Image and Video Analysis: Sistema di riconoscimento automatico del livello di dolore percepito

Alessandro Arezzo
E-mail address
alessandro.arezzo@stud.unifi.it

Abstract

Il progetto descritto nel merito della presente relazione è stato realizzato come elaborato per l'insegnamento Image and Video Analysis previsto dal corso di laurea magistrale in ingegneria informatica dell'Università degli studi di Firenze.

1. Introduzione

Negli ultimi anni il task riguardante la stima automatica dell'indice di dolore percepito da un soggetto risulta essere al centro di un sempre maggior numero di lavori, venendo questo considerato come un obiettivo di estrema sensibilità. Risulta difatti ovvio come, ad esempio, un sistema di tale tipo possa consentire di comprendere i disturbi fisici patiti da persone che non sono in grado di dichiararne esplicitamente la presenza, come nel caso di bambini e pazienti con problemi neurolgici o di demenza. Cambiando contesto applicativo, di primario interesse è anche la predizione del dolore percipito dagli animali, le espressioni dei quali non ci consentono di valutarne il disagio con la stessa facilità rispetto a quanto possibile per gli esseri umani.

L'elaborato in oggetto alla presente relazione è stato realizzato nello specifico con l'obiettivo di definire un sistema capace di predire il livello di dolore provato da una persona per mezzo dell'analisi di un video che ne mostra l'espressione. Il progetto è stato implementato in linguaggio Python e prevede l'utilizzo di una parte del dataset *UNBC-McMaster* per l'apprendimento supervisionato dell'indice di dolore. Il dataset in questione, illustrato nel dettaglio nel corso del paragrafo 2, per ogni frame di ciascuna sequenza contenuta al suo interno, associa alla posizione dei landmarks facciali del soggetto raffigurato nel video il corrispondente indice *VAS (Visual Analog Scale)*.
Nel corso del progetto le sequenze sono state rappresentate utilizzando i *Fisher Vectors*, i quali hanno inoltre consentito di escludere da ogni video le configurazioni dei landmarks associabili all'espressione neutrale, non caratterizzanti cioè per la classificazione dell'indice VAS. Le sole componenti rilevanti estratte durante tale prima fase di clustering preliminare sono quindi state utilizzate per estrarre dei descrittori dalle sequenze, i quali sono poi stati posti in input per l'addestramento di un modello *SVRM (Support Vector Regression Machine)*, atto a predire l'indice di dolore percepito da un soggetto a partire dal descrittore della sequenza che lo ritrae.

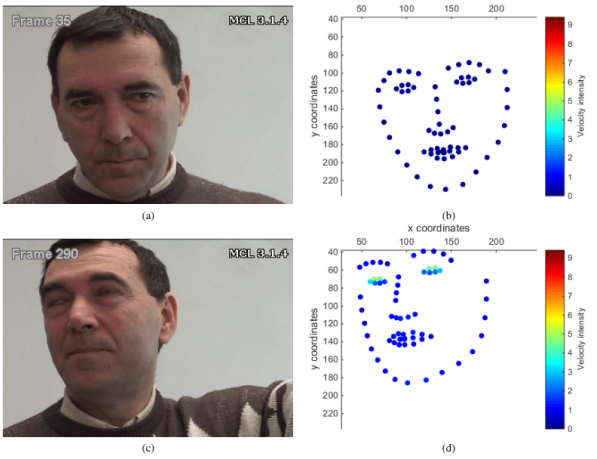

Figure 1. Predizione del dolore percepito da un soggetto attraverso l'analisi dei landmarks facciali che ne caratterizzano l'espressione.

Al termine dell'implementazione del framework, la cui struttura ed implementazione sono descritte rispetttivamente nel paragrafo 3 e 4, sono stati svolti dei tests al fine di valutare le performance dei modelli ricavati al variare del valore dei parametri caratterizzanti. In merito a ciò, all'interno del capitolo 5 sono riportati i risultati di tali esperimenti.

2. Dataset UNBC-McMaster Pain Archive

Ai fini dello svolgimento del progetto, è stata utilizzata l'ultima versione del dataset *UNBC-McMASTER Shoulder Pain Expression*, il quale contiene i dati di 200 sequenze

video associate a 25 soggetti differenti. Le sequenze che compongono l'insieme di dati sono a loro volta composte da un arbitrario numero di frames ed hanno associato un indice denominato *VAS (Visual Analog Scale)*, il quale indica il livello di dolore percepito dal soggetto rappresentato nel video in una scala compresa tra 0 e 10.
Ogni frame è caratterizzato dalle posizioni di 66 landmarks facciali estratti dal volto del soggetto, espresse per mezzo delle loro coordinate x ed y. Si noti infine come anche il numero di video associati ai soggetti risulti arbitrario.
Per una maggiore conoscenza del dataset in questione, questo è descritto nel dettaglio nell'articolo [2].

3. Framework per il riconoscimento automatico dell'indice VAS

L'architettura del sistema, del quale è riportato uno schema in figura 2, prevede la presenza di due fasi principali da eseguire in cascata l'una di seguito all'altra. Inizialmente, viene eseguita una prima parte dedita allo svolgimento di un'operazione di clustering preliminare, all'interno della quale sono in primo luogo estratte le features scalate atte a descrivere singolarmente ogni frame delle sequenze (paragrafi 3.1.1 e 3.1.2)). Tali descrittori vengono quindi successivamente clusterizzati utilizzando i Fisher Vectors, in modo da individuare così quelle configurazioni che essendo associate all'espressione neutrale non sono da considerarsi come caratterizzanti per la classificazione dell'indice VAS (paragrafi 3.1.3 e 3.1.4).
Utilizzando poi le sole configurazioni rilevanti vengono estratti i descrittori delle sequenze (paragrafo 3.2.1), i quali sono infine posti in input ad un modello SVRM che una volta addestrato si occupa di restituire l'indice VAS di una sequenza a partire dal vettore multidimensionale che la descrive (paragrafo 3.2.2).

Si noti inoltre come il dataset descritto nel paragrafo 2, venga diviso a priori in due parti applicando un qualche protocollo di validation, in modo da ottenere un insieme di sequenze da utilizzare per il training ed un altro per il testing (paragrafo 3.3). In tal modo, la suddetta procedura si applica considerando i soli dati di train ed il modello generato viene valutato per mezzo delle sole sequenze di test.

3.1. Clustering preliminare

3.1.1 Features dei frames

La prima operazione prevista dal framework riguarda l'estrazione delle features atte a caratterizzare ciascun frame delle sequenze appartenenti al training set. Nello specifico, ogni immagine che compone una certa sequenza viene descritta per mezzo di un vettore contenente le velocità di un arbitrario numero di landmarks stabiliti a priori. Sia quindi D il numero di punti facciali da considerare, si definisce $\lambda_{ij} = (a^n_{ij}, n = 1, .., D)$ il descrittore dell'i-esimo frame appartenente alla j-esima sequenza di train.
Per quanto concerne la velocità di un singolo landmark, questa viene calcolata come la variazione della sua posizione nel passaggio dal frame precedente a quello corrente, ovvero come:

$$a^n_{ij} = \sqrt{(x^n_{i-1j} - x^n_{ij})^2 + y^n_{i-1j} - y^n_{ij})^2} (1)$$

con x ed y coordinate dell'n-esimo landmark rispettivamente rispetto all'asse delle ascisse ed a quello delle ordinate.

Si noti, inoltre, come proprio le coordinate utilizzate in tale espressione siano in realtà il risultato di un'operazione dedita a centrarle rispetto alla posizione del naso estratta dal volto del soggetto raffigurato all'interno del video.

3.1.2 Scaling

Al fine di rendere le features estratte dai frames maggiormente robuste rispetto alla presenza di outliers che si possono venire a verificare a seguito della successiva operazione di clustering, queste vengono scalate a priori. L'operazione prevede nel dettaglio che a ciascuna feature (per ogni sua dimensione) venga sottratta la media dei dati presenti nel training set e che il risultato sia ridimensionato dividendo per la sua varianza.

3.1.3 Fisher Vectors

Per rappresentare ciascun frame a partire dal suo vettore di features si è scelto di utilizzare i *Fisher Vectors (FV)*. Tale codifica consente di catturare le differenze di primo e di secondo livello tra le caratteristiche locali ed un dizionario basato su *Gaussian Mixture Model (GMM)*, dove tali differenze sono nello specifico correlate alla deviazione dalle medie e dalle varianze dei kernels della GMM stessa.
In primo luogo, viene quindi eseguito il fitting di una GMM avente un arbitrario numero di kernels K sui vettori di features estratti dai frame appartenenti ad ogni sequenza del training set. La mistura di gaussiane ricavata sarà quindi caratterizzata da un vettore di parametri del tipo $\theta = \{\pi_k, m_k, \sigma_k, k = 1, .., K\}$ con π_k, m_k e σ_k che rappresentano rispettivamente la probabilità a priori, la media e la varianza della k-esima gaussiana.
La GMM associa poi ogni vettore di features λ_i dell'i-esimo frame appartenente all'insieme di train, a ciascuna gaussiana k, con un peso dato dalla probabilità a posteriori definita come:

$$q_{ik} = \frac{(\lambda_i - m_k)^T \sum_k^{-1} \lambda_i - m_k)}{\sum_{t=1}^{K} (\lambda_i - m_t)^T \sum_t^{-1} (\lambda_i - m_t)} \quad (2)$$

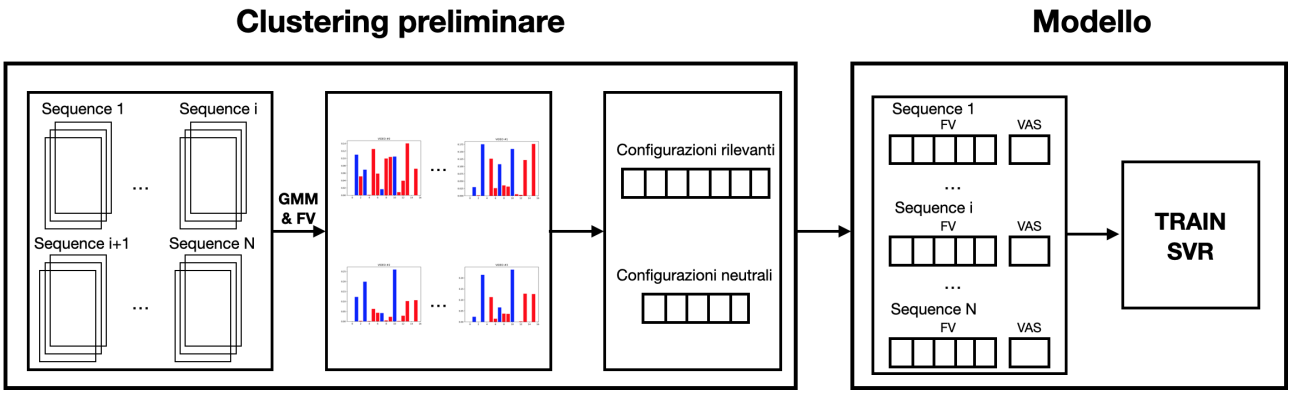

Figure 2. Schema del framework proposto

Il Fisher Vector associato ad un generico frame viene quindi definito concatenando la media (u_{nk}) e la varianza (v_{nk}) di ogni gaussiana k per $k = 1, .., K$, le quali sono calcolate come:

$$u_{nk} = \frac{1}{\sqrt{\pi_k}} q_{ik} \frac{a_{ij}^n - m_k}{\sigma_k} \tag{3}$$

$$v_{nk} = \frac{1}{\sqrt{2\pi_k}} q_{ik} [(\frac{a_{ij}^n - m_k}{\sigma_k})^2 - 1] \tag{4}$$

dove $n = 1, .., D$.
Il FV ottenuto viene infine normalizzato attraverso l'applicazione di una normalizzazione di tipo power-law seguita da una di tipo L2.

Ricapitolando quindi, l'i-esimo frame appartenente alla j-esima sequenza di train viene descritto con un FV indicato con FV_{ij}, un vettore multidimensionale avente dimensioni $2 \times D \times K$ dove D è il numero di landmarks considerati e K è il numero di kernels della GMM.
Per maggiori dettagli relativi alla codifica descritta, si rimanda al contenuto dell'articolo [1].

3.1.4 Estrazione delle configurazioni rilevanti

Sfruttando la rappresentazione dei frames per mezzo dei FV, è possibile ricavare quelle configurazioni associate all'espressione facciale neutrale che in quanto tali non sono in alcun modo rappresentative dell'indice di dolore percepito dal soggetto raffigurato nel video.
Per realizzare ciò, ogni sequenza di train viene innanzitutto rappresentata attraverso un istogramma atto ad indicare la ricorrenza delle configurazioni rilevate al suo interno. Consideriamo una sequenza di train j composta da N frames. L'istogramma H_j associato a tale sequenza sarà composto da K componenti (con K numero di kernels della GMM) ovvero sarà del tipo

$$H_j = (h_j^k, k = 1, .., K) \tag{5}$$

dove per $k = 1, .., K$ si ha che

$$h_j^k = \sum_{i=1}^{N} (sum_k(FV_{ij}) + sum_{k+K}(FV_{ij})) \tag{6}$$

con l'operatore $sum_k(FV_{ij})$ che rappresenta la somma degli elementi della riga k-esima del FV associato all'i-esimo frame della j-esima sequenza.
Fondamentalmente quindi, il valore della k-esima componente di un istogramma atto a descrivere una certa sequenza è dato dalla somma delle componenti delle righe associate a media e varianza del k-esimo kernel della GMM, nei FV che caratterizzano i frames appartenenti alla sequenza stessa. In tal modo, viene quindi effettuata un' operazione di soft clustering, attraverso la quale è possibile valutare quale sia la frequenza con la quale ciascuna delle configurazioni associate ai K kernels della GMM si presenta all'interno dei video del training set.

Una volta generati gli istogrammi seguendo tale metodologia, vengono selezionate come configurazioni associate all'espressione neutrale tutte quelle componenti che si verificano con una frequenza maggiore o uguale ad una **soglia prestabilita** in almeno una sequenza di train con associato indice VAS pari a 0. Tali sequenze, presentandosi difatti per un numero di volte sufficientemente elevato all'interno di almeno un video che raffigura un soggetto mentre questi non sta provando dolore, sono considerate essere semplicemente associabili a movimenti dovuti a microespressioni non correlate alla presenza di un disturbo fisico, e per questo possono essere scartate in seguito nel processo di classificazione dell'indice VAS. In figura 3 è riportato l'esempio di un istogramma in cui le componenti in blu sono quelle associate alle configurazioni delle features considerate neutrali, mentre quelle in rosso sono quelle correlate a velocità dei landmarks estratte come rilevanti. Si noti come sia estremamente importante il valore della soglia utilizzata per l'identificazione delle componenti neutrali, e

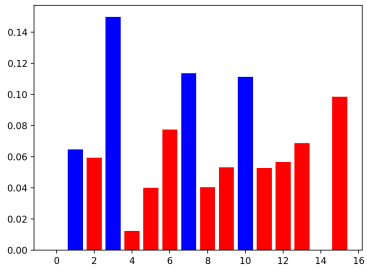

Figure 3. Esempio di un istogramma ricavato con un modello di 16 kernels della GMM. In rosso sono riportate le componenti estratte come rilevanti per la classificazione dell'indice VAS, mentre in blu quelle associate all'espressione neutrale.

di come perciò la variazione di tale parametro sia alla base della fase sperimentale svolta nel corso dell'elaborato, la cui analisi è riportata nel capitolo 5.

3.2. Modello SVR

3.2.1 Features delle sequenze

Una volta terminata la fase di clustering preliminare, le configurazioni delle features considerate rilevanti per la classificazione dell'indice VAS vengono in primo luogo utilizzate per generare i FV atti a descrivere interamente le sequenze a partire da quelli dei frames che le compongono. Nello specifico, il FV associato ad una generica sequenza j, che denotiamo come FV_j, è dato dalla somma delle righe correlate a media e varianza delle configurazioni rilevanti, dei FV associati ai frames che compongono la sequenza stessa. Posto quindi D il numero di landmarks considerati e P il numero di configurazioni rilevanti estratte durante il clustering preliminare, ciascuna sequenza viene descritta da un FV di dimensioni $2 \times D \times P$. A tali vettori multidimensionali viene poi applicata un' operazione di flatten, attraverso la quale ciascun video viene descritto per mezzo di un vettore unidimensionale di lunghezza pari a $2DP$.

3.2.2 Training

I descrittori delle sequenze di train vengono successivamente utilizzati per addestrare un *SVRM (Support Vector Regression Machine)* con kernel RBF.
Tale procedura di fitting è eseguita nello specifico in modo da ricavare i parametri corrispondenti a regolarizzazione, epsilon e gamma che minimizzano il mean absolute error applicando il protocollo di 5-fold-cross-validation sui dati di train.

Inoltre, dal momento che il dataset UNBC-McMaster non risulta bilanciato, a ciascun esempio viene attribuito per l'addestramento un peso inversamente proporzionale alla frequenza della classe associata al campione in questione all'interno del training set. Presa cioè una generica classe k (per $k = 0, .., 10$), ad ogni esempio con associato indice VAS pari a k viene assegnato un peso γ_k definito come

$$\gamma_k = \frac{N}{11 * occ_k} \tag{7}$$

dove N è il numero di esempi del training set, occ_k quello dei campioni di train con associata classe k ed 11 sono le classi del dataset.

3.3. Valutazione delle performance

Il modello SVR, addestrato come descritto nel sottoparagrafo 3.2.2, consente di predire l'indice VAS associato ad una certa sequenza a partire dal FV che la descrive.
Nel processo di inferenza quindi, per ogni sequenza appartenente al test set, viene generato il vettore multidimensionale che la caratterizza esattamente come già visto per le sequenze di train. Vengono cioè calcolati i FV dei frames che la compongono e successivamente questi sono sommati nelle sole componenti associate alle configurazioni rilevanti delineate dal clustering preliminare per generare il descrittore della sequenza stessa. I FV dei video di test sono quindi posti in input al metodo di predizione del modello, il quale ne restituisce l'indice VAS espresso come un valore decimale.

La bontà del modello, viene quindi in primo luogo valutata calcolando il **mean absolute error (mse)** sulle sequenze di test. A seguito di ogni predizione, il risultato viene mappato all'interno dell'intervallo compreso tra 0 e 10, per poi essere arrotondato al numero intero più vicino. Il mse medio è infine calcolato come la media dei singoli errori ricavati su ogni sequenza di test.
Oltre a ciò viene anche prodotta in output una **matrice di confusione** atta a descrivere chiaramente le prestazioni del modello durante la predizione delle varie classi. Una matrice di confusione M è cioè composta da 11 righe e da altrettante colonne (una per ogni classe) ed un suo generico elemento m_{ij} indica la quantità di volte in cui il modello predice un esempio di classe i con la classe j. La matrice in questione, una volta calcolata viene poi normalizzata. Idealmente quindi, un modello che predice i dati perfettamente ha tutti elementi pari ad 1 sulla diagonale e valori nulli al di fuori di essa mentre, viceversa, più i valori tendono a concentrarsi al di fuori della diagonale e peggiori saranno le sue performance.
Infine per valutare le prestazioni di un modello, viene utilizzata nel progetto anche una matrice di confusione che rappresenta le predizioni di tre classi corrispondenti a livelli di dolore nullo, debole ed elevato. Per il calcolo di tale matrice, gli indici VAS vengono mappati sulle suddette classi di modo che gli esempi con VAS pari a 0 siano considerati come aventi associato dolore nullo, quelli con VAS compreso nell'intervallo tra 1 e 3 debole ed infine sono mappati

nella classe associata ad un dolore forte i campioni con valore predetto tra 4 e 10.

4. Implementazione

Il framework descritto nel paragrafo 3 è stato implementato in linguaggio di programmazione Python. Il progetto si articola su due script denominati **PreliminaryClustering.py** e **ModelSVR.py**, i quali hanno lo scopo di implementare, come meglio descritto di seguito, rispettivamente la fase atta all'estrazione delle configurazioni rilevanti e quella relativa alla gestione della SVRM.

4.1. PreliminaryClustering.py

All'interno dello script PreliminaryClustering.py, è contenuta la classe PreliminaryClustering, definita ai fini dell'implementazione della fase di clustering preliminare illustrata a livello teorico nel corso del paragrafo 3.1.
Tale classe, riceve in input al suo costruttore in primo luogo informazioni riguardanti il dataset da utilizzare, come il percorso dei files all'interno dei quali questo è contenuto ed il numero di landmarks che sono utilizzati per descrivere ogni frame. Inoltre, il modulo richiede gli indici delle sequenze del dataset da utilizzare per il train, una maschera contenente gli indici dei landmarks da considerare per l'analisi, e delle informazioni necessarie per la definizione della GMM, quali il numero dei suoi kernels ed il tipo della matrice diagonale (che può essere di tipo full oppure diagonale).
La classe espone poi un metodo pubblico principale, il quale una volta ricevuta come parametro la soglia da utilizzare per identificare le configurazioni neutrali (vedi paragrafo 3.1.4), esegue in cascata i seguenti metodi privati:

- **__get_velocities_frames():** estrae le features da ogni frame, dove ciascun descrittore è rappresentato dalla velocità dei landmarks aventi indici stabiliti nell' apposita maschera posta in input alla classe.
- **__scale_features():** applica lo scaling alle features estratte dal metodo precedente utilizzando la classe RobustScaler interna alla libreria *sklearn*.
- **__prepare_training_features():** esegue un preprocessing dei dati in modo da restituire una collezione contenente le features di tutti i frame di train raccolti in una sola struttura dati. Ciò è necessario dal momento che la libreria fishervector, utilizzata per l'addestramento della GMM, richiederebbe altrimenti in input sequenze aventi tutti eguale lunghezza, mentre nel dataset in uso il numero di frame per ogni video è arbitrario.
- **__generate_gmm():** utilizza la libreria fishervector per eseguire il fitting di una GMM avente numero di kernels definiti in input alla classe sui dati di train generati dal metodo precedente.
- **__calculate_FV():** calcola i FV atti a descrivere ogni frame delle sequenze del dataset, utilizzando la GMM appena addestrata.
- **__generate_histograms():** genera gli istogrammi di ogni sequenza del dataset, come descritto nel corso del paragrafo 3.1.4.
- **__extract_relevant_and_neutral_configurations():** estrae gli indici delle configurazioni da considerarsi come rilevanti per la classificazione dell'indice VAS, e li memorizza all'interno di un apposito attributo della classe.

Infine, il metodo descritto riceve in input anche un valore booleano che indica se salvare o meno le immagini degli istogrammi prodotti. In tal caso questi vengono memorizzati in file .png enfatizzando la differenza tra le configurazioni considerate neutrali e quelle associate a componenti rilevanti, rappresentando le prime in blu e le seconde in rosso.
Si noti poi di come i parametri maggiormente caratterizzanti ed in grado di influire sul risultato dell'analisi siano il numero di kernels della GMM ed il valore da usare come soglia per rilevare le configurazioni neutrali. In relazione a ciò, occorre specificare come sia possibile passare alla classe anche due liste della stessa dimensione, una con i numeri di kernels da considerare ed un'altra contenente un insieme di soglie, specificando inoltre un parametro booleano denominato fit_by_BIC con valore True. Così facendo il metodo di addestramento della GMM, utilizza il numero di kernels che, tra quelli specificati, minimizza il valore della metrica *Bayesian information metric (BIC)*. La soglia neutrale utilizzata in questo caso sarà quindi estratta dalla lista di soglie passata alla classe, e posizionata al suo interno all'indice corrispondente a quello del numero di kernels scelto nella rispettiva lista.

4.2. ModelSVR.py

Completata l'operazione di clustering preliminare, l'oggetto della classe PreliminaryClustering, preposto allo svolgimento di tale analisi, viene posto in ingresso al costruttore di ModelSVR, classe interna al file ModelSVR.py.
Tale modulo, consente nello specifico di gestire il modello SVRM. Per farlo questo riceve in input, oltre al risultato del clustering preliminare ed alle informazioni riguardanti la posizione e la composizione del dataset, gli indici delle sequenze di train e di test ed un parametro booleano che stabilisce se gli esempi debbano essere pesati (vedi paragrafo 3.2.2). La classe implementa poi i tre seguenti metodi pubblici principali:

- **train_SVR():** estrae i descrittori delle sequenze di training come descritto nel paragrafo 3.2.1 ed addestra

un modello SVR con kernel RBF utilizzando il modulo svm.SVC interno alla libreria *sklearn*. Salva quindi il modello generato all'interno di un attributo di classe, così da potervi accedere in futuro per il calcolo delle sue performance. Si noti come il metodo riceva anche un valore booleano che stabilisce se il processo di addestramento deve essere eseguito scegliendo il valore dei parametri di regolarizzazione, gamma ed epsilon che minimizzano il mse su un insieme di validation. In tal caso ciò viene svolto utilizzando la classe GridSearchCV del modulo *sklearn.model_selection*. In caso contrario, è possibile passare esplicitamente al metodo il valore di tali parametri.

- **evaluate_performance_model():** applica il modello generato al fine di predire l'indice VAS delle sequenze del test set. Restituisce quindi al chiamante il mean absolute error rilevato e la relativa matrice di confusione. Passando degli appositi percorsi in input al metodo, è anche possibile salvare i risultati di ogni predizione all'interno di un file csv e l'immagine rappresentante la matrice di confusione in un file .png. Si noti, come l' absolute error di ogni esempio venga calcolato come illustrato nel corso del paragrafo 3.3.
- **evaluate_performance_on_scaled_pain():** applica il modello generato per predire gli indici VAS delle sequenze del test set per poi mapparli su tre classi, rispettivamente corrispondenti a livello di dolore nullo (VAS=0), debole (VAS da 1 a 3) e dolore consistente (VAS da 4 a 10). Restituisce quindi una matrice di confusione 3×3 che ne riassume i risultati.

5. Tests e risultati

Al fine di svolgere dei tests atti a verificare le performance dei modelli ricavati sono stati definiti due scripts. Il primo di questi, denominato **test.py**, è stato implementato allo scopo di testare il comportamento del modello al variare del valore utilizzato come soglia per le configurazioni neutrali. Fissato cioè il numero di kernels da utilizzare per la GMM ed una lista di soglie da testare, il modello viene generato come descritto in precedenza per ciascuna delle soglie presenti all'interno di essa. La procedura di valutazione delle performance, nel dettaglio, prevede in primo luogo che le sequenze del dataset vengano divise sulla base di uno dei tre protocolli di validation supportati, ovvero quello di Leave-One-Subject-Out, 5-fold-cross-validation e Leave-One-Sequence-Out. Col primo, ad ogni round vengono selezionate le sequenze di un certo soggetto per il testing, mentre quelle di tutti i rimanenti 24 sono usate per il training. Al round successivo viene scelto un altro soggetto per il testing e la procedura viene così iterata fino a quando tutti i soggetti non sono stati usati per valutare le performance. Il protocollo di 5-fold-cross-validation è simile a quello appena descritto, con la differenza che ad ogni fase vengono scelte le sequenze di 5 soggetti per il test, anzichè di uno solo. Infine Leave-One-Sequence-Out prevede che ad ogni round venga selezionata una sola sequenza di test mentre tutte le restanti 199 sono utilizzate per il training del modello. Ovviamente, il numero di round generati dipende dal protocollo in uso, ed essendo il dataset composto da 200 sequenze appartenenti a 25 soggetti, questo corrisponde rispettivamente a 20, 5 e 200.
Per ogni soglia definita all'interno della lista contenente quelle da testare quindi, il modello viene addestrato per un numero di volte pari a quello dei round generati dal protocollo di validation. Una volta eseguita la procedura di training per un certo round, le performance del modello vengono valutate calcolando il mse e la matrice di confusione delle predizioni sugli esempi di test. Una volta che ciò è stato fatto per tutti i round, le performance della soglia testata vengono riassunte per mezzo di un valore calcolato come la media dei singoli mse rilevati ad ogni round e di una matrice di confusione complessiva normalizzata data dalla somma delle matrici di confusione relative. Inoltre, per ogni soglia viene calcolato il numero di configurazioni rilevanti estratte in media. Quando ciò è stato eseguito per tutte le soglie di test vengono stampati due grafici che riassumono l'andamento del numero di configurazioni rilevanti e del mean absolute error in funzione della soglia utilizzata. Lo script in questione, può anche essere eseguito in modalità fit_by_BIC. In questo caso viene definita a priori una lista di kernels anzichè un solo valore e ad ogni round viene utilizzato il kernel che minimizza il BIC sui dati di train. Al termine dell'intera procedura, per ogni numero di kernels della lista che è stato selezionato almeno una volta durante il processo, viene generato un grafico che ne illustra l'andamento in termini di mse rilevato al variare della soglia usata.

L'altro script implementato, denominato **generate_model_predictor.py**, consente invece di valutare le performance di un modello fissato sia il numero di kernels della GMM che la soglia da usare per l'estrazione delle configurazioni neutrali. L'analisi che viene effettuata da tale modulo ha lo scopo di concentrarsi maggiormente su tutti gli aspetti del modello a differenza di quanto avviene con lo script test.py. Al termine della sua esecuzione difatti, oltre alla matrice di confusione complessiva, viene salvato un grafico che illustra i mse rilevati ad ogni round, uno che mappa i centri dei cluster della GMM su 2 dimensioni e le matrici di confusione relative estratte durante il processo. Inoltre, viene salvata una matrice di confusione ricavata mappando i valori predetti su tre classi atte a rappresentare rispettivamente sequenze con associato livello di dolore nullo, basso ed alto (vedi paragrafo 3.3).
Si noti infine come anche tale script, al pari di quello precedentemente illustrato, possa essere eseguito in modalità

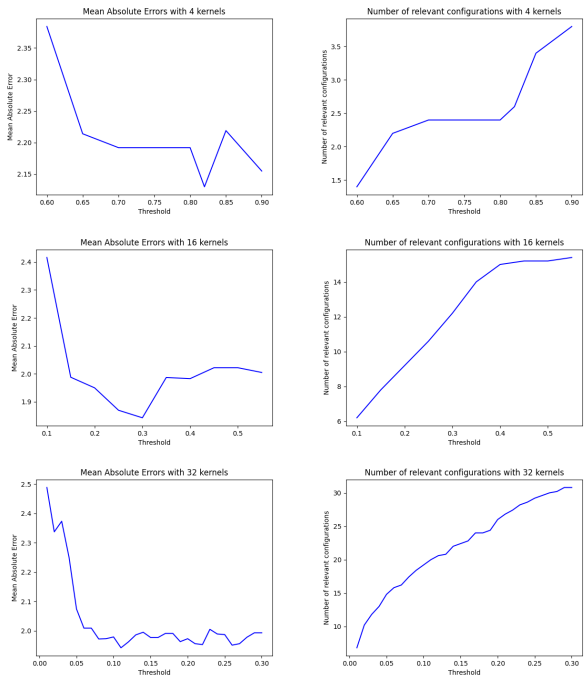

Figure 4. Andamento del mean absolute error (sulla sinistra) e del numero di configurazioni rilevanti estratte (sulla destra) in funzione della soglia usata rispettivamente per GMM con 4, 16 e 32 kernels.

fit_by_BIC.

5.1. Esperimenti

I due moduli descritti sono quindi stati utilizzati per realizzare una serie di tests in grado di valutare le performance dei modelli. Tutti gli esperimenti sono stati effettuati considerando solo 12 dei 66 landmarks che caratterizzano i frames nel dataset. Si sono cioè nello specifico considerati solo quelli relativi agli occhi ed alla bocca (in entrambi i casi sia per la loro parte superiore che inferiore), al mento ed alle sopracciglia. Tutti gli altri landmarks non sono stati valutati come rilevanti ai fini della classificazione dell'indice di dolore percepito dai soggetti.

Considerando tali landmarks quindi, si sono eseguite due fasi di esperimenti. La prima, ha coinvolto il testing di modelli ricavati con un numero di kernels della GMM fissato, mentre la seconda si è concentrata sull'analisi delle prestazioni ricavate con una strategia di scelta di tale valore minimizzando il BIC sulle features.

Per quanto riguarda la prima parte questa è stata a sua volta divisa in due blocchi. In primo luogo, si è scelto di testare modelli ricavati rispettivamente con 4, 16 e 32 kernels della GMM, e per ciascuno di tali valori è stato eseguito il modulo test.py utilizzando il protocollo 5-fold-cross-validation. Grazie a tale esecuzione, per ognuno dei suddetti numeri di cluster, è stato possibile ricavare quale fosse la soglia in grado di minimizzare il mse rilevato in media sui 5 round. Applicando quindi tali soglie, è stato eseguito il modulo generate_model_predictor.py usando tutti i tre protocolli di validation implementati. In tal modo, per ogni numero di kernels testato, è stato possibile analizzare nel dettaglio le prestazioni dei modelli ricavati nella loro configurazione migliore.

I grafici riportati in figura 4 mostrano i risultati prodotti durante la prima fase dell'esperimento. Osservando l'andamento degli errori in funzione della soglia usata si nota innanzitutto come la curva tenda a decrescere fino al raggiungimento di un punto di minimo, per poi risalire oppure rimanere stazionaria attorno al valore della funzione in quest'ultimo. Ciò è dovuto al fatto che utilizzando una soglia piccola, come si nota dai grafici delle configurazioni rilevanti estratte, la maggior parte dei cluster viene rilevata come presente con una frequenza maggiore della soglia stessa in almeno una sequenza con indice VAS nullo. In conseguenza di ciò, solo pochissime componenti vengono considerate rilevanti per la classificazione e questo causa che le sequenze vengano descritte per mezzo di FV con poche righe, non sufficientemente significative per caratterizzare in alcun modo l'indice di dolore associatovi. Al contrario, utilizzando una soglia troppo elevata, saranno troppe le configurazioni considerate importanti e di conseguenza, non venendo scartate quelle correlate all'espressione neutrale, i FV conteranno informazioni ridondanti. Ciò causa anche in questo caso un errore rilevato maggiore. Il giusto compromesso si raggiunge proprio nel citato punto di minimo, la cui soglia corrispondente consente di estrarre solo i cluster effettivamente caratterizzanti tralasciando gli altri. Si nota che per una GMM con 4 kernels il minor mse si ottiene con 3 componenti estratte in media sui 5 round, mentre per i test con 16 e 32 clusters il risultato migliore si ha rispettivamente con 12 e 20 configurazioni considerate rilevanti.

Come detto in precedenza, utilizzando per ciascuna delle tre configurazioni le soglie che consentono di ottenere i migliori risultati, è stato applicato lo script generate_model_predictor.py usando inizialmente il protocollo di 5-fold-cross-validation. Le performance sono riassunte per mezzo delle matrici di confusione ricavate, le quali sono riportate in figura 5. Osservando quelle che tra queste riassumono il comportamento del modello evidenziando le predizioni di ciascun indice VAS, si nota come le loro componenti tendano a popolarle lungo la diagonale ma al tempo stesso presentano numerosi valori al di fuori di essa. Dall'analisi delle matrici di confusione dei livelli mappati nelle classi associate a livelli di dolore nulli, deboli e forti, si evince invece quanto i modelli si comportino bene in generale nel predire gli ultimi due tra questi. La principale problematica che emerge risiede altresì nella difficoltà

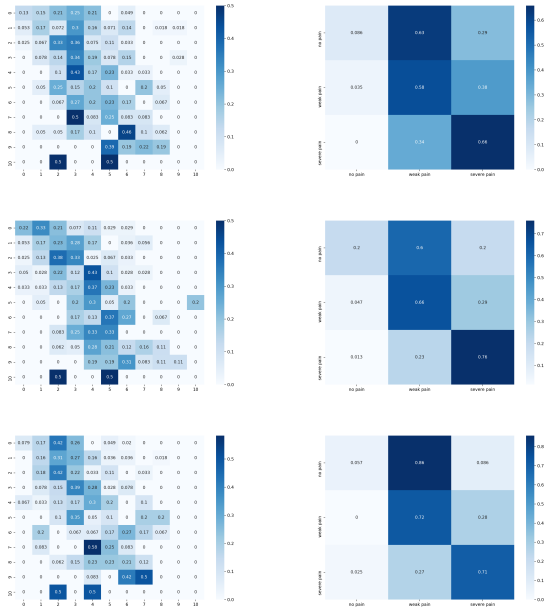

Figure 5. Matrici di confusione ricavate utilizzando la configurazione che minimizza il mse sui 5 round del protocollo di 5-fold-cross-validation. Alla prima riga sono mostrati i risultati del modello con 4 kernels per la GMM, mentre alla seconda ed alla terza quelli rispettivamente con 16 e 32 clusters.

di contraddistinguere le sequenze con indice VAS nullo, le quali vengono spesso associate ad un dolore debole.
Tra le tre tipologie testate, la migliore sembra essere quella con 16 kernels. Difatti, l'mse nel caso migliore, con tale configurazione, risulta essere solo di poco superiore a quello ottenuto dal modello con 32 kernels, a discapito di risultati qualitativi però migliori. Ciò si poteva dedurre anche dalla semplice analisi dei grafici degli mse riportati in figura 4, poichè nel caso di 32 kernels, una volta raggiunto il punto di minimo, aumentando la soglia e dunque le configurazioni rilevanti estratte, l'errore tende a rimanere stazionario, mentre nel modello a 16 clusters questo aumenta. Ciò indica che con quest'ultima configurazione la soglia che minimizza l'errore consente effettivamente di trarre benefici dall'esclusione di componenti ininfluenti ai fini della classificazione dell'indice VAS.
Utilizzando quindi la configurazione con 16 kernels ed una soglia che minimizza l'errore del modello, si è valutato quest'ultimo anche applicando i protocolli di validation Leave-One-Subject-Out e Leave-One-Sequence-Out. Come si nota dai risultati prodotti, i quali sono rappresentati dalle matrici di confusione riportate in figura 6, le performance del modello sono del tutto simili a quelle ottenute con il protocollo di 5-fold-cross-validation.

Una volta terminata la fase appena descritta, si è eseguito un'altra parte di esperimenti atta a valutare le perfor-

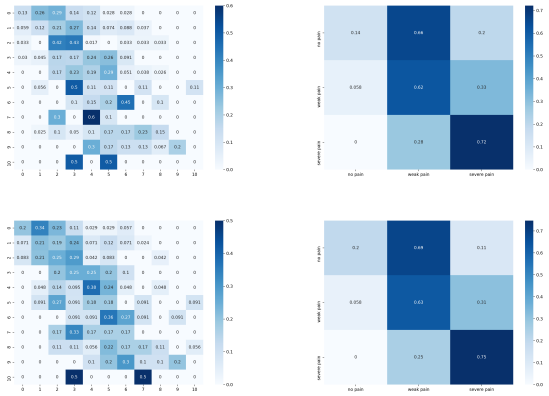

Figure 6. Matrici di confusione ricavate con la configurazione che minimizza il mse nel modello con 16 kernels della GMM applicando i protocolli di validation Leave-One-Subject-Out (prima riga) e Leave-One-Sequence out (seconda riga).

mance di modelli generati scegliendo il numero di kernels della GMM che minimizza il *Bayesian information metric (BIC)* sui vettori di features scalati dei frames. Nel dettaglio, si è selezionata una lista di kernels nell'intervallo compreso tra 12 e 32. Eseguendo alcune prove, è emerso che nella maggior parte dei casi, vengono scelte GMM con 26, 28 e 32 clusters, ed in conseguenza di ciò, si è eseguito il modulo test.py con protocollo di 5-fold-cross-validation considerando tali valori. Ricavati quindi, per mezzo di tale esecuzione, i valori delle soglie che per ciascuna delle tre configurazioni minimizza l'mse, si è valutato il modello applicando tutti i 3 protocolli di validation implementati.

Le matrici di confusione ricavate, riportate in figura 7, denotano come in realtà l'approccio descritto non consenta di migliorare qualitativamente le prestazioni rispetto al modello con 16 clusters, anche se occorre notare che l'mse ricavato in questo caso è risultato di poco inferiore rispetto a quello ottenuto con l'altro modello citato.

6. Conclusioni e sviluppi futuri

L'elaborato discusso nel merito della presente relazione consiste in un'implementazione in linguaggio di programmazione Python di un sistema in grado di predire il livello di dolore percepito da un soggetto raffigurato all'interno di un video analizzandone il movimento dei landmarks facciali.
L'analisi dei risultati prodotti ha denotato che, nella migliore configurazione possibile, sebbene la predizione degli specifici indici di dolore VAS non si possa ritenere così accurata, i dolori di debole ed elevata intensità sono in generale correttamente classificati dai modelli generati applicando il framework proposto. La problematica principale riguarda invece la capacità di predire l'indice VAS di sequenze con associato un dolore nullo, le quali vengono

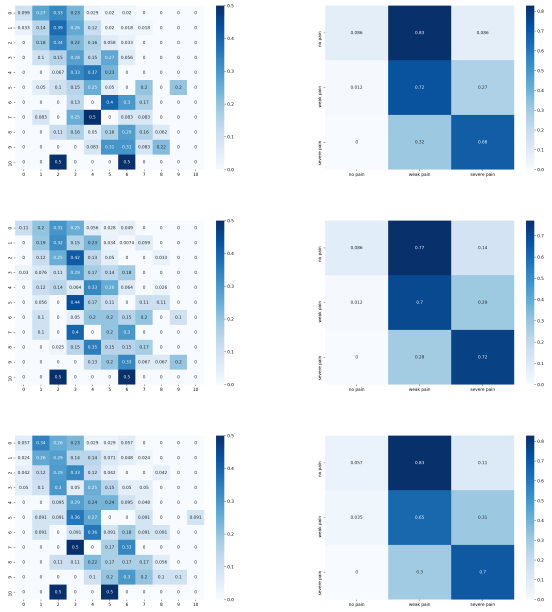

Figure 7. Matrici di confusione ricavate con la migliore configurazione del modello avente numero di kernels della GMM scelto minimizzando il BIC sulle features scalate tra 26, 28 e 32 clusters ed applicando i protocolli di validation 5-fold-cross-validation (prima riga), Leave-One-Subject-Out (seconda riga) e Leave-One-Sequence-Out (terza riga).

altresì spesso associate ad un disturbo fisico di bassa intensità.
Futuri lavori quindi potrebbero estendere quanto qua illustrato al fine di migliorare proprio la qualità nella predizione di video raffiguranti il volto di soggetti mentre questi non stanno provando alcun dolore.

References

[1] R. G. Abhinav Dhall. A temporally piece-wise fisher vector approach for depression analysis.

[2] K. M. P. P. E. S. I. M. Patrick Lucey, Jeffrey F. Cohn. Painful data: The unbc-mcmaster shoulder pain expression archive database, 2010.