

# Accelerometer Based Gesture Recognition using HMMs

Andrea Tarocchi, Marco Magnatti

8 gennaio 2009



# Outline

- 1 Introduzione
- 2 Quantizzazione dei dati
- 3 HMM
  - Definizione
  - Prediction
  - Learning
  - Decoding
- 4 Decisore
- 5 Test e risultati



## Descrizione del problema

Da un device dotato di accelerometri si ricevono i dati relativi alle accelerazioni a cui è sottoposto durante il moto (lungo le tre coordinate spaziali).

Tale device può essere impugnato per eseguire gesture.

Il nostro obiettivo è realizzare un sistema che, dopo adeguato addestramento, sia in grado di riconoscere gesti effettuati tramite tale dispositivo.



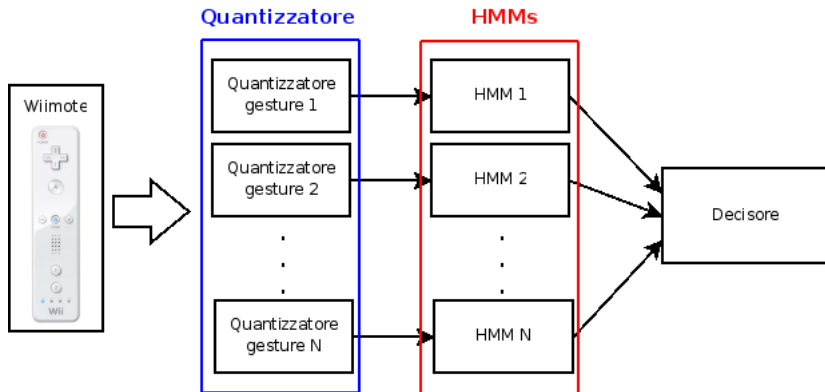
## Strumenti utilizzati



- Device di acquisizione dati: Nintendo Wiimote
- Linguaggio di programmazione utilizzato: C/C++
- wiiuse: libreria C per comunicazione PC - Wiimote
- boost C++: librerie C++ con numerose strutture dati
- HMM (Hidden Markov Model): per il riconoscimento



# Struttura generale



## Quantizzazione con k-means

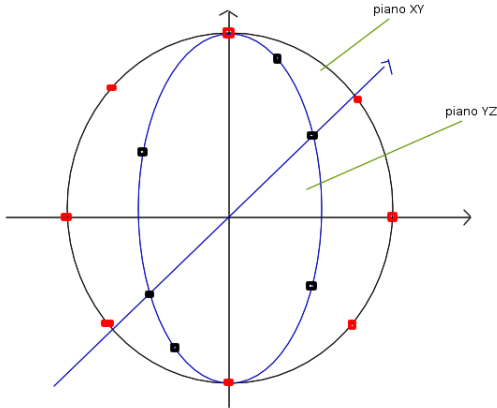
Poiché i dati in arrivo dal Wiimote sono valori in campo reale compresi tra  $-4G$  e  $+4G$ , è necessaria una discretizzazione prima di poterli sottoporre ad un HMM che, per scelta implementativa, lavora in campo discreto.

La discretizzazione viene fatta tramite l'algoritmo k-means.

Per via sperimentale, si è stabilito la posizione iniziale e il numero dei centroidi da utilizzare (e di conseguenza il numero di differenti simboli che l'HMM può emettere).



# Posizione dei centroidi



Possibili approcci alternativi:

- Inizializzazione random
- Inizializzazione lungo la direzione del vettore medio



# HMM

Con HMM è possibile affrontare problemi del tipo “modellare la probabilità  $P(X)$  di una sequenza di osservazioni  $X$ ”.

Nel nostro caso, le sequenze osservate sono costituite dai vettori di accelerazioni quantizzati in un numero finito di simboli.

Si assume che la sequenza osservata sia generata da una sequenza  $S$  di variabili nascoste, chiamate stati.

Altra assunzione fondamentale è che “il futuro è indipendente dal passato, dato il presente”, ovvero vale:

$$S[t+1] \perp S[1], S[2], \dots, S[t-1], X[1], X[2], \dots, X[t] | S[t]$$

$$X[t+1] \perp S[1], S[2], \dots, S[t], X[1], X[2], \dots, X[t] | S[t+1]$$





# Elementi di un HMM

Un HMM  $\lambda = (\pi, A, B)$  è caratterizzato da:

- $N$ , il numero di stati (nascosti) nel modello
- $M$ , il numero di distinti simboli osservabili (alfabeto)
- $\pi = \{\pi_i\}$ , con  $\pi_i = P[q_1 = S_i]$ , il vettore delle probabilità iniziali degli stati
- $A = \{a_{ij}\}$ , con  $a_{ij} = P[q_{t+1} = S_j | q_t = S_i]$ , la matrice di transizione tra gli stati
- $B = \{b_{jk}\}$ , con  $b_{jk} = P[\text{osservo il simbolo } v_k \text{ al tempo } t | q_t = S_j]$ , la matrice di emissione (essendo il modello stazionario,  $B$  è indipendente dal tempo)



## I 3 problemi di base

Ci sono 3 problemi la cui risoluzione è indispensabile affinché il modello risulti utilizzabile in applicazioni reali:

- 1 **Prediction:** data una sequenza di simboli  $V = \{v_1, \dots, v_T\}$  e un modello  $\lambda$ , calcolare la *likelihood*  $P(V|\lambda)$
- 2 **Learning:** date una o più sequenze di training, calcolare i parametri del modello  $\lambda$  che meglio spiegano i dati
- 3 **Decoding:** data una sequenza di simboli  $V = \{v_1, \dots, v_T\}$  e un modello  $\lambda$ , determinare la più probabile sequenza di stati che possa aver generato i simboli dati



# Soluzione al problema 1

Vogliamo calcolare la probabilità della sequenza di osservazioni  $O = (O_1, O_2, \dots, O_T)$  dato il modello  $\lambda$ , ossia  $P(O|\lambda)$ .

Il modo più immediato di farlo comporterebbe enumerare tutte le possibili sequenze di stati  $Q_k$ , sommando poi le probabilità condizionate  $P(O|Q_k, \lambda) \cdot P(Q_k|\lambda)$ , ma arriverebbe ad avere una complessità intrattabile (dell'ordine di  $2T \cdot N^T$ ).

Pertanto, si ricorre alla *Forward-Backward Procedure*, che vedremo avere una complessità dell'ordine di  $N^2 \cdot T$ .



# The Forward-Backward Procedure

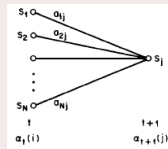
Definiamo la variabile forward  $\alpha_t(i)$  come:

$$\alpha_t(i) = P(O_1, O_2, \dots, O_t, q_t = S_i | \lambda)$$

ossia la probabilità di osservare  $O_1, \dots, O_t$  (sequenza parziale) ed essere nello stato  $S_i$  al tempo  $t$ , dato il modello  $\lambda$ .

Procediamo per induzione su  $\alpha_t(i)$ :

- 1 Inizializzazione:  $\alpha_1(i) = \pi_i b_i(O_1)$
- 2 Induzione:  $\alpha_{t+1}(j) = (\sum_{i=1}^N \alpha_t(i) a_{ij}) \cdot b_j(O_{t+1})$
- 3 Terminazione:  $P(O|\lambda) = \sum_{i=1}^N \alpha_T(i)$



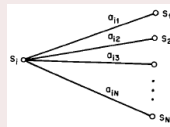
Definiamo la variabile backward  $\beta_t(i)$  come segue:

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \dots, O_T | q_t = S_i, \lambda)$$

ossia, la probabilità di osservare la parte finale della sequenza, da  $t + 1$  fino alla fine, dato lo stato  $S_i$  al tempo  $t$  e il modello  $\lambda$ .

Procediamo per induzione su  $\beta_t(i)$ :

- 1 Inizializzazione:  $\beta_T(i) = 1$
- 2 Induzione:  $\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$



## Soluzione al problema 2

Non si conosce un metodo analitico per determinare il modello che massimizza la probabilità della sequenza data. Si può usare l'algoritmo di Baum-Welch per massimizzare localmente  $P(O|\lambda)$ .

Definiamo la variabile  $\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda)$  ovvero la probabilità di trovarsi nello stato  $S_i$  al tempo  $t$  e nello stato  $S_j$  al tempo  $t + 1$ , dati  $O$  e  $\lambda$ . Possiamo esprimerla in funzione delle variabili forward e backward:

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)}$$



Definiamo la variabile  $\gamma_t(i) = P(q_t = S_i | O, \lambda)$ , ossia la probabilità di trovarsi nello stato  $S_i$  al tempo  $t$  dati  $O$  e  $\lambda$ .

Questa può essere ricavata dalle variabili forward e backward come segue:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} \quad (1)$$

e in funzione della variabile  $\xi_t(i, j)$ :

$$\gamma_t(i) = \sum_{j=1}^N \xi_t(i, j) \quad (2)$$



Sommando  $\gamma_t(i)$  rispetto a  $t$ , si ottiene una quantità che indica il numero atteso di volte che passeremo per lo stato  $S_i$ . Analogamente, la somma di  $\xi_t(i, j)$  su  $t$  può essere interpretata come il numero (atteso) di transizioni tra  $S_i$  e  $S_j$ . Pertanto:

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{numero atteso di transizioni da } S_i$$

$$\sum_{t=1}^{T-1} \xi_t(i, j) = \text{numero atteso di transizioni da } S_i \text{ a } S_j$$

Usando le formule appena introdotte e il concetto di conteggiare le occorrenze degli eventi ai fini di stimare le probabilità, è possibile fornire un metodo per la ristima dei parametri dell'HMM.





## Formule di ristima dei parametri

$$\overline{\pi_i} = \text{numero atteso di volte in stato } S_i \text{ al tempo } 1 = \gamma_1(i)$$

$$\overline{a_{ij}} = \frac{\text{numero atteso di transizioni da } S_i \text{ a } S_j}{\text{numero atteso di transizioni da } S_i} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\overline{b_j(k)} = \frac{\text{numero atteso di volte in } S_j \text{ osservando il simbolo } v_k}{\text{numero atteso di volte nello stato } S_j} = \frac{\sum_{t=1, s.t. O_t=v_k}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)}$$



## Soluzione al problema 3

Ci sono differenti modi per trovare la sequenza di stati “*ottimale*” data la sequenza di osservazioni.

Un possibile criterio è di scegliere gli stati  $q_t$  che sono individualmente più probabili

Utilizzando le variabili  $\gamma_t(i)$  prima definite si può trovare lo stato  $q_t$  più probabile al tempo  $t$  come:

$$q_t = \arg \max_{1 \leq i \leq N} [\gamma_t(i)], \quad 1 \leq t \leq T$$



Un criterio largamente usato è quello di trovare la singola miglior sequenza ossia massimizzare  $P(Q|O, \lambda)$  che è equivalente a massimizzare  $P(Q, O|\lambda)$ .

Per fare questo si utilizza l'algoritmo di Viterbi.  
E' utile introdurre la quantità:

$$\delta_t(i) = \max_{q_1, q_2, \dots, q_{t-1}} P(q_1, q_2, \dots, q_t = i, O_1 O_2 \dots O_t | \lambda)$$

Ossia è la più alta probabilità lungo un singolo cammino al tempo  $t$  che tiene conto delle prime  $t$  osservazioni e finisce nello stato  $S_i$



## Algoritmo di Viterbi

### 1 Inizializzazione:

- $\delta_1(i) = \pi_i b_i(O_1), \quad \psi_1(i) = 0, \quad 1 \leq i \leq N$

### 2 Ricorsione:

- $\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(O_t), \quad 1 \leq j \leq N$

- $\psi_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}], \quad 2 \leq t \leq T$

### 3 Terminazione

- $P^* = \max_{1 \leq i \leq N} [\delta_T(i)]$

- $q_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)]$

### 4 Sequenza degli stati:

- $q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \dots, 1$



# Riconoscimento della gesture

- Per ciascuna classe di gesture che si intende riconoscere, si addestra un distinto HMM.
- Data una gesture  $O$  da riconoscere, questa viene sottoposta a ciascun modello  $\lambda_i$ , e tramite la procedura Forward si calcolano le probabilità  $P(O|\lambda_i)$ .
- La gesture viene attribuita alla classe relativa all'HMM che produce la probabilità maggiore (se almeno una di queste supera una soglia minima, altrimenti la gesture viene classificata come “sconosciuta”).



Quello che vogliamo sapere è la probabilità che una data sequenza  $O$  appartenga alla classe  $C_i$ , ovvero  $P(C_i|O)$ .

La procedura Forward ci permette invece di determinare  $P(O|C_i)$ , ma vale

$$P(C_i|O) = \frac{P(O|C_i)P(C_i)}{P(O)}$$

Pertanto, per prendere la decisione, dovremmo scegliere la  $P(O|C_i)P(C_i)$  massima. Possiamo però limitarci a decidere sulla base della sola  $P(O|C_i)$  in quanto si assume che ciascuna classe di gesture sia equiprobabile.


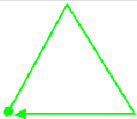
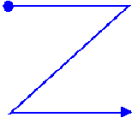


## Test eseguiti

- Si è messo alla prova il software su 3 classi di gesture: triangolo, quadrato e zeta.
- La topologia utilizzata è quella left-to-right, con span 2 e 5 diversi stati (combinazione di parametri che ha prodotto i risultati migliori).
- Il dataset utilizzato comprende circa 50 gesture per classe, eseguite da 2 persone distinte.
- La valutazione delle prestazioni è stata effettuata tramite 3-fold cross validation.



# Risultati

Classe 1:	Fold 1:	12/17		
	Fold 2:	12/17		
	Fold 3:	12/17		
	Tot:	40/51	78%	
Classe 1:	Fold 1:	16/16		
	Fold 2:	17/17		
	Fold 3:	17/17		
	Tot:	50/50	100%	
Classe 1:	Fold 1:	14/14		
	Fold 2:	14/14		
	Fold 3:	14/14		
	Tot:	42/42	100%	



# Bibliografia



**Lawrence R. Rabiner**

*A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.*



**Xuedong Huang, Alex Acero, Hsiao-Wuen Hon**

*Spoken Language processing. A guide to theory, algorithm and system development.*

Prentice Hall, 2001



**Vesa-Matti Mäntylä**

*Discrete Hidden Markov Models with application to isolated user-dependent hand gesture recognition.*

VTT Publications, 2001

