concatenazione delle feature HOG e dei Landmark
- Il vettore di feature così ottenuto viene infine passato alle SVM e SVR

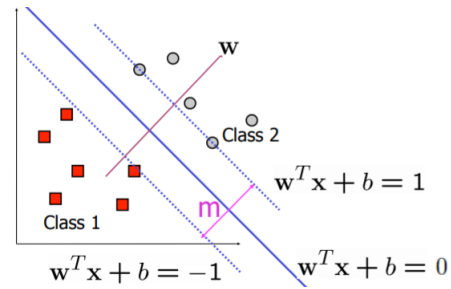


Fig. 5: Rappresentazione geometrica dell'iperpiano separatore a margine massimo.

3 MODELLO TEORICO [4] [5] [6] [7] [8]

3.1 Support Vector Machine (SVM)

Consideriamo di lavorare con un training set $(x_1, y_1), \dots, (x_n, y_n) \in \mathbb{R}^d \times \{+1, -1\}$. Lo scopo delle SVM è quello di trovare l'iperpiano a margine massimo che divide l'insieme dei punti con etichetta +1 dall'insieme di punti con etichetta -1. Il margine sarebbe la somma tra la distanza dell'iperpiano separatore dal punto a lui più vicino con etichetta +1 e la distanza dell'iperpiano separatore dal punto a lui più vicino con etichetta -1. Questi punti più vicini sono i cosiddetti Vettori di Supporto.

Tale iperpiano può essere considerato come l'insieme dei punti x tali per cui

$$w^T x + b = 0$$

dove w è la normale dell'iperpiano. In Figura 5 è mostrata la rappresentazione geometrica di quanto appena descritto.

3.1.1 Caso linearmente separabile

Se il training set è linearmente separabile (come in Figura 5), possiamo considerare due iperpiani paralleli che separano i dati in due classi. Scegliamo questi iperpiani in modo tale che la distanza tra essi, ovvero il margine, sia massimizzata. Notiamo che in questa situazione l'iperpiano separatore a margine massimo è l'iperpiano che giace a metà tra i due iperpiani paralleli.

Assumendo che i dati del training set si trovino almeno a distanza 1 dall'iperpiano separatore, allora possiamo descrivere i due iperpiani paralleli nel seguente modo:

- $w^T x + b = +1$
- $w^T x + b = -1$

Notiamo di conseguenza tre cose:

- vogliamo assegnare l'etichetta +1 ad ogni punto x_t tale per cui $w^T x_t + b \geq 1$
- vogliamo assegnare l'etichetta -1 ad ogni punto x_t tale per cui $w^T x_t + b \leq -1$
- i vettori di supporto giaceranno proprio sui due iperpiani paralleli

Geometricamente (vedi Appendice A), la distanza tra i due iperpiani paralleli, cioè il margine, è pari a

$$\frac{2}{\|w\|} \quad (1)$$

Il nostro scopo è ovviamente quello di classificare correttamente i dati x_t , quindi dobbiamo considerare i seguenti due vincoli:

- $w^T x_t + b \geq +1$ se $y_t = +1$
- $w^T x_t + b \leq -1$ se $y_t = -1$

che possono essere riscritti così:

$$y_t(w^T x_t + b) \geq +1 \quad \forall t = 1, \dots, n \quad (2)$$

Mettendo insieme queste informazioni, capiamo che dobbiamo risolvere un problema di massimo vincolato. Al posto di massimizzare l'equazione (1), possiamo minimizzare $\|w\|$. Per comodità, decidiamo di minimizzare

$$\frac{\|w\|^2}{2}$$

sotto il vincolo (2).

Utilizziamo il metodo dei moltiplicatori di Lagrange, definendo come funzione obiettivo da minimizzare

$$L = \frac{\|w\|^2}{2} - \sum_{t=1}^n \alpha_t [y_t(w^T x_t + b) - 1]$$

dove (x_t, y_t) è il t-esimo esempio del training set e α_t è il moltiplicatore di Lagrange associato a tale esempio.

Calcoliamo la seguente derivata e poniamola uguale a zero per trovare il minimo del nostro problema:

$$\frac{\partial L}{\partial w} = w - \sum_{t=1}^n \alpha_t y_t x_t = 0$$

da cui calcoliamo che il migliore w per il nostro problema è il seguente:

$$w^* = \sum_{t=1}^n \alpha_t y_t x_t$$

dove α_t sarà sempre nullo tranne quando x_t è un vettore di supporto (ciò si dimostra risolvendo tramite programmazione quadratica il problema duale rispetto a quello considerato). Capiamo quindi che il decision boundary è determinato unicamente dai vettori di supporto.

Quando arriva in input un nuovo dato x_t da classificare, la sua etichetta sarà

$$y_t = \text{sign}(w^T x_t + b)$$

3.1.2 Caso non linearmente separabile

Consideriamo il caso in cui i dati non sono linearmente separabili, ma vogliamo applicare lo stesso un decision boundary lineare, dandogli la possibilità di commettere qualche errore nella classificazione (Figura 6). In questi casi si parla di margine "soft".

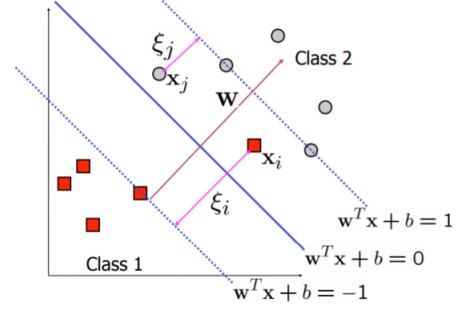


Fig. 6: Caso non linearmente separabile gestito tramite variabili slack.

La variabile slack ξ_t indica l'errore nella classificazione di x_t . A seconda del valore di ξ_t rispetto al dato x_t possono esserci tre casi possibili:

- se $\xi_t = 0$ la classificazione è corretta
- se $\xi_t = 1$ la classificazione è errata
- se $0 < \xi_t < 1$ allora la classificazione è corretta perchè x_t giace tra l'iperpiano separatore e l'iperpiano oltre il quale si trovano i punti che appartengono alla stessa classe di x_t .

I vincoli che consideriamo in questa situazione saranno

- $w^T x_t + b \geq +1 - \xi_t$ se $y_t = +1$
- $w^T x_t + b \leq -1 + \xi_t$ se $y_t = -1$

e possiamo sintetizzarli con una singola disuguaglianza:

$$y_t(w^T x_t + b) \geq 1 - \xi_t \quad (3)$$

dove $\xi_t \geq 0 \quad \forall t$. Dato che vogliamo minimizzare gli errori di classificazione, nel caso non linearmente separabile minimizzeremo il termine

$$\frac{\|w\|^2}{2} + C \sum_{t=1}^n \xi_t$$

sotto il vincolo (3).

La variabile C permette di controllare la penalizzazione per le classificazioni errate. Aumentando il valore di C , l'algoritmo tenderà durante l'addestramento ad evitare di classificare erroneamente i dati del training set. Di conseguenza, in questi casi la SVM tende a soffrire di overfitting, in quanto rischia di adattarsi troppo ai dati del training set.

3.1.3 Metodi kernel per decision boundary non lineari

Consideriamo ora il caso in cui vogliamo avere un decision boundary non lineare per risolvere i casi in cui i dati non sono linearmente separabili. L'idea è quella di proiettare i dati originari dello spazio di input in uno spazio a dimensioni maggiori (chiamato spazio delle feature) in cui diventano separabili linearmente (Figura 7). Tale proiezione viene fatta attraverso una funzione di mapping non lineare $\phi(\cdot)$.

Il problema di questo approccio è che nello spazio delle feature è molto costoso calcolare il prodotto scalare

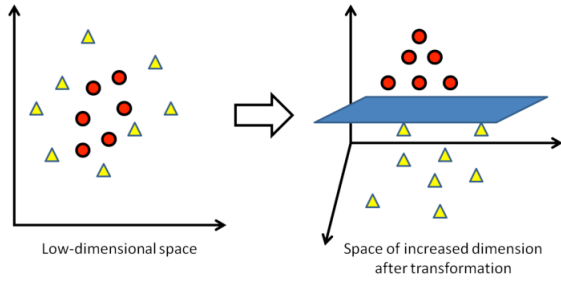


Fig. 7: Caso non linearmente separabile gestito tramite metodi kernel.

tra dati proiettati

$$\phi(x_p)^T \phi(x_q)$$

La soluzione è quindi quella di definire una funzione kernel

$$K(x_p, x_q) = \phi(x_p)^T \phi(x_q)$$

che permetta di calcolare il prodotto scalare tra due vettori dello spazio delle feature senza bisogno di conoscere e calcolare la funzione di mapping ϕ .

Una delle più utilizzate è la funzione kernel Radial Basis Function (RBF):

$$K(x_p, x_q) = \exp \left(- \frac{\|x_p - x_q\|^2}{2\sigma^2} \right)$$

3.2 Support Vector Machine per la Regressione (SVR) lineari

Nel caso della regressione, lavoriamo con un training set in cui le etichette $y_t \in \mathbb{R}$. Il nostro scopo è quello di trovare un regressore $f(x)$ che abbia un grado di precisione ϵ , cioè che non accetti durante l'addestramento nessun errore più grande di ϵ .

Il modello lineare è rappresentato dalla funzione

$$f(x) = w^T x + b$$

Nel caso della regressione, lo scopo è quello di minimizzare

$$\frac{\|w\|^2}{2}$$

sotto il vincolo che gli errori siano minori di ϵ :

$$|y_t - (w^T x_t + b)| \leq \epsilon$$

3.3 Principal Component Analysis (PCA)

La Principal Component Analysis (PCA) è una trasformazione lineare che genera in output tanti segnali, chiamati componenti principali, quante sono le variabili osservate passate in input. Ogni componente principale (PC) è una combinazione lineare delle variabili passate in input. I pesi di tale combinazione lineare sono scelti in modo da massimizzare la varianza della componente principale. Allo stesso tempo ogni PC viene calcolata in modo da risultare ortogonale rispetto a tutte le altre

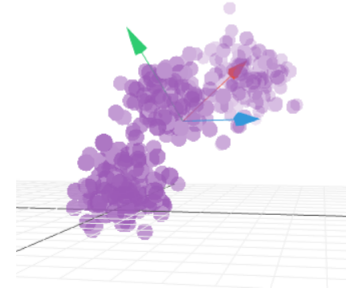


Fig. 8: Dati passati in input all'algoritmo PCA. I tre vettori colorati sono le tre componenti principali.

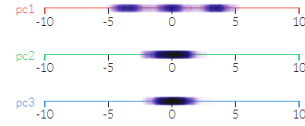


Fig. 9: Proiezione dei dati sulle tre componenti principali calcolate dall'algoritmo PCA.

PC. Di conseguenza, le componenti principali ottenute dall'algoritmo PCA sono tutte ortogonali tra loro. Questa tecnica ci permette di ridurre la dimensionalità dei dati, poichè possiamo scartare le componenti principali che presentano una varianza dei dati trascurabile per i nostri scopi.

Per capire meglio il concetto, vediamo da un punto di vista geometrico [9]. Nella Figura 8 sono mostrati i dati passati in input all'algoritmo PCA. I tre vettori colorati della Figura 8 sono le tre componenti principali calcolate dall'algoritmo PCA. Come possiamo notare, queste sono ortogonali tra loro. Nella Figura 9 notiamo inoltre che la prima componente principale (colore rosso) è quella che presenta la maggiore varianza dei dati. Questo è dovuto al fatto che ogni componente principale avrà varianza massima, ma allo stesso tempo deve rispettare il vincolo di ortogonalità con tutte le componenti principali che sono già state calcolate.

4 SIMULAZIONE ED ESPERIMENTI

4.1 Dataset

I dataset utilizzati per addestrare le SVM e le SVR sono tre:

- CK+
- UNBC
- DISFA+

In questo progetto sono state considerate solo le AU necessarie per classificare le seguenti emozioni di base: felicità, tristezza, sorpresa, rabbia, paura, disgusto e disprezzo. Quindi, nelle seguenti descrizioni dei dataset considereremo solo le AU relative a tali emozioni. Tali AU sono visibili nella tabella mostrata in Figura 2. Il dataset CK+ [10] contiene tutte le AU che permettono di classificare le emozioni di nostro interesse. Il dataset

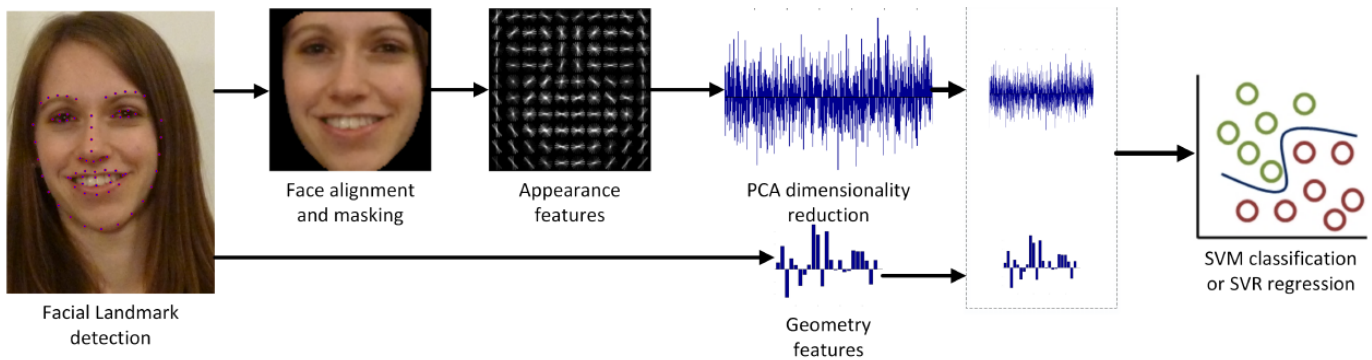


Fig. 10: Pipeline di operazioni applicate ad un'immagine di input, al fine di rilevare la presenza o l'intensità delle AU. L'immagine usata è presa da [2].

UNBC [11] non contiene tra le proprie etichette le seguenti AU: AU1, AU2, AU5, AU14, AU15, AU16 e AU23. Il dataset DISFA+ [12] non contiene tra le proprie etichette le seguenti AU: AU7, AU14, AU16 e AU23.

4.2 Architettura del Sistema

In questo lavoro è stata implementata una libreria Python che fornisce le seguenti funzioni:

- `get_img_aus_occurencies`: data un'immagine in input, restituisce la lista di AU rilevate all'interno di essa tramite l'utilizzo delle SVM
- `get_img_aus_intensities`: data un'immagine in input, restituisce un dizionario in cui ad ogni AU rilevata all'interno dell'immagine viene associata la relativa intensità stimata tramite l'utilizzo delle SVR
- `get_img_emotions`: data un'immagine in input, restituisce un dizionario in cui ad ognuna delle emozioni di base considerate in questo lavoro viene associata la percentuale di AU relative a tale emozione rilevate nell'immagine

Inoltre, è stato implementato lo script `get_au` che, dato come argomento il path di un'immagine, stampa a video le informazioni ottenute dalle 3 funzioni sopra descritte.

Analizziamo ora la sequenza di operazioni che vengono effettuate quando viene chiamata la funzione di libreria `get_img_aus_occurencies` (vedi Figura 10):

- innanzitutto viene rilevato il volto del soggetto presente all'interno dell'immagine
- dal volto, vengono estratti i landmark, che rappresentano le sue feature geometriche
- a partire dai landmark, è possibile allineare l'immagine in un sistema di riferimento comune per tutte le immagini considerate all'interno del progetto
- viene poi applicata una maschera con lo scopo di isolare la porzione del volto interna ai landmark dal resto dell'immagine. In realtà, tale maschera è applicata in modo da prelevare una porzione maggiore della fronte rispetto a quella che verrebbe considerata se usassimo solamente i riferimenti dei landmark

- da questa porzione del volto vengono estratte le feature HOG
- il vettore che contiene le feature HOG viene dato in input ad un modello PCA per ridurre la dimensionalità
- il vettore delle feature HOG ridotto dalla PCA viene concatenato al vettore che contiene la posizione dei diversi landmark
- questo vettore finale viene passato alle diverse SVM per il rilevamento della presenza delle AU

4.3 Dettagli Implementativi

Per rilevare i volti ed i relativi landmark all'interno delle immagini, sono state implementate le funzioni `get_face_locations` e `get_landmarks` (nello script `face_helpers`), basate sull'utilizzo della libreria Face Recognition [13]. Per estrarre la porzione del volto interna ai landmark, è stata utilizzata la funzione `fill_ConvexPoly` di OpenCV [14]. Per allineare i volti in un sistema di riferimento comune, è stata implementata la funzione `align_and_crop` (in `face_helpers`), adattata dal tutorial [15]. Al termine di questa operazione, ogni immagine viene trasformata in un'immagine di 112×112 pixel, che contiene solo la porzione del volto interna ai landmark (come si può vedere dall'immagine in Figura 10 etichettata con "Face alignment and masking"). Per estrarre da tali immagini le feature HOG, è stata implementata la funzione `get_hog_features` (in `face_helpers`), basata sull'utilizzo della libreria `scikit-image` [16].

In seguito, sono elencati i passi necessari per addestrare la SVM relativa alla AU27.

- Si scorrono le immagini presenti nei tre dataset a disposizione e le si dividono in due liste sulla base della presenza o meno della AU27. L'informazione sulla presenza della AU27 è ricavata dalle etichette presenti nei dataset stessi;
- per poter ottenere due liste bilanciate, si aggiungono esempi negativi solo se la lista degli esempi positivi è più grande di quella degli esempi negativi;
- consideriamo il caso in cui siamo riusciti ad estrarre 78 immagini che presentano la AU27. All'interno delle due liste, avremo un totale di 156 immagini

(78 per ognuna delle liste). A partire da queste, possiamo ottenere le seguenti strutture dati:

- una matrice 156x6084, che in ogni riga contiene le 6084 feature HOG estratte da una singola immagine
- una matrice 156x144, che in ogni riga contiene la posizione dei 72 landmark estratti da una singola immagine (quindi $72 \times 2 = 144$ valori per riga)
- una lista di 156 etichette, una per ogni immagine
- per ogni immagine, concateniamo il vettore delle feature HOG con quello delle posizioni dei landmark, ottenendo quindi una matrice 156x6228;
- dividiamo questa matrice in Training set e Test set, usando il 20% dei dati per il Test set;
- applichiamo la PCA alle feature HOG dei due set, effettuando il fitting del modello usando solo il Training set e mantenendo il 95% della variabilità espressa dai dati. In questo esempio, passiamo da un vettore di 6084 feature HOG ad uno di 103. A queste, concateniamo nuovamente i 144 valori relativi ai landmark. Otteniamo quindi
 - un Training set di forma (124, 247)
 - un Test set di forma (32, 247)
- prima di addestrare la SVM, scaliamo i dati dei due set usando lo StandardScaler della libreria sklearn [17];
- effettuiamo il tuning dell'iperparametro C della SVM tramite la tecnica cross-validation [18];
- infine addestriamo la SVM utilizzando i dati del training set, appoggiandoci nuovamente su sklearn [19]. Come funzione kernel è stata usata la RBF.

Come detto nella Sezione 4.2, prima di poter passare una nuova immagine ai modelli SVM per rilevare la presenza di Action Unit, abbiamo bisogno di applicarle la pipeline di operazioni mostrate nella Figura 10.

Per consentire una migliore comprensione del lavoro svolto, sono stati implementati quattro notebook con JupyterLab:

- 1) `face_helpers`: mostra l'implementazione delle funzioni necessarie ad applicare la pipeline di operazioni di Figura 10 ad un'immagine, fino all'estrazione delle feature HOG.
- 2) `svm_training`: mostra come vengono estratti i dati e le etichette dai dataset e come sono state addestrate le SVM.
- 3) `svr_training`: mostra come sono state addestrate le SVR.
- 4) `test_au_lib`: testa le funzioni implementate nello script "au_lib".

5 RISULTATI OTTENUTI

In generale, le performance peggiori sono state raggiunte dalle SVM e SVR relative alle AU non presenti nel dataset DISFA+: AU7, AU14, AU16 e AU23.

Quasi tutte le SVM hanno raggiunto un'accuratezza sul test set maggiore del 94%, a fronte di un'accuratezza sul training set sempre maggiore del 99.94%. Le eccezioni sono state le seguenti:

- AU5: test accuracy pari al 91.61%, a fronte di una training accuracy del 100%. Per l'addestramento sono stati utilizzati 5483 esempi estratti dai diversi dataset.
- AU7: test accuracy pari al 93.50%, a fronte di una training accuracy del 99.98%. Per l'addestramento sono stati utilizzati 5165 esempi estratti dai diversi dataset.
- AU16: test accuracy pari al 90%, a fronte di una training accuracy del 100%. Per l'addestramento sono stati utilizzati 36 esempi estratti dai diversi dataset.
- AU23: test accuracy pari al 79.17%, a fronte di una training accuracy del 100%. Per l'addestramento sono stati utilizzati 94 esempi estratti dai diversi dataset.

Le performance ottenute dalle SVR sono decisamente peggiori. Senza considerare le AU non presenti nel dataset DISFA+, il risultato migliore è stato ottenuto dalla SVR relativa alla AU9 con una test accuracy pari al 92%, a fronte di una training accuracy del 99.76%. Il risultato peggiore è stato invece ottenuto dalla SVR relativa alla AU20: test accuracy pari al 79.33%, a fronte di una training accuracy del 99.10%. Per quanto riguarda le AU non presenti nel dataset DISFA+:

- AU7: test accuracy pari al 78.11%, a fronte di una training accuracy del 99.98% (con 5165 esempi).
- AU14: test accuracy pari al 70.37%, a fronte di una training accuracy del 100% (con 59 esempi).
- AU16: test accuracy pari al 57.33%, a fronte di una training accuracy del 61.96% (con 36 esempi).
- AU23: test accuracy pari al 57.40%, a fronte di una training accuracy del 86.90% (con 94 esempi).

La maggior parte delle SVR soffre quindi di overfitting. Una possibile soluzione sarebbe quella di considerare anche gli altri dataset utilizzati da OpenFace, dato che già l'aggiunta del dataset DISFA+ ha permesso di migliorare notevolmente le performance di alcune SVR: ad esempio, la SVR relativa alla AU15 è passata da avere una test accuracy del 28.96% ad avere una test accuracy dell'86.55% (sempre a fronte di una training accuracy maggiore del 99.30%). Per cercare di ridurre il problema dell'overfitting, si sono considerate anche le seguenti possibilità:

- utilizzo di funzioni kernel diverse da quella RBF (ad esempio quella polinomiale)
- utilizzo di diverse forme di normalizzazione e standardizzazione dei dati
- validazione dell'iperparametro epsilon

La validazione dell'iperparametro epsilon è l'unica modifica che ha portato a dei miglioramenti delle performance per alcune SVR.

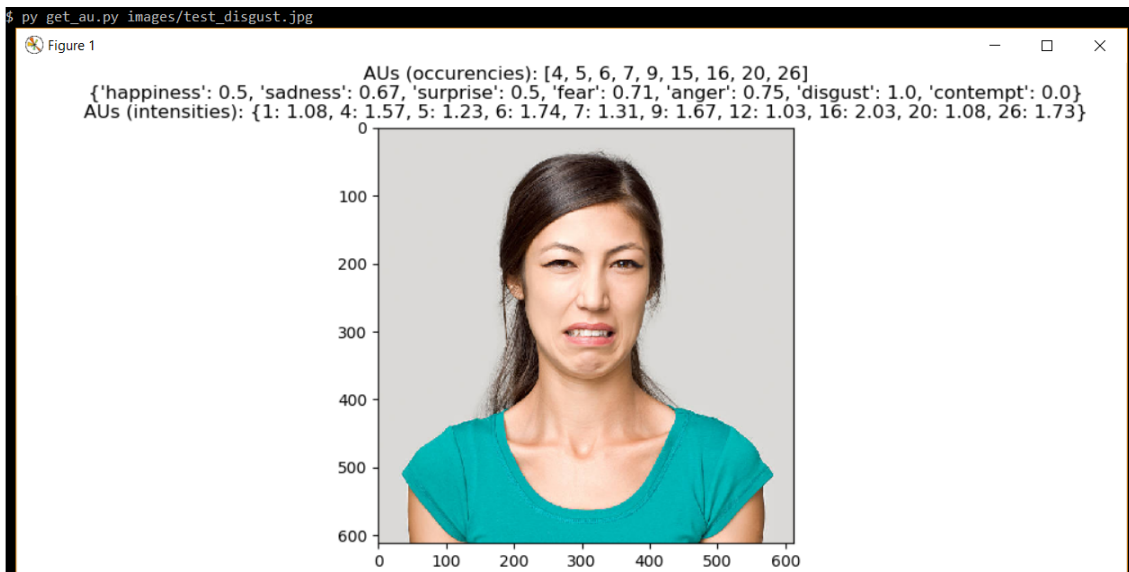


Fig. 11

In Figura 11, è mostrato un esempio di funzionamento dello script `get_au`. Le tre stringhe stampate sopra l'immagine sono l'output delle tre diverse funzioni implementate nella libreria `au_lib`, descritte nella Sezione 4.2.

6 COMMENTI CONCLUSIVI

In questo progetto sono state addestrate diverse SVM e SVR per il rilevamento della presenza e dell'intensità delle Action Unit che compaiono sul volto di un soggetto all'interno di un'immagine. Mentre le performance ottenute dalle SVM sono piuttosto buone, quelle delle SVR sono purtroppo soggette ad overfitting. Come detto nella Sezione 5, per migliorare le performance delle SVR si potrebbero considerare gli altri dataset usati in OpenFace:

- Bosphorus Database [20]
- BP4D [21]
- SEMAINE [22]
- GEMEP [23]

Dal punto di vista dell'aggiunta di nuove funzionalità, si potrebbe estendere questo lavoro implementando uno script a cui viene dato come argomento il path di un video e che genera in output un csv, in cui viene annotata la presenza delle AU con le rispettive intensità frame per frame. Si potrebbe anche far sì che questo script rilevi in tempo reale le AU presenti nei frame ripresi da una webcam.

APPENDIX

Dimostriamo che il margine di Figura 12 è pari a

$$\frac{2}{\|w\|}$$

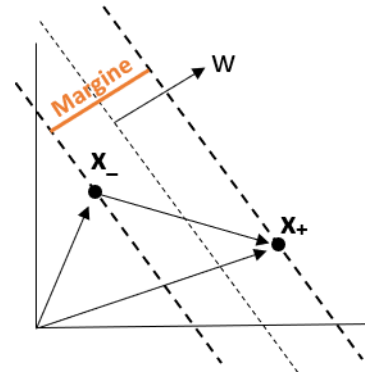


Fig. 12

Dalla figura, notiamo che i punti x_- e x_+ appartengono ai due iperpiani paralleli, descrivibili tramite le seguenti equazioni

- $w^T x + b = +1$
- $w^T x + b = -1$

Di conseguenza otteniamo che

$$w^T x_+ + b = +1 \rightarrow w^T x_+ = 1 - b \quad (4)$$

Analogamente

$$w^T x_- + b = -1 \rightarrow w^T x_- = -1 - b \quad (5)$$

Il margine può essere definito come

$$\frac{w^T}{\|w\|} (x_+ - x_-)$$

da cui, considerando (4) e (5), ricaviamo

$$\frac{w^T x_+ - w^T x_-}{\|w\|} = \frac{1 - b + 1 + b}{\|w\|} = \frac{2}{\|w\|}$$

REFERENCES

- [1] E. Friesen and P. Ekman, "Facial action coding system: a technique for the measurement of facial movement," *Palo Alto*, 1978.
- [2] T. Baltrušaitis, M. Mahmoud, and P. Robinson, "Cross-dataset learning and person-specific normalisation for automatic action unit detection," in *Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on*, vol. 6. IEEE, 2015, pp. 1–6.
- [3] Histogram of oriented gradients. [Online]. Available: <https://www.learnopencv.com/histogram-of-oriented-gradients/>
- [4] Computazione per l'interazione naturale: Richiami di ottimizzazione. [Online]. Available: http://boccignone.di.unimi.it/IN_2018_files/LezIN_Ottimizzazione_3_1.pdf
- [5] Support vector machines. [Online]. Available: <http://cesa-bianchi.di.unimi.it/MSA/Note/14-svm.pdf>
- [6] Teoria e tecniche del riconoscimento. [Online]. Available: <http://www.di.univr.it/documenti/OccorrenzaIns/matdid/matdid090730.pdf>
- [7] Support vector machines. [Online]. Available: <http://cs229.stanford.edu/notes/cs229-notes3.pdf>
- [8] Support vector machines. [Online]. Available: https://en.wikipedia.org/wiki/Support_vector_machine
- [9] Principal component analysis - explained visually. [Online]. Available: <http://setosa.io/ev/principal-component-analysis/>
- [10] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*. IEEE, 2010, pp. 94–101.
- [11] P. Lucey, J. F. Cohn, K. M. Prkachin, P. E. Solomon, and I. Matthews, "Painful data: The unbc-mcmaster shoulder pain expression archive database," in *Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*. IEEE, 2011, pp. 57–64.
- [12] M. Mavadati, P. Sanger, and M. H. Mahoor, "Extended disfa dataset: Investigating posed and spontaneous facial expressions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 1–8.
- [13] A. Geitgey, "Face recognition," https://github.com/ageitgey/face_recognition, 2018.
- [14] Opencv - drawing functions. [Online]. Available: https://docs.opencv.org/3.0-beta/modules/imgproc/doc/drawing_functions.html#fillconvexpoly
- [15] Face alignment with opencv and python. [Online]. Available: <https://www.pyimagesearch.com/2017/05/22/face-alignment-with-opencv-and-python/>
- [16] Scikit-image - histogram of oriented gradients. [Online]. Available: http://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html
- [17] Scikit-learn - preprocessing data. [Online]. Available: <https://scikit-learn.org/stable/modules/preprocessing.html>
- [18] Scikit-learn - gridsearchcv. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
- [19] Scikit-learn - svc. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>
- [20] A. Savran, N. Alyüz, H. Dibeklioglu, O. Çeliktutan, B. Gökberk, B. Sankur, and L. Akarun, "Bosphorus database for 3d face analysis," in *European Workshop on Biometrics and Identity Management*. Springer, 2008, pp. 47–56.
- [21] X. Zhang, L. Yin, J. F. Cohn, S. Canavan, M. Reale, A. Horowitz, P. Liu, and J. M. Girard, "Bp4d-spontaneous: a high-resolution spontaneous 3d dynamic facial expression database," *Image and Vision Computing*, vol. 32, no. 10, pp. 692–706, 2014.
- [22] G. McKeown, M. Valstar, R. Cowie, M. Pantic, and M. Schroder, "The semaine database: Annotated multimodal records of emotionally colored conversations between a person and a limited agent," *IEEE Transactions on Affective Computing*, vol. 3, no. 1, pp. 5–17, 2012.
- [23] Gemep-fera2011. [Online]. Available: <https://gemep-db.sspnet.eu/>