TECNICHE DI RICONOSCIMENTO E VISUALIZZAZIONE

MANUEL IURISSEVICH

Indice

Elenco delle figure

1. Acquisizione

Le sequenze di esempio sono state registrate con l'accelerometro triassiale di un telefono cellulare GT-S7501, grazie all'applicazione *Accelerometer Monitor*.

Consistono in 16 movimenti di sollevamento di cui i primi 8 sono considerati corretti, gli altri errati. I sollevamenti sono effettuati a gomito vincolato, quindi soltanto con l'avambraccio. Durante i movimenti "giusti" la traiettoria è mantenuta il più possibile dritta, mentre durante quelli "sbagliati" si devia dall'asse verticale.

Il cellulare è tenuto in mano in maniera che lo schermo guardi nella stessa direzione del palmo. Il polso è rigido.

1.1. **Specifiche tecniche.** Forse a questo punto non sono granché interessanti.

1.2. **Il punto di vista biomeccanico.** Fissata che sia la posizione della punta del gomito, l'avambraccio tre gradi di libertà di rotazione. Il polso pu'o ruotare in altri due modi, per un totale di cinque gradi di libertà.

In questa dimostrazione si cerca di sopprimere i movimenti del polso e la rotazione sull'asse dell'avambraccio. Fatto questo, si considera corretta l'esecuzione del movimento quando non c'è variazione dell'angolo longitudinale ma solo di quello latitudinale.[1]

1.3. **Elaborazione preliminare.** I dati grezzi in uscita dal sw di acquisizione devono subire un trattamento prima di passare attraverso qualunque algoritmo di classificazione.
Smoothing. Si fa innanzitutto passare una media mobile per togliere un po' di rumore. Buoni risultati con un filtro lungo 13 campioni. Il filtro è applicato "a mano" con un ciclo for e non con l'apposito comando $signal : filter$ perché contemporaneamente si esegue un sottocampionamento di fattore 3. Si potrebbe ovviamente fare prima il filtraggio con $filter$ e comunque sottocampionare dopo e credo anzi sia consigliato.
Suddivisione. La seconda fase consiste nella divisione della registrazione (che contiene tutti e 16 gli esempi) in esempi separati. Questo è fatto sia utilizzando tecniche software che cercando le divisioni a mano. Si fa in modo che tutte le sequenze abbiano una lunghezza di 64 campioni ma non si bada al fatto che inizino tutte allo stesso istante. Si calcolano quindi alcune caratteristiche statistiche dei dati acquisiti. Basandoci su *un lavoro precedente*, riteniamo che valga la pena concentrarsi su media e varianza.
Visualizzazione. Siamo passati quindi dal rappresentare i dati in uno spazio a $64 \times 3 = 192$ dimesioni ad uno di $2 * \times 3 = 6$ dimensioni. Per la visualizzazione dei dati in questa fase si pu'o ricorrere alle proiezioni ortogonali rappresentando due coordinate per volta (per un totale di $\sum_{n=1}^{6-1} n = 15$ grafici in 2D), oppure ricorrere a tecniche di scalaggio multidimensionale con il comando $statistics : mdscsale$ al quale diamo in pasto le distanze (euclidee) trai punti in 6D. Abbiamo seguito quest'ultima via.

2. Tecniche classiche

K-mean clustering, apprendimento competitivo, neurone.

2.1. **K-mean clustering.** Tecnica di classificazione non supervisionata (non bisogna etichettare i dati prima di far partire l'algoritmo) che consiste in

(1) gettare più o meno casualmente K *centroidi* nello spazio
(2) etichettare ogni punto con l'ID del centroide più vicino
(3) spostare ogni centroide sulla media dei punti a esso assegnati
(4) ripetere dal punto 2 finché i centroidi non sono quasi fermi

Con Matlab. Il toolbox *statistics* mette a disposizione il comando *kmeans* che, fornitigli i punti e il numero di gruppi desiderati, restituisce le etichette per ogni punto e le coordinate dei centroidi. Altro scalaggio per visualizzare i dati in 2D.
Crossvalidazione. Il numero di centri K è fissato a priori ma non sempre si hanno buoni motivi per scegliere un valore pittosto che un altro. Per trovare il K migliore si fa di solito correre l'algoritmo più volte, ogni volta cambiando il K e valutando le prestazioni. L'insieme dei dati dev'essere preventivamente diviso in un gruppo di addestramento e un gruppo di test, o validazione.

[1] Terminologia!!

2.2. **Apprendimento competitivo.** Secondo me concettualmente simile al K-means, è l'apprendimento competitivo. (*Attenzione!* Non sono la stessa cosa e non portano agli stessi risultati)

Consiste nello strutturare una rete di K neuroni ciascuno con il suo vettore dei pesi inizialmente casuale ma sempre di modulo 1. In questo modo a ogni neurone è assegnato un versore in un qualche spazio. I neuroni sono tra loro collegati tramite linee di segnali inibitori. Quando le coordinate di un punto di addestramento sono date in pasto alla rete, ci sarà un neurone che indica in quella direzione, o per lo meno un neurone sarà più vicino degli altri: questo è l'unico neurone autorizzato ad apprendere e modificherà i suoi pesi in maniera da avvicinarsi un po' alla direzione del punto. L'apprendimento degli altri neuroni è inibito.

Questa tecnica richiede ulteriori pretrattamenti dei dati in ingresso in maniera che siano distribuiti (meglio possibile) su una sfera centrata nell'origine.

2.3. **Il percettrone classico.** Il percettrone cerca sempre di dividere l'universo mettendoci in mezzo un piano. Tale piano è individuato dai pesi del neurone e da una soglia, che in pratica è un ulteriore sinapsi con l'ingresso sempre a 1 (o -1, se si preferisce). La funzione di attivazione pu'o essere tranquillamente la funzione segno, senza scomodare esponenziali o tangenti iperboliche. Tanto c'è un solo neurone e non dobbiamo fare backpropagation.

Fatto sta che per ogni vettore d'ingresso il percettrone calcola

$$y = sgn\,(x \cdot w - \theta)\,; \tag{1}$$

in fase di addestramento, si confronta l'uscita calcolata con quella desiderata: $e = d - y$ e si correggono i pesi di un fattore $\Delta w = a\,e\,x$, dove $a \in [0, 1]$ è il fattore di apprendimento. Analogo per la soglia.

Ad apprendimento concluso i pesi non variano più ed il piano di separazione è finalmente individuato dall'equazione 1, posta uguale a 0.

2.4. **La selezione delle caratteristiche.** Nonostante l'applicazione richieda di classificare, uno degli aspetti più interessanti di tutto il discorso è valutare le caratteristiche importanti che permettano di classificare le sequenze con la massima efficacia. Ridurre insomma al minimo le variabili e concentrarsi su quelle che varamente portano informazione.

- quanto è utile questa caratteristica a farmi classificare i dati?
- quanto sono mescolati i dati lungo questa caratteristica (vista come coordinata)
- che è il contrario di "quanto sono ordinati?"
- quanto mi è utile sapere che (su questa caratteristica) un dato viene prima di un altro?

In una prima fase. si ordinano i valori secondo la caratteristica e moltiplica le etichette ordinate per le etichette non riordinate. La somma risultante (prod.scalare) dà l'importanza della caratteristica.
Il neuroncino. Ha un solo peso $w = 1$ e deve soltanto variare la soglia.
A questo punto. Si possono dedicare attenzioni soltanto alle caratteristiche più importanti, oppure "scartarle" e concentrarsi su relazioni più sottili. Bisogna fate poi il discorso delle caratteristiche dipendenti: se due caratteristiche sono dipendenti, non occorre considerarle entrambe.

3. Tecniche moderne

Dopo esserci concentrati sulle tecniche vintage, consideriamo quelle in voga negli ultimi vent'anni. Si attribuisce a un certo Vapnik l'applicazione fruttuosa del kernel all'apprendimento automatico (1992).

3.1. **Il Kernel.** Prende il nome da qualche teoria matematica avanzata. Per capirsi

- il prodotto scalare è un kernel e
- il kernel è un prodotto scalare.

Dalla prima affermazione si pu'o intuire che, presi due vettori, due elementi di uno spazio in cui sia definito un prodotto scalare. il kernel sarà un'estensione del prodotto scalare consueto.

La seconda introduce l'idea di portarsi in un comodo spazio dotato di prodotto scalare per poi applicare il kernel.

Formalmente:

$$k = \langle \Phi(x_1), \Phi(x_2) \rangle \tag{2}$$

dove $\Phi(x)$ è una mappa -se si vuole- non lineare che porta i generici elementi x in uno spazio dotato di prodotto scalare.

NB_1: Gli elementi originali possono quindi non essere neanche vettori.

NB_2: In generale è più semplice scrivere la funzione kernel piuttosto che la mappa.

Oss: Praticamente si stanno estraendo le caratteristiche.

3.2. **Il kernel e l'apprendimento automatico.** Rimanendo nell'ambito della classificazione binaria la funzione discriminante è

$$y_n = sgn\left(\langle w, \Phi(x_n) \rangle + b\right) \tag{3}$$

ma si definiscono i pesi come

$$w = \sum_i y_i \alpha_i \Phi(x_i) \tag{4}$$

con α_i che come si chiama? Sostituendo la 4 nella 3 si ottiene

$$y_n = sgn\left(\sum_i y_i \alpha_i \langle \phi_i, \phi_n \rangle + b\right) \tag{5}$$

$$= sgn\left(\sum_i y_i \alpha_i k(x_i, x_n) + b\right) \tag{6}$$

3.3. **Il kernel perceptron.** (Tratto da un articolo dell'Università di Trento). L'algoritmo di apprendimento è molto simile a quello del percettrone classico. La differenza sta nel calcolo della funzione (vedi eq. precedente) e nell'aggiornamento dei pesi: gli α_i sono incrementati ogni volta che il neurone sbaglia la previsione su un x_i.

3.4. **Simple Pattern Recognition Algorithm (SPRA).** Schoelkopf e Smola propongono un semplice algoritmo di classificazione introduttivo alle Macchine a Vettori di Supporto (SVM).

Non si fa altro che prendere i punti medi delle due classi (c_1 e c_2), prendere il vettore differenza trai due $w = c_1 - c_2$ il loro punto medio ($c = (c1 + c2)/2$).

Ora ogni altro punto di classe ignota può essere etichettato con $y = sgn\left(\langle \boldsymbol{x} - \boldsymbol{c}, \boldsymbol{w} \rangle\right)$.
Osservazioni.

3.5. **Macchine a vettori di supporto (SVM).** Sono algoritmi che usano l'espediente del kernel per mappare le sequenze in degli spazi comodi (tipicamente con molte dimensioni più) dove trovare il piano di separazione che massimizza il margine. Nello spazio di partenza la separazione può assumere le forme più diverse (dipende dalla mappa non lineare).

Il problema è posto nei termini di minimizzazione della lagrangiana

$$L_P = lagrangiana \tag{7}$$

o massimizzazione della sua duale

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \, \alpha_j \, y_i \, y_j \, k(\boldsymbol{x_i}, \boldsymbol{x_j}) \tag{8}$$

$$\langle \boldsymbol{\alpha}, \boldsymbol{y} \rangle = 0 \tag{9}$$

$$\alpha_i \geq 0 \tag{10}$$

3.6. **Prova di ottimizzazione.** Il progetto *ottimizzazione* è una dimostrazione della funzione $fimncon$ di ottimizzazione vincolata.

Questa strumento prende in ingresso la funzione obiettivo e la funzione supporto, vale a dire la funzione da minimizzare e i vincoli non lineari da rispettare; il punto di partenza; gli eventuali vincoli lineari di uguaglianza e disuguaglianza.

3.7. **I consigli di Chang e Lin.** I creatori della libreria *libsvm* danno alcuni suggerimenti per iniziare presto ad avere risultati soddifacienti con le SVM.

- trasformare i dati nel formato di un pacchetto svm
- eseguire uno scalaggio semplice dei dati
- considerare i kernel a base radiale (gaussiani)
- crossvalidare per trovare i C e γ migliori
- usare questi C e γ per addestrare sull'intero insieme di adestramento
- testare

C e γ. Il parametro C è il costo. Serve a rendere più laschi i vincoli dell'ottimizzazione

$$0 \leq \alpha_i \leq C \tag{11}$$

Necessario quando le due classi **non** sono separabili, è utile quando le classi sono separabili in maniera "forzata" cioè l'insieme di addestramento le mostra separabili ma dagli insiemi di validazione o test appare che non lo sono. Il parametro γ fa riferimento alla formula del kernel a base radiale (RBFK)

$$k(\boldsymbol{x_1}, \boldsymbol{x_1}) = \gamma \, ||\boldsymbol{x_1} - \boldsymbol{x_2}||^2 \tag{12}$$

ma non è utilizzato nell'esempio perché invece del kernel a base radiale se ne usa uno polinomiale di grado 2.

3.8. **Esempio di SVM.** Nell'esempio *svmxor* [2] si fanno quindi due passate con una griglia di 5 elementi, prima con $C \in [10^{-5}, \ldots, 10^5]$ poi intorno al C che si è mostrato migliore.

Si chiama quindi la funzione di addestramento $my\_svmtrain$ che per ogni C chiama a sua volta $fmincon$. Terminato l'addestrameto si calcola sull'insieme di validazione la percentuale di previsioni esatte, indicativa della bontà del parametro C usato in quel ciclo.

Si fa girare un'altra volta l'apprendimento con il C migliore trovato e si collauda il tutto sull'insieme di test.

Il resto del documento è dedicato alla visualizzazione dei dati e del processo di apprendimento.

Si è usato l'insieme di dati "Iris".

[2] Il nome deriva dal fatto che il primo problema posto alla macchina era uno XOR

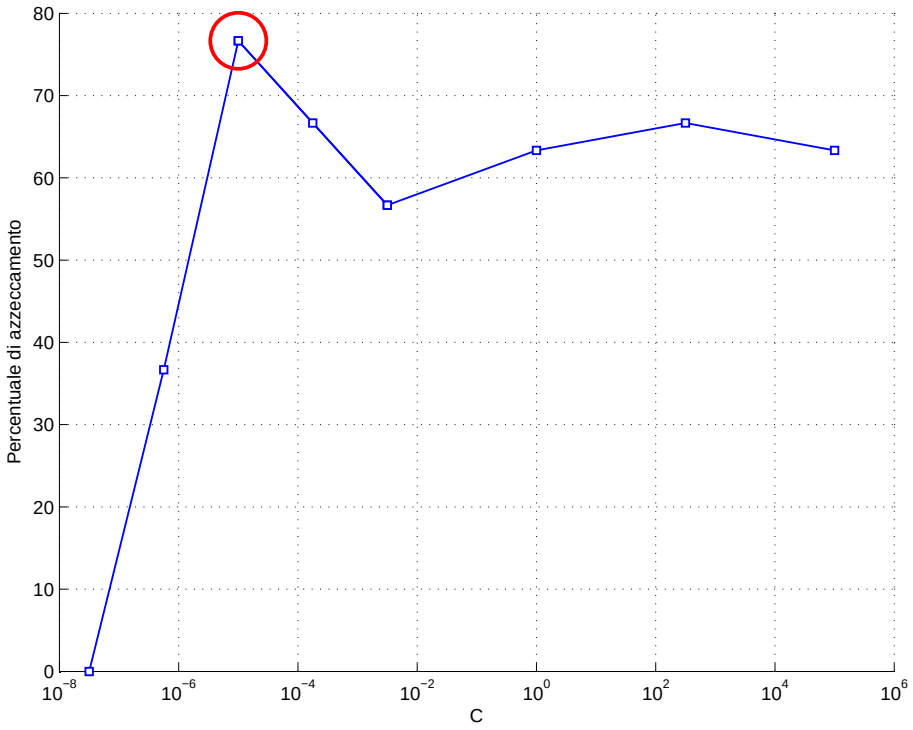

FIGURA 1. Validazione incrociata per la ricerca del miglior parametro C

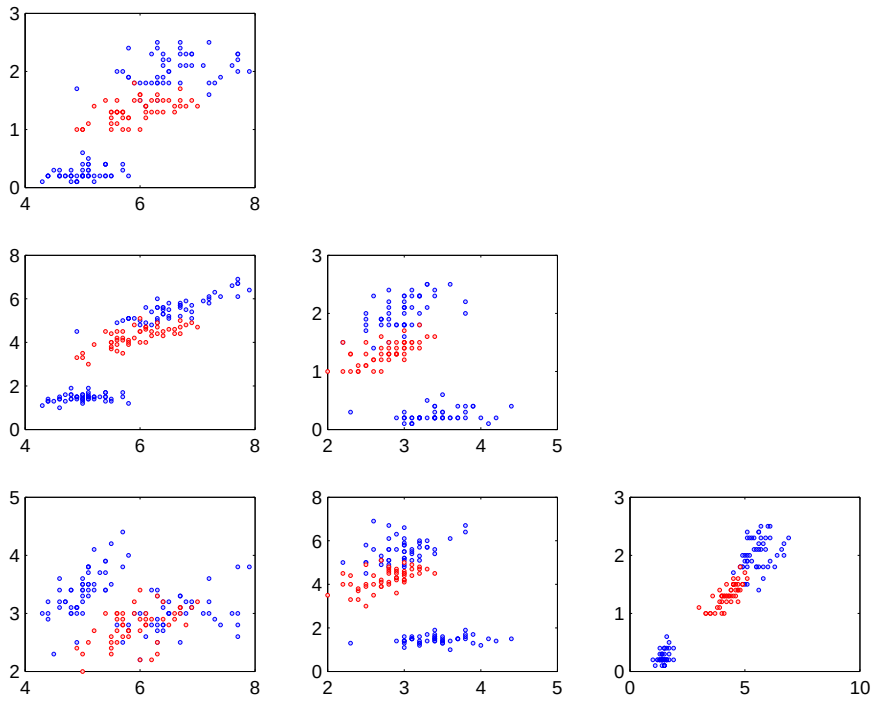

FIGURA 2. Proiezioni ortogonali dei dati di Iris: in rosso le Versicolor (?). Quattro falsi positivi

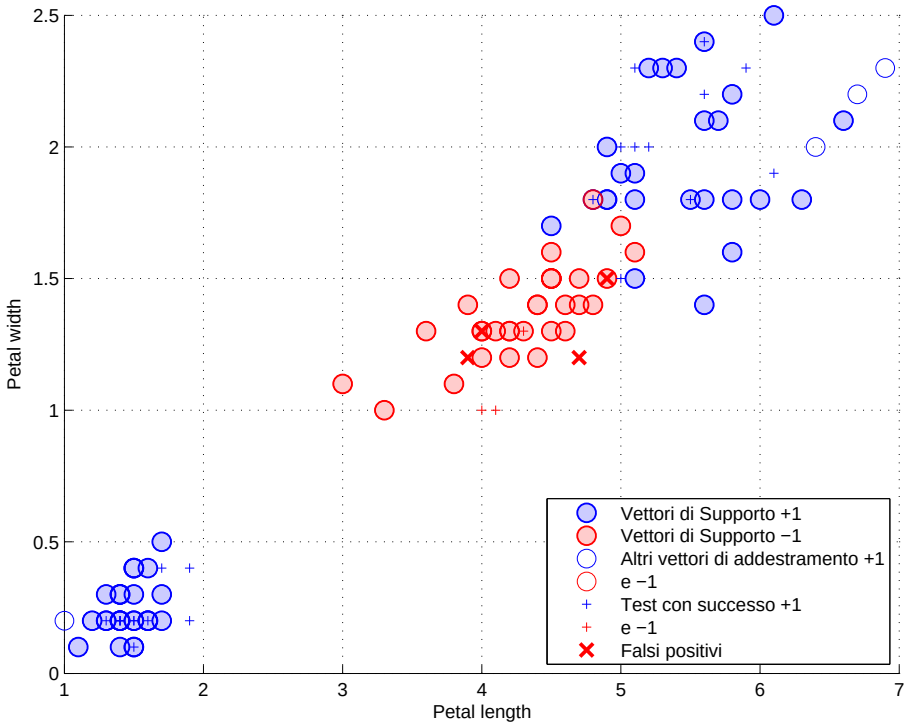

FIGURA 3. Risultati della SVM: si vogliono riconoscere le Versicolor (?)