

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.



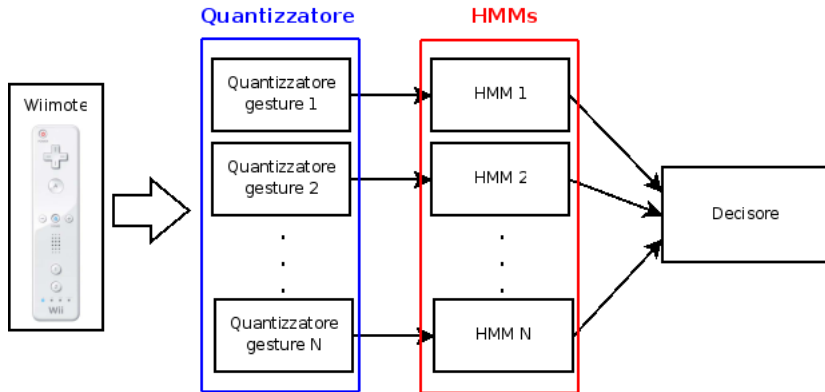
Strumenti utilizzati



- 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

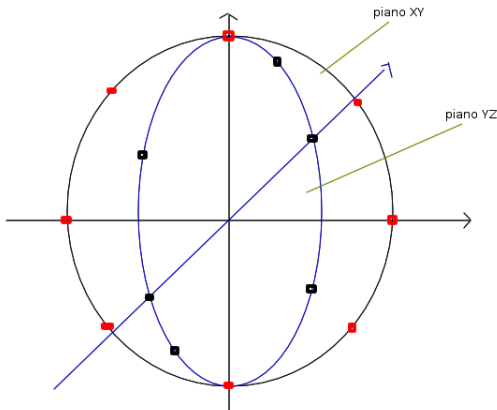
Poiché i dati in arrivo dal Wiimote sono valori in campo reale compresi tra $-4G$ e $+4G$, è necessaria una discretizzazione prima di poterli sottoporre ad un HMM che, per scelta implementativa, lavora in campo discreto.

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 $\pi_i = P[q_1 = S_i]$, è il vettore delle probabilità iniziali degli stati
- $A = \{a_{ij}\}$, con $a_{ij} = P[q_{t+1} = S_j | q_t = S_i]$, è la matrice di transizione tra gli stati
- $B = \{b_{jk}\}$, con $b_{jk} = P[\text{osservo il simbolo } v_k \text{ al tempo } t | q_t = S_j]$, è 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 λ , calcolare la *likelihood* $P(V|\lambda)$
- 2 **Learning:** date una o più sequenze di training, calcolare i parametri del modello λ che meglio spiegano i dati
- 3 **Decoding:** data una sequenza di simboli $V = \{v_1, \dots, v_T\}$ e un modello λ , determinare la più probabile sequenza di stati che possa aver generato i simboli dati



Soluzione al problema 1

Vogliamo calcolare la probabilità di una sequenza di osservazioni $O = O_1, O_2, \dots, O_T$ dato il modello λ , 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

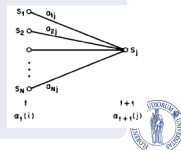
Definiamo la variabile forward $\alpha_t(i)$ come:

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

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

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: $P(O|\lambda) = \sum_{i=1}^N \alpha_T(i)$



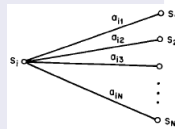
Definiamo la variabile backward $\beta_t(i)$ come segue:

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

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 λ .

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 λ .

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 λ .

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 stima dei parametri

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

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

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



CCCP-2

Algoritmo Iterativo

Si può dimostrare che:

- $J(\Theta^t)$ decresce ad ogni iterazione;
- la procedura converge ad un minimo locale.



CCCP per TSVMs

La Ramp Loss può essere riscritta come differenza tra due Hinge Loss

$$R_s(z) = H_1(z) - H_s(z) \quad (3)$$

È quindi possibile applicare la procedura CCCP al problema di minimizzazione TSVM (3). Il costo $J^s(\Theta)$ può essere decomposto in una parte convessa $J_{\text{vex}}^s(\Theta)$ e in una parte concava $J_{\text{cav}}^s(\Theta)$



CCCP per TSVMs-2

$$\begin{aligned}
 J^s(\Theta) &= \frac{1}{2} \|w\|^2 + C \sum_{i=1}^L H_1(yif_{\Theta}(\mathbf{x}_i)) + C^* \sum_{i=L+1}^{L+2U} R_s(yif_{\Theta}(\mathbf{x}_i)) = \\
 &= \underbrace{\frac{1}{2} \|w\|^2 + C \sum_{i=1}^L H_1(yif_{\Theta}(\mathbf{x}_i)) + C^* \sum_{i=L+1}^{L+2U} H_1(yif_{\Theta}(\mathbf{x}_i))}_{J_{vex}^s(\Theta)} - \\
 &\quad \underbrace{C^* \sum_{i=L+1}^{L+2U} H_s(yif_{\Theta}(\mathbf{x}_i))}_{J_{cav}^s(\Theta)}.
 \end{aligned}$$

(4)



CCCP per TSVMs-3

La decomposizione precedente permette di applicare la procedura CCCP. Il problema convesso alla base della procedura viene espresso utilizzando le variabili duali α . Dobbiamo anche implementare il vincolo di bilanciamento (4). Per semplificare l'approssimazione del primo ordine della parte concava, denotiamo:

$$\beta_i = \frac{\partial J_{cav}^s(\Theta)}{\partial f_{\Theta}(x_i)} = \begin{cases} C^* & \text{if } y_i f_{\Theta}(\mathbf{x}_i) \\ 0 & \text{altrimenti} \end{cases}, \quad (5)$$

per gli esempi non etichettati ($\beta_i = 0$ per $i \leq L$).



CCCP per TSVMs-4



Complessità

Addestrare con la procedura CCCP-TSVM richiede la risoluzione di una serie di problemi di ottimizzazione SVM con $(L + 2U)$ variabili. Sebbene SVM abbia complessità $\mathcal{O}(L + 2U)^2$ nel caso peggiore, nel caso tipico scala quadraticamente e lo stesso vale per i sottoproblemi di TSVMs. Assumendo un numero costante di iterazioni TSVMs con CCCP dovrebbero scalare quadraticamente nella maggior parte dei casi.



Esperimenti eseguiti

Si sono condotti esperimenti sul database MNIST trattato come un problema a 10-classi. Il dataset ha 42k esempi di training, 18k di validation e 10k di test. Il training set \tilde{A} è stato campionato per ottenere i punti etichettati ed $\tilde{A}_{\frac{1}{2}}$ è stato usato il training set per gli esempi non etichettati. Abbiamo replicato il set-up del paper ovvero:

- $C = 10$ e $\gamma = 0.0128$ con kernel RBF;
- C^* variabile con il numero di esempi non etichettati e $s = 0.01$.

L'errore $\tilde{A}_{\frac{1}{2}}$ è stato valutato sul validation test.







Commento dei risultati

Dal grafico dell'errore si nota che l'**aggiunta di dati non etichettati porta ad un significativo miglioramento delle prestazioni**. A meno di uno scarto medio dell' 1,71% l'andamento dell'errore degli esperimenti effettuati è $\frac{1}{2}$ lo stesso di quelli riportati nel paper. Dal grafico dei tempi di vede chiaramente che il tempo di ottimizzazione ha un andamento **quadratico** con il numero dei dati non etichettati confermando le ipotesi fatte in precedenza.



SSL Vs Transductive Learning

Se si considera l'apprendimento trasduttivo come una versione di SSL dove i dati non etichettati sono il test set di interesse, $\tilde{\mathcal{Z}}_{\frac{1}{2}}$ possibile comparare le prestazioni dei due approcci. L'esperimento prevede l'utilizzo di dati non etichettati diversi dal test set per l'SSL. Nello specifico abbiamo utilizzato 10000 punti non etichettati, un test set di 10000 punti e un training set di 1000 punti. I risultati ottenuti:

- Testing error SSL: 6,944%;
- Testing error Transductive: 6,350%.



Considerazioni

- Si è verificato che l'utilizzo di dati non etichettati migliora le prestazioni di un classificatore induttivo.
- Comunque il trade-off tra costi computazionali e benefici non è soddisfacente.
- Dal momento che l'algoritmo ottimizza problemi SVMs standard, ogni miglioramento delle tecniche di risoluzione di questi si riflette immediatamente sulle TSVMs.
- Interessante l'approccio trasduttivo in particolari situazioni.



Bibliografia



Collobert R. and Sinz F. and et al.
Large Scale Transductive SVMs.



Schlkopf, Bernhard and Smola, Alexander J.
Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning).
The MIT Press, 2001.



Chapelle, O. and Schölkopf, B. and Zien, A.
Semi-Supervised Learning (Adaptive Computation and Machine Learning).
The MIT Press, 2006.



Bibliografia



Yuille A. L. and Rangarajan A.
The concave-convex procedure (CCCP).
The MIT Press, 2002.



Vapnik, Vladimir N.
The Nature of Statistical Learning Theory.
Springer, 1999.



N. Cristianini and J. Shawe-Taylor.
Kernel Methods for Pattern Analysis.
Cambridge Univ. Press, 2004.

