Progetto T2
IUM 2 - Anno accademico 2012/2013

Marco Bottin 748881

Abstract—Nel presente documento verrá illustrato l'interfacciamento e l'utilizzo dei classificatori **K-NearestNeighbor e naïveBayes** di PMTK3 [1] nell'ambito del ChaLearn Gesture Challenge (riconoscimento e classificazione di gesti acquisiti da Kinect).

Gli input utilizzati, sono video RGB registrati con Kinect e forniti direttamente da ChaLearn mentre la feature utilizzata come base per la classificazione é la *"Static posture history"* presente in GROP, tale feature produce una immagine contenente una sorta di "scia" del movimento. Un esempio di output di tale feature, é rappresentato in Figura 1.

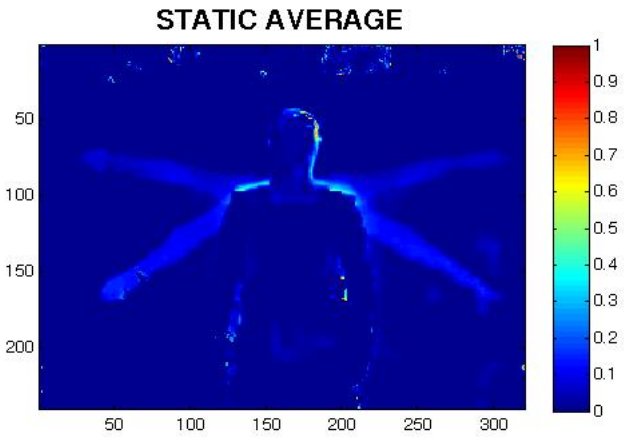

Fig. 1: Esempio feature "Static posture history"

L'algoritmo utilizzato per la classificazione tramite i classificatori Knn e naïveBayes, lavora in modo soddisfacente su input contenenti un solo gesto. Qualora l'input contenga piu gesti, i due classificatori forniscono risultati differenti e in alcuni casi errati, in alcuni casi di test, si riscontra che il classificatore restituisce il gesto che é presente con maggiore frequenza (numero di volte) all'intenro del video.

Nel paragrafo 5 verrano infine analizzati i risultati ottenuti dalle prove di learning e classificazione tramite i due classificatori tratti da PMTK3, con i risultati ottenuti tramite uno dei classificatori forniti direttamente da GROP, ovvero il *"basic_recog"*.

Marco Bottin, Corso di Interazione Uomo Macchina 2, A/A 2012-2013, Universitá di Milano, via Comelico 39/41, Milano, Italy
E-mail: marco.bottin@studenti.unimi.it

1 Introduzione

Il ChaLearn Gesture Challenge affronta il problema della classificazione e riconoscimento di movimenti gestuali, catturati da dispositivi hardware quali ad esempio telecamere, webcam o dispositivi simili quali il Kinect. L'apprendimento avviene sulla base del *"One Shot Learning"* ovvero la sottomissione di un solo esempio per il training di un particolare gesto da riconoscere e classificare.

ChaLearn mette a disposizione una serie di funzioni Matlab (raggruppate nello strumento GROP) contenente diversi estrattori di features ed alcuni classificatori di gesti.

Lo scopo di questo progetto é quello sviluppare una procedura MatLab che attraverso la selezione ed estrazione di una feature dai video campione, sia in grado di apprendere e classificare i gesti, attraverso l'utilizzo dei classificatori Knn e naïveBayes.

Verranno quindi messi a confronto i risultati ottenuti dai nuovi classificatori, con i risultati ottenuti dai classificatori giá presenti in GROP e messi a disposizione dell'utente.

Importanza del problema: La classificazione e il riconiscimento di gesti attraverso dispositivi hardware, sta diventando sempre di piú di uso comune. Alcune applicazioni di questa interazione naturale si possono trovare ad esempio in campo ludico (vedi Xbox, PlatyStation e Nintendo Wii) ed altre ancora in dispositivi quali ad esempio le nuove SmartTV di Samsung, nelle quali é possibile interagire con il televisore, senza l'ausilio del telecomando. Una webcam integrata é in grado di riconoscere i gesti prodotti dall'utente, consentendo ad esempio funzionalitá come "cambio canale" o "regolazione volume".

Approccio seguito: Dal dataset di ChaLearn, sono stati selezionati alcuni video campione per creare un training set ed un test set ad hoc per il progetto. In particolare per il training set sono stati selezionati 5 gesti campione, e per ogni classe vengono forniti uno o piú esempi di training. Per quanto riguarda il test set sono stati selezionati diversi video contenenti sia gesti singoli che gesti multipli, sempre appartenenti alle classi prese in esame dal training set.

La procedura Matlab sviluppata per il progetto, estrae dai due Data Set la feature *"Static posture history"* di

GROP che verrá utilizzata dal classificatore Knn per il riconoscimento del gesto eseguito nel video in esame. I risultati ottenuti, verranno successivamente analizzati e raffrontati ai risultati ottenuti utilizzando uno dei classificatori giá presenti in GROP

Contributi: La procedura Matlab sviluppata per il progetto, mette a confronto i risultati del classificatore *"basic_recog"* fornito da GROP, con i risultati ottenuti interfacciando alcuni classificatori tratti da PMTK3 (Knn e naïveBayes). Come vedremo nel seguito, i due classificatori di PMTK3 forniscono risultati soddisfacenti nel caso in cui il video da classificare contenga un solo gesto e nel caso in cui vengano forniti piu esempi di training per la classe di interesse. Inoltre la classificazione sará tanto piú precisa quanto piú il movimento eseguito sará simile all'originale. Nel caso in cui nel video siano presenti piu gesti, i due classificatori tendono a dare risultati differenti e talvolta imprecisi, in alcuni casi viene fonito in risposta il gesto che in "media" viene eseguito piú volte oppure, nel caso peggiore, un gesto completamente diverso da quello eseguito.

2 Analisi dello stato dell'arte

Esistono in letteratura diversi lavori che affrotnano il problema del riconoscimento gestuale, dal Chalearn Gesture Challenge, utilizzato come fonte principale per lo sviluppo del progetto, a testi piú generali che trattano il tema del *Machine Learning*, come ad esempio il libro di K. Murphy sul machine learning [2] utilizzato come fonte per alcuni concetti presentati nel paragrafo inerente il modello teorico. Lo stesso autore ha contribuito allo sviluppo del toolbox PMTK3 [1] dal quale sono stati estratti ed utilizzati gli script inerenti ai classificatori oggetto del progetto.

3 Modello teorico

In questo paragrafo verranno presentati gli aspetti formali dei classificatori PMTK3 utilizzati, ovvero:

- Classificatore K-NearestNeighbor (k-NN);
- Classificatore naïveBayes;

3.1 Classificatore K-NearestNeighbor

Il primo classificatore utilizzato nel progetto, é il K-NearestNeighbor (k-NN). k-NN viene utilizzato nel riconoscimento e la classificazione di oggetti, basandosi sulle caratteristiche degli oggetti "vicini" a quello considerato. Tale classificatore é di tipo non-probabilistico discriminativo, ovvero la classificazione avviene per mezzo di una funzione $f : X \rightarrow Y$ discriminante che mappa ogni input x in una classe C_i, senza tenere conto della distribuzione di probabilitá associata alle classe stessa.

L'insieme

$$\mathcal{D} = \{(x_1, c_1), (x_2, c_2)..., (x_n, c_n)\}$$

viene detto ***training set*** e contiene i *prototipi* che descrivono le classi, nella forma $(training\_sample, class)$. Utilizzando piú prototipi per la stessa classe C_i, si rende il riconoscimento e la classificazione della classe piú precisa.

Il classificatore Knn "cerca" i K punti nel training set che sono piú "vicini" (distanza euclidea sulle features) al valore x_i di test, successivamente ragruppa i K punti piú vicini per classe, e restituisce la classe che contiene il maggior numero di elementi. Nel caso in cui non ci sia una classe in particolare contenente il maggior numero di elementi K, viene fatta una scelta random della classe di appartenenza del dato di test.

La figura 2 mostra un esempio di classificazione con $K = 3$ in uno spazio 2d

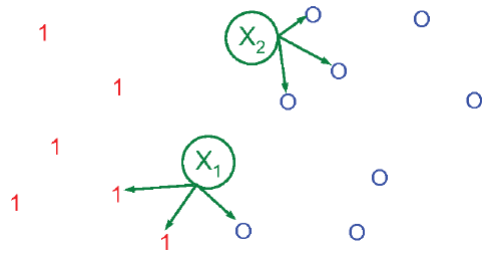

Fig. 2: Esempio di classificatore K-nearest neighbors in uno spazio 2d con K = 3. I 3 elementi piú vicini al dato di test x_1 hanno rispettivamente le label 1, 1 e 0, quindi la probabilitá $p(y = 1|x1, D, K = 3) = 2/3$. Mentre per x_2 le labels sono 0, 0, e 0 quindi $p(y = 1|x2, D, K = 3) = 0/3$.

3.2 Training del modello con K-NearestNeighbor

Come abbiamo specificato nel paragrafo precedente, K-NN é un classificatore di tipo discriminativo non probabilistico. Non dovendo quindi calcolare i parametri del modello per la generazione delle probabilitá $P(X|C)$ e $P(C)$ (rispettivamente la probabilitá di ottenere il gesto X conoscendo la classe C e la probabilitá delle classi C), la fase di training si limita ad associare semplicemete le labels delle classi ai campioni di training.

3.3 Classificazione con K-NearestNeighbor

La classificazione di un nuovo elemento appartenete al TestSet, avviene scegliendo l'etichetta del prototipo x' a minima distanza da x. Gli oggetti appartenenti al TrainingSet e TestSet, sono la rappresentazione vettoriale della matrice contenente l'immagine prodotta dalla feature *"Static posture history"*. Come abbiamo visto nella figura 1, la *"Static posture history"* produce un'immagine che viene memorizzata in MatLab in una matrice di dimensione 240 x 320. la vettorizzazione consiste nel concatenare le 240 righe della matrice in un unica riga, producendo in output un vettore di dimensione 1 x 76800

Il calcolo della distanza vettoriale avviene utilizzando la *distanza euclidea* espressa formalmente dall'equazione 1

$$\delta(x,y) = \sqrt{\sum_{d=1}^{d}(x_d - y_d)^2} \tag{1}$$

La figura 3 mostra un esempio di classificazione utilizzando come campioni di train e test le immagine di "Static posture history".

Come si puó vedere, l'elemento t_1 é molto piú simile al training example x_1. Il classificatore quindi associerá x_1 a t_1 con maggiore probabilitá

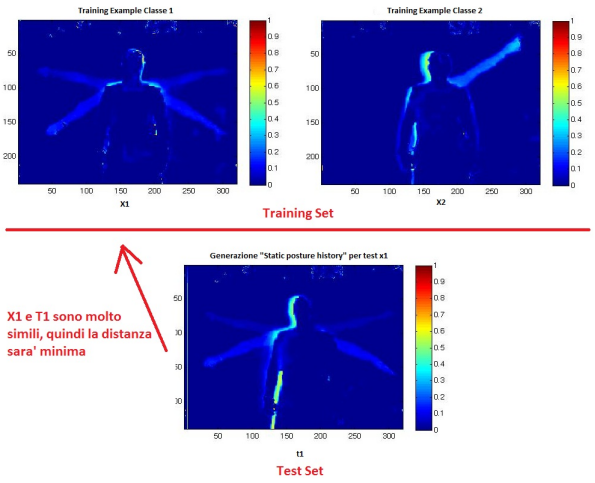

Fig. 3: Esempio di classificazione considerando la feature *"Static posture history"* di GROP. il classificatore associerá t_1 alla classe x_1

3.4 Classificatore naïveBayes [3]

Il secondo classificatore utilizzato é il classificatore naïveBayes, di tipo generativo probabilistico. In tale classificatore viene derivato, per ogni possibile outut, un modello (sotto forma di di distribuzione di probabilitá) degli elementi associati a quell'output.

A partire dal TrainingSet viene derivata la probabilitá congiunta $p(x, c|T)$ per avere una descrizione completa della situazione, nella forma

$$p(x, y|T) = p(y|x, T)p(x|T)$$

che attraverso la regola di Bayes ci consente di calcolare la probabilitá a posteriori associata alla classe C dato il gesto X tramite l'equazione 2

$$p(y|x, T) = \frac{p(x, y|T)}{p(x|T)} \tag{2}$$

Un approccio completamente bayesiano richiederebbe un numero di training sample molto elevato, per poter calcolare in modo accurato la probalitá condizionata $P(X|C)$ e $P(C)$. (nel nostro contesto, la probabilitá di estrarre il video X dato il gesto C)

Il classificatore naïveBayes riduce la complessitá dell'approccio Bayesiano assumendo che le feature siano condizionalmente indipendenti data la classe c_i.

3.5 Indipendenza Condizionale

Definizione: Date le variabili casuali X, Y e Z diciamo che X é **condizionalmente indipendente** da Y data Z se e solo se la distribuzione di probabilitá che governa X é indipendente dal valore assunto da Y dato Z; come espresso dalla equazione 3

$$(\forall i, j, k)P(x_i|y_j, z_k) = P(x_i|z_k) \tag{3}$$

Se per esempio l'insieme delle features contenesse solo due elementi e quindi

$$X =< X_1, X_2 >$$

la probabilitá condizionata $P(X|C)$ sarebbe:

$$\begin{aligned} P(X|C) &= P(X_1, X_2|C) && (4)\\ &= P(X_1|X_2, C)P(X_2|C) && (5)\\ &= P(X_1|C)P(X_2|C) && (6) \end{aligned}$$

dove l'ultima riga dell'equazione rispecchia la definizione di indipendenza condizionale definita in 3. Piú in generale, quando X contiene n attributi che sono condizionalmente indipendenti, dato Y, é possibile calcolare la probabilitá condizionata della classe, tramite l'equazione 7

$$P(x_1...x_n|C_k) = p(x|C_k) = \prod_i p(x_i|C_k) \tag{7}$$

dove X é il vettore delle features del video e c_i é la classe di appartenenza (il gesto)

L'espressione che definisce la probabilitá di ottenere la classe c_k dato un nuovo caso di test $X_{new} =< x_1, ..., x_n >$, per il teorema di Bayes é:

$$P(C = c_k|X_1...X_n) = \frac{P(C = c_k)P(X_1, ..., X_n|C = c_k)}{\sum_j P(C = c_j)P(X_1, ..., X_n|C = c_j)} \tag{8}$$

Sfruttando quindi l'equazione 7 nell'equazione del teorema di Bayes 8, possiamo calcolare la probabilitá che il video X appartenga al gesto C tramite l'equazione 9

$$P(C = c_k|X = x_i) = \frac{P(C = c_k)\prod_i P(X_i|C = c_k)}{\sum_j P(C = c_j)\prod_i P(X_i|C = c_j)} \tag{9}$$

Essendo interessati, nella fase di classificazione di un nuovo campione di test, a calcolare solo la classe c_i piú probabile che descrive il gesto presente nel video x_i, e non la probabilitá di tutte le classi presenti nel training set e considerando che il denominatore dell'equazione 9 non dipende da c_k la regola di classificazione puó essere espressa dalla equazione 13

$$C \leftarrow \arg\max_{y_k} P(C = c_k)\prod_i P(X_i|C = c_k) \tag{10}$$

3.6 Training e classificazione naïveBayes

La fase di learning consiste nello stimare i parametri del modello relativi alle probabilitá $P(X|C)$ e $P(C)$ a partire dai dati del training set, mentre la fase di classificazione consiste nel calcolare, tramite la regola del teorema di Bayes la probabilitá $P(Y|X_{new})$ espressa formalmente nell'equazione 13

Sulla base dei concetti espressi nel paragrafo precedente, i parametri da stimare nella fase di learning sono i seguenti:

$$\theta_{ij} \equiv P(X = x_i|C = c_j) \quad (11)$$

$$\pi_j \equiv P(C = c_j) \quad (12)$$

mentre la regola di classificazione per un nuovo campione di test, é giá stata formalizzata nell'equazione 13 e che riportiamo qui di seguito per completezza

$$C \leftarrow \arg\max_{y_k} P(C = c_k) \prod_i P(X_i|C = c_k) \quad (13)$$

L'algoritmo qui di seguito proposto, mostra lo pseudo codice della fase di training del modello

Algorithm 1 Algoritmo di Training per naïveBayes

for each value* c_k **do**
  estimate $\pi_k \equiv P(C = c_k)$
  for each value* x_{ij} of each attribute X_i **do**
    estimate $\theta_{ijk} \equiv P(X_i = x_{ij}|C = c_k)$

* la somma dei parametri deve dare 1

3.7 Calcolo dei parametri del modello naïveBayes

Possiamo stimare i parametri θ_{ijk} e π_k usando sia una stima MLE (Maximum likelihood Estimation, basata sul calcolo delle frequenze dei singoli eventi presenti nel Training Set) che una stima MAP (Maximum a Posteriori, (utilizzando una distribuzione a priori sui valori dei parametri).

Il calcolo della MLE per il parametro θ_{ijk}, dato il TrainingSet D é data dalla seguente equazione:

$$\widehat{\theta}_{ijk} = \widehat{P}(X_i = x_{ij}|C = c_k) = \frac{\#D\{X_i = x_{ij} \wedge C = c_k\}}{\#D\{C = c_k\}} \quad (14)$$

Dove l'operatore $\#D\{x\}$ restituisce il numero di elementi nel TrainingSet D che soddisfano la propietá x.

Analogamente la MLE per il parametro π_k é dato da:

$$\widehat{\pi}_k = \widehat{P}(C = c_k) = \frac{\#D\{C = c_k\}}{|D|} \quad (15)$$

Dove $|D|$ indica il numero di elementi presenti nel TrainingSet D

Utilizzando la MLE nella stima dei parametri, si puó incappare nel problema dell'overfitting. Ad esempio se non sono presenti dati nel TrainingSet che soddisfano la condizione del numeratore, la stima di θ e di π sará pari a zero. Per mitigare il problema dell'overfitting si potrebbe utilizzare una stima MAP con un termine di regolarizzazione l. La stima dei parametri θ e di π utilizzando una MAP con termine di regolarizzazione diventa quindi:

$$\widehat{\theta}_{ijk} = \widehat{P}(X_i = x_{ij}|C = c_k) = \frac{\#D\{X_i = x_{ij} \wedge C = c_k\} + l}{\#D\{C = c_k\} + lJ} \quad (16)$$

dove l é il termine di regolarizzazione (che dipende dalla distribuzione di probabilitá che si vuole utilizzare sui parametri θ_{ijk}) e J é il numero di valori distinti che X_i puó assumere. Il termine lJ indica il numero di esempi che provocano "rumore" (ovvero il numero di esempi per il quale il numeratore, con la stima MLE potrebbe dare zero).

Analogamente la stima di $\widehat{\pi}_k$ diventa:

$$\widehat{\pi}_k = \widehat{P}(C = c_k) = \frac{\#D\{C = c_k\} + l}{|D| + lK} \quad (17)$$

Dove K indica il numero di valori distinti che puó assumere C

4 Simulazione e esperimenti

I video selezionati come campioni del TestSet vengono sottoposti alla procedura MatLab per la classificazione tramite K-nn e naïveBayes. La procedura estrae per ogni video di test, la feature *"Static posture history"* che verrá quindi confrontata dal classificatore con tutti i dati di Training. In output verrá quindi fornita la classe "predetta" da ognuno dei classificatori utilizzati.

4.1 Dataset

Il dataset utilizzato nel progetto, é stato creato sulla base del dataset del ChaLearn Gesture Challenge (scaricabile all'indirizzo http://gesture.chalearn.org/data). Partendo dai video preregistrati, é stato creato un training set contenente 5 classi (sono stati selezionati 5 gesti dai video messi a disposizione) e ad ogni classe é stato assegnato uno o piú video di training. Analogamente il test set é stato creato selezionando dal dataset di ChaLearn i soli video contenenti i gesti presenti nel Training Set. Non tutti i video del Test Set contengono singoli gesti da classificare, infatti alcuni elementi possono contenere gesti multipli.

La tabella 1 mostra la corrispondenza dei video utilizzati nel progetto con i video presenti nel dataset di ChaLearn.

4.2 Architettura del sistema

Il diagramma raffigurato in figura 4 rappresenta lo schema concettuale dell'architettura del sistema. Il sistema puó essere scomposto nei seguenti macroblocchi:

- Generazione Training Set e relative classi;

TABLE 1: Corrispondenza video Dataset ChaLearn - Dataset Progetto

ChaLearn Video	Class#	Training#	Test#	Multi Gesto
valid02_4	1	1	-	N
valid02_15	1	2	-	N
valid02_3	2	3	-	N
valid02_44	2	4	-	N
valid04_3	3	5	-	N
valid04_16	3	6	-	N
valid09_11	4	7	-	N
valid09_14	4	8	-	N
valid18_2	5	9	-	N
valid18_28	5	10	-	N
valid02_26	1	-	1	N
valid02_40	1	-	2	Y
valid02_46	2	-	3	N
valid04_17	3	-	4	Y
valid04_22	3	-	5	Y
valid09_16	4	-	6	Y
valid09_39	4	-	7	N
valid18_40	5	-	8	N
valid18_45	5	-	9	N

  - Estrazione della feature *"Static posture history"* dai video di training ;
  - Associazione del video alla classe di appartenenza sulla base della feautre estratta ;
- Elaborazione video del Test Set ed estrazione della feature *"Static posture history"* ;
- Applicazione del modello per il riconoscimento e classificazione del gesto sulla feature estratta dal Test Set
- Restituzione dei risultati all'utente

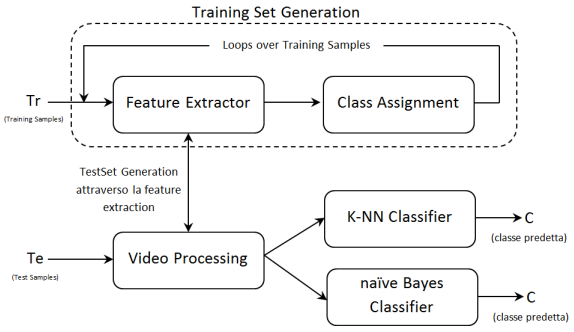

Fig. 4: Architettura funzionale del sistema

4.3 Dettagli implementativi

La procedura MatLab sviluppata per il progetto fa uso dei seguenti toolbox:

- Matlab GROP fornito da ChaLearn (scaricabile all'indirizzo http://gesture.chalearn.org/data);
- PMTK3 di Kevin Murphy - Classificatori KNN e naïveBayes ;

GROP é stato utilizzato per l'elaborazione dei video campione, ed in particolare per l'estrazione della feature da utilizzare per la classificazione. Nella sezione Appendix verranno dati i dettagli implementativi relativi alle funzioni utilizzate

PMTK3 é stato invece utilizzato per gli script che implementano le funzioni di *fit* e *predict* dei due classificatori. Nel progetto sono stati utilizzati gli script originali forniti dal toolbox

Nel paragrafo successivo, verrá descritta la logica applicativa sotto forma di pseudo-codice

4.3.1 *logica applicativa - pseudo-codice*

Algorithm 2 generazione training set e classificazione Knn/naïveBayes

for $i = 1$ to $max\_number\_of\_trainings$ **do**
  $Tr_i \leftarrow read\_video(i)$
  $Feature \leftarrow motion\_trail(V0)$
  $Feature \leftarrow vettorizza(V0)$
  $Xtrain[i] \leftarrow Feature$
$ytrain \leftarrow [1, 2, 3, 4, 5]$
for $i = 1$ to $max\_number\_of\_tests$ **do**
  $Te_i \leftarrow read\_video(i)$
  $Feature \leftarrow motion\_trail(T0)$
  $Feature \leftarrow vettorizza(T0)$
  $Knn\_class \leftarrow Knn\_Predict(Feature)$
for $i = 1$ to $max\_number\_of\_tests$ **do**
  $Te_i \leftarrow read\_video(i)$
  $Feature \leftarrow motion\_trail(T0)$
  $Feature \leftarrow vettorizza(T0)$
  $naiveBayes\_class \leftarrow naiveBayes\_Predict(Feature)$
$print\_results$

5 RISULTATI OTTENUTI

Prima di presentare i risultati ottenuti dai classificatori utilizzati nella procedura MatLab, occorre dare qualche dettaglio in piú sul funzionamento e comportamento del classificatore GROP "basic_recog" utilizzato per il confronto. Tale classificatore, divide il video in segmenti di lunghezza L e ogni segmento, viene elaborato e confrontato con segmenti derivanti dai video di training. A differenza dei classificatori K-nn e naïveBayes utilizzati, il classificatore di GROP esegue la fase di learning utilizzando un solo campione di training per classe. La classificazione, anche in questo caso, avviene utilizzando un approccio KNN, ovvero viene restituita la classe piú simile al segmento considerato.

A differenza dei classificatori K-NN e naïveBayes, il "basic_recog" é in grado di riconoscere piú gesti all'interno dello stesso video, in virtú del fatto che il video viene "spezzettato" in segmenti ed ogni segmento viene classificato separatamente.

La tabella 2 mostra i risultati ottenuti dall'elaborazione dei dati con i diversi classificatori. Il simbolo "???" indica la presenza di un gesto che non appartiene a nessuna classe conosciuta.

É facile notare che tutti i classificatori presentano anomalie nel riconoscere la classe "3" (**Test#04 e Test#05**). Solo il classificatore GROP é in grado di restituire, almeno per un caso di test, un risultato molto

TABLE 2: Risultati classificazione

Test#	K-nn Class	naïveBayes Class	GROP Class	Truth Class
Test#01	1	1	1	1
Test#02	1	5	1 1 ??? 1	1 1 ??? 1
Test#03	2	2	2	2
Test#04	1	1	3	??? 3
Test#05	2	5	1	3 ???
Test#06	4	4	4 4	4 ???
Test#07	4	4	4	4
Test#08	5	5	5	5
Test#09	5	5	5	5

vicino a quello reale, mentre per tutte le altre classi, i classificatori si comportano in modo pressoché corretto.

La figura 5 mostra l'immagine prodotta dal motion_trail per il campione di training della classe "3" e il motion_trail generato dal campione di test #04. Le immagini pur sembrando simili, presentano delle grosse differenze nella parte del tronco del soggetto, causando quindi errori nella classificazione.

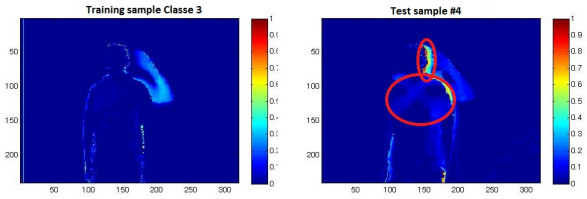

Fig. 5: Motion Trail generato per il training sample della classe 3 e per il sample test 4. i cerchi rossi mostrano le forti differenze nelle immagini che portano a risultati errati

Aumentando i campioni del training set, per la classe "3", i classificatori si comportano in modo piú corretto restituendo almeno per un campione di test, la classe corretta.

La figura 6 mostra l'output della procedura, aggiungendo 2 campioni di training per la classe "3". Come si puó vedere, il classificatore Knn riconosce in modo corretto per il Test#4 la classe 3

```
Training numero 01 predetto con Knn: 1
Training numero 02 predetto con Knn: 1
Training numero 03 predetto con Knn: 2
Training numero 04 predetto con Knn: 3
Training numero 05 predetto con Knn: 2
Training numero 06 predetto con Knn: 4
Training numero 07 predetto con Knn: 4
Training numero 08 predetto con Knn: 5
Training numero 09 predetto con Knn: 5
```

Fig. 6: il Test#04 riconosce in modo corretto la classe di appartenenza

GROP risulta decisdamente preciso nel riconoscere e classificare il **Test#02** nel quale il gesto della classe "1" viene ripetuto "3" volte, intervallato da un gesto sconosciuto. É interessante notare come GROP sia riuscito a classificare in modo corretto un caso di test cosí complesso con il solo ausilio di un campione di training. Anche K-nn fornisce una risposta valida, restituendo la classe che compare piú volte nel campione di test (la classe "1"). La precisione della classificazione di questo caso di test, deriva dal fatto che lo stesso gesto viene ripetuto piú volte nel video, questo comportamento genera quindi una "scia" nel "motion_trail" piú marcata e piú facilmente riconoscibile.

Altro caso interessante é il **Test#06** nel quale tutti i classificatori restituiscono la classe "4" in modo corretto. GROP é in grado di riconoscere il "multi gesto" anche se associa il secondo gesto del campione di test, alla classe "4" in modo errato

6 Commenti conclusivi

In conclusione, possiamo affermare che i classificatori utilizzati nel progetto, sono stati in grado di rispondere in modo soddisfacente ai requisiti di classificaizone del progetto. L'accuratezza e la precisione della clasificazione, cresce con il numero di campioni di train che vengono forniti per ciascuna classe, cosí come illustrato in fugura 6

Lo strumento GROP fornito da ChaLearn, é molto piú preciso e completo rispetto ai classificatori interfacciati nel progetto e lasciano quindi i seguenti problemi aperti, che possono essere approfonditi in futuro:

- classificazione multi gesto;
- Miglioramento della classificazione con un numero limitato di training samples (one shot learning);

Appendix
Note su Algoritmo 1

Qui di seguito viene riportato uno spezzone di codice relativo alla classificazione tramite Knn. Il codice presenta gli statement eseguiti per l'estrazione della feature (tramite motion_trail), la vettorizzazione della matrice e il fitting e predizione tramite Knn.

```
Xtest=[];
for i=1:9

 %Leggo l'i-esimo video
 M0=read_movie([test_dir '/K_'num2str(i)'.avi']);

 %motion_trail restituisce l'immagine contenente
 %la "scia" del movimendo eseguito, e lo
 %memorizza in una matrice

 [STATIC_AVERAGE] = motion_trail(M0,1);

 %le 3 righe seguenti, vettorizzano la matrice
 %creata dal motion_trail e memorizzano il
 %risultato in Xtest

 B=STATIC_AVERAGE';
 Xtest=B(:);
```

```
 Xtest=Xtest';

 %creo il modello Knn, utilizzando
 %la funzione knnFit, e passando
 %come parametri il training_set
 %e le classi
 model = knnFit(Xtrain, ytrain, 1,max(ytrain));

 %tramite la funzione knnPredict
 %classifico il video sulla base
 %del modello appreso
 ypred = knnPredict(model, Xtest);
end
```

Le funzioni "KnnFit" e "KnnPredict" sono state estratte direttamente dal PMTK3 (file KnnFit.m e KnnPredict.m) e lincate al progetto.

REFERENCES

[1] M. Dunham and K. Murphy, "Pmtk3: Probabilistic modeling toolkit for matlab/octave, version 3," 2012.

[2] K. P. Murphy, *Machine learning: a probabilistic perspective*. The MIT Press, 2012.

[3] T. M. Mitchell, "Naïve bayes and logistic regression," *Machine learning*, vol. 10, p. 701, 2005.