Università degli studi di
Milano-Bicocca

F1801Q145

Modelli probabilistici per le decisioni

HAR Bayesian Network

Studenti:	*Matricole:*
Artifoni Mattia	807466
Brena Luca	808216
Bottoni Federico	806944

Giugno 2019

Indice

1 Introduzione

Il progetto ha l'obiettivo di creare un modello di Rete Bayesiana capace di predire il tipo di azione che sta effettuando un ipotetico individuo che indossa il "HAR wearable devices setup", una particolare sistema indossabile composto da 4 accelerometri che permette di analizzare i vettori accelerazione dei sensori in questione. Viene fornito dal progetto di riferimento[1] un dataset contenente dati sufficienti per effettuare training e testing del modello

1.1 Dominio di riferimento

La semantica dei dati utilizzati è definita nel paper[5] del progetto di provenienza. La singola entry del dataset rappresenta uno snapshot acquisito dai sensori e consiste in:

- user: username dell'individuo in oggetto (string).
- gender: genere del soggetto (string).
- age: età dell'individuo (int).
- how_tall_in_meters: altezza del soggetto espressa in metri (int).
- weight: peso espresso in kilogrammi (int).
- body_mass_index: indice di massa corporea. Si ottiene dividendo il peso per il quadrato dell'altezza (float).
- xi: intero che esprime la componente x del vettore accelerazione nel sensore i-esimo (int).
- yi: intero che esprime la componente y del vettore accelerazione nel sensore i-esimo (int).
- zi: intero che esprime la componente z del vettore accelerazione nel sensore i-esimo (int).
- class: è la variabile target della previsione e indica l'azione eseguita dal soggetto al momento della rilevazione dei dati. Può assumere il valore di "walking", "standing", "standingup", "sitting" e "sittingdown" (string).

1.2 Ipotesi e assunzioni

Durante lo studio del caso sono state discrimate le features utili al training della rete (i vettori dei sensori) da quelle assunte come superflue (user, gender, age, weight, body_mass_index) le quali potrebbero essere utilizzate per specializzarla ulteriormente.

La scelta riguardo all'attributo *how_tall_in_meters* non è stata particolarmente immediata dato che il training set è definito su un range di 13cm (1.58m - 1.71m) che distribuito in un corpo umano non fornisce l'informazione necessaria per poter affermare che tutti i sensori si trovano 13cm più o meno vicini al terreno. La rete dovrebbe essere comunque in grado di predire le azioni di un bambino, il quale ha altezza decisamente inferiore rispetto a quella precedentemente descritta, ciò nonostante assumiamo che l'altezza dell'utente ricada all'interno del range descritto dato che in alcuni test affrontati, la complessità della rete era tale da scatenare MemoryError nella rappresentazione dei dati.

2 Scelte di design

2.1 Analisi statistica e qualitativa

Il dataset ha subito una prima fase di pulizia, in cui sono stati individuati e eliminati alcuni caratteri non necessari tra i campi e timestamp inaspettati tra le entry della tabella, ed una seconda di shuffle, nella quale i record sono stati randomizzati.
E' stata effettuata inoltre una fase di analisi statistico-descrittiva considerando le features che assumono valori in range indefiniti per cercare di individuare qualche distribuzione particolare o comportamento anomalo. Osservando le misurazioni di media e deviazione standard tra le variabili, si è notato che per alcune di esse i valori assunti di massimo e di minimo risultavano particolarmente distanti dal valor medio. Si è deciso pertanto di individuare e contare i record contenenti tali valori. Rappresentando una minoranza non significativa per la successiva fase di modellazione della rete, si è deciso pertanto di rimuoverli.

Dalla tabella 1 si può notare che i range di variabilità degli attributi non seguono comportamenti particolari, tanto meno le distribuzioni che in alcuni casi sono caratterizzati da deviazione standard particolarmente bassa (come il caso di *x1*) mentre in altri casi molto alta (come *y2*).

2.2 Normalizzazione

Nella fase embrionale della progettazione la normalizzazione era stata ignorata. Infatti la predizione lavorava su features che avevano range piuttosto incosistenti al variare del sensore e della componente del vettore accelerazione considerata. Questo si può notare dalle colonne *Min* e *Max* in tabella 1, che si riferisce ai dati grezzi parzialmente ripuliti. Quindi si è pensato di effettuare una normalizzazione per ogni feature, in modo da compattare i valori tra -1 e 1 andando anche a eliminare gli outliers.

Tabella 1: Analisi descrittiva del dataset

Campo	Min	Max	Media	Moda	DevStd
x1	-306	509	-6.649327127	-1	11.61623803
y1	-271	533	88.29366732	95	23.89582898
z1	-603	411	-93.16461092	-98	39.40942342
x2	-494	473	-87.82750418	-492	169.4351938
y2	-517	295	-52.06504742	-516	205.1597632
z2	-617	122	-175.0552004	-616	192.8166147
x3	-499	507	17.42351464	38	52.63538753
y3	-506	517	104.5171675	108	54.15584251
z3	-613	410	-93.88172647	-102	45.38964613
x4	-702	-13	-167.6414483	-164	38.31134199
y4	-526	86	-92.62517131	-94	19.96861022
z4	-537	-43	88.29366732	-162	13.22102006

Di seguito, nelle figure 1 e 3, sono presentati alcuni grafici che illustrano la distribuzione dei valori di alcune features prima e dopo la normalizzazione. A seguito dell'operazione di normalizzazione il dataset appare come riportato in figura 2.

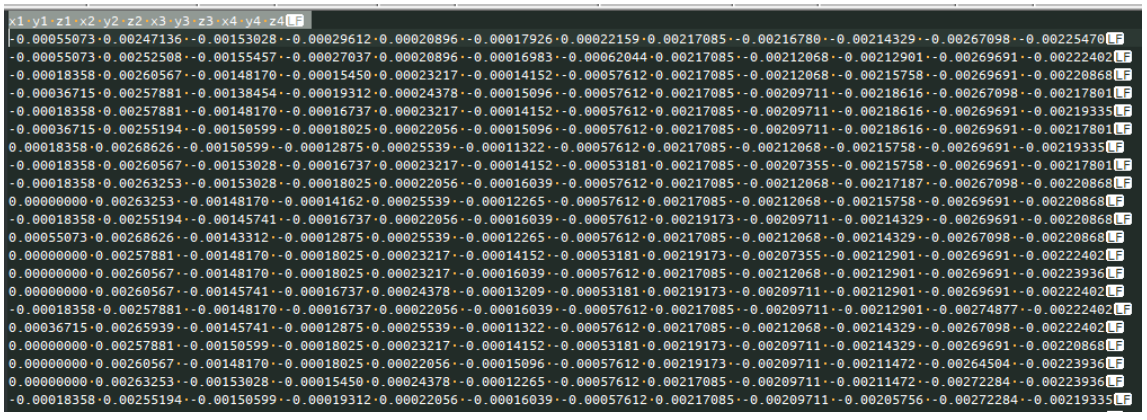
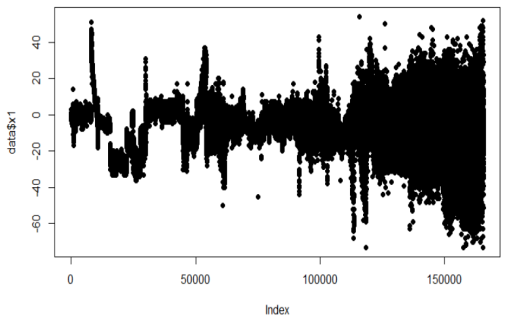


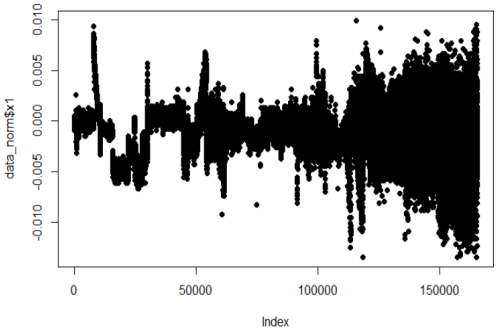
Figura 2: Il dataset normalizzato.

2.3 Discretizzazione

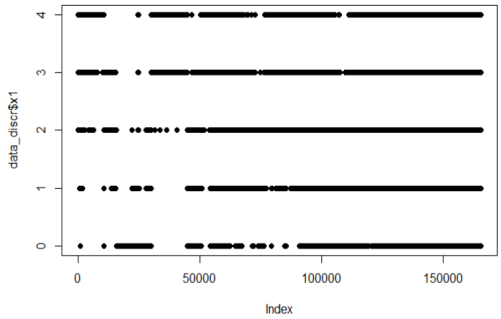
I valori delle features nel dataset originale sono continui. Per lavorare con le reti Bayesiane, fornite dalle librerie [4] e [3] (che non implementano la gestione di variabili continue), è stato necessario ricorrere alla discretizzazione dei dati.



(a) x1 non normalizzata.



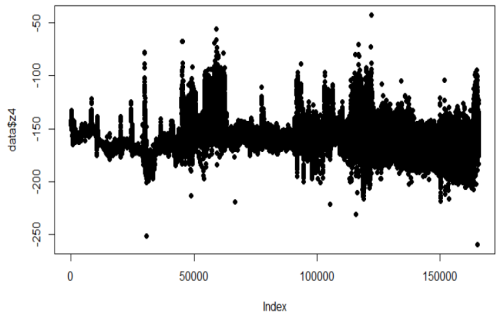
(b) x1 normalizzata.



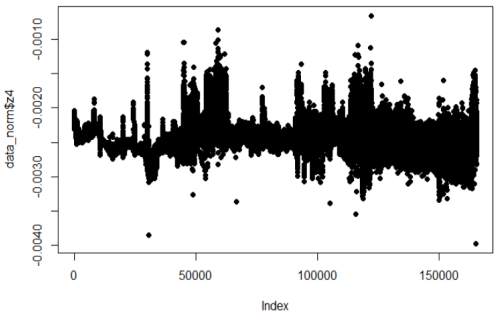
(c) x1 normalizzata e discretizzata.

Figura 1: Normalizzazione e discretizzazione della feature x1.

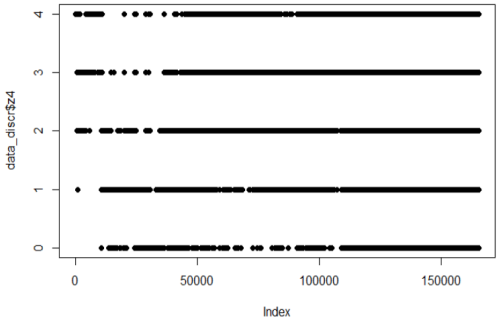
Inizialmente i dati sono stati suddivisi in partizioni a range identici senza tenere conto di come questi fossero distribuiti. Successivamente abbiamo realizzato come questa procedura fosse imprecisa dopo aver osservato il modo in cui le distribuzioni di probabilità si sbilanciavano prevalentemente verso una sola classe della variabile. La discretizzazione definitiva è stata applicata sul dataset normalizzato. Per discretizzare i dati sono stati scelti 5 intervalli o bins. La funzione scelta (KBinsDiscretizer della libreria sklearn [4]) adatta automaticamente il numero dei bins in modo che la distribuzione dei dati in essi sia omogenea. Uno sapshot dei dati discretizzati è raffigurato nell'immagine 4



(a) z4 non normalizzata.



(b) z4 normalizzata.



(c) z4 normalizzata e discretizzata.

Figura 3: Normalizzazione e discretizzazione della feature x1.

3 I modelli di rete

3.1 pgmpy

Il software inizialmente scelto è pgmpy[2] di Python, una libreria che permette di modellare le dipendenze in modo agile e stimare le CPT delle variabili sfruttando dei metodi che accettano il dataset ed infine effettuare inferenze dichiarando la variabile di query e le evidenze.
Utilizzando la libreria ci siamo resi conto di come sia performante utilizzando modelli semplici e correlati da pochi record, tuttavia appena è avvenuta l'esecuzione della stima delle CPT nel primo modello completo ideato abbiamo riscontrato le prime difficoltà.

	A	B	C	D	E	F	G	H	I	J	K	L
1	x1	y1	z1	x2	y2	z2	x3	y3	z3	x4	y4	z4
2	3	2	3	1	2	4	0	2	2	3	1	4
3	3	2	3	1	2	4	0	2	2	4	1	4
4	3	3	3	2	2	4	0	2	2	3	1	4
5	3	3	4	2	2	4	0	2	2	3	1	4
6	3	3	3	2	2	4	0	2	2	3	1	4
7	3	2	3	2	2	4	0	2	2	3	1	4
8	4	3	3	2	2	4	0	2	2	3	1	4
9	3	3	3	2	2	4	0	2	2	3	1	4
10	3	3	3	2	2	4	0	2	2	3	1	4
11	3	3	3	2	2	4	0	2	2	3	1	4
12	3	2	4	2	2	4	0	2	2	3	1	4
13	4	3	4	2	2	4	0	2	2	3	1	4
14	3	3	3	2	2	4	0	2	2	4	1	4
15	3	3	3	2	2	4	0	2	2	4	1	4
16	3	3	4	2	2	4	0	2	2	4	1	4
17	3	3	3	2	2	4	0	2	2	4	1	4
18	4	3	4	2	2	4	0	2	2	3	1	4
19	3	3	3	2	2	4	0	2	2	3	1	4
20	3	3	3	2	2	4	0	2	2	4	1	4
21	3	3	3	2	2	4	0	2	2	4	1	4
22	3	2	3	2	2	4	0	2	2	4	1	4
23	4	3	3	2	2	4	0	2	2	4	1	4
24	3	3	3	2	2	4	0	2	2	3	0	4
25	3	3	3	2	2	4	0	2	2	4	0	4
26	4	3	3	2	2	4	0	2	2	4	0	4
27	3	3	3	2	2	4	0	2	2	3	1	4
28	4	3	3	2	2	4	0	2	2	3	1	4
29	3	2	3	2	2	4	0	2	2	2	1	4
30	4	3	3	2	2	4	0	2	2	1	1	4
31	3	3	3	2	2	4	0	2	2	1	1	4
32	3	3	3	2	2	4	0	1	2	1	1	4
33	3	3	3	2	2	4	0	2	2	0	1	4
34	3	3	3	2	2	4	0	2	2	0	1	4
35	3	3	3	2	2	4	0	2	2	0	1	4
36	3	3	3	2	2	4	0	2	2	0	1	4

Figura 4: Il dataset discretizzato.

3.1.1 Modello correlato

L'idea che sta alla base di questo primo modello consiste nello stimare le dipendenze tramite lo strumento statistico *indice di correlazione di Paerson*, assumendo che se due feautres hanno distribuzioni simili (e quindi alta correlazione diretta o inversa) allora vi è una dipendenza tra le due. Sono state così selezionate le relazioni scartando le simmetrie e stimate le CPT tramite il metodo *MaximumLikelihoodEstimator* della libreria basandosi sui sample passati come parametro.

x1	y1	0.345808064	x2	x1	0.198233156	x3	x1	0.07121	x4	x1	0.099018401
x1	z1	0.030417499	x2	y1	0.043861069	x3	y1	-0.1816	x4	y1	-0.29045831
x1	x2	0.198233156	x2	z1	0.109970588	x3	z1	0.13028	x4	z1	0.50628732
x1	y2	0.202451431	x2	y2	0.977115735	x3	x2	0.00802	x4	x2	0.157007278
x1	z2	0.251138704	x2	z2	0.953083727	x3	y2	0.00636	x4	y2	0.081715165
x1	x3	0.071207709	x2	x3	0.008022688	x3	z2	0.00665	x4	z2	0.287828127
x1	y3	-0.136539934	x2	y3	-0.140455652	x3	y3	0.32843	x4	x3	0.166694382
x1	z3	0.00417011	x2	z3	0.107520027	x3	z3	0.27106	x4	y3	-0.111224183
x1	x4	0.099018401	x2	x4	0.157007278	x3	x4	0.16669	x4	z3	0.035357614
x1	y4	-0.142551934	x2	y4	-0.23983578	x3	y4	0.04272	x4	y4	-0.600982199
x1	z4	-0.025592835	x2	z4	0.164505939	x3	z4	-0.2023	x4	z4	-0.068008246
y1	x1	0.345808064	y2	x1	0.202451431	y3	x1	-0.1365	y4	x1	-0.142551934
y1	z1	-0.5159614	y2	y1	0.138219068	y3	y1	0.19162	y4	y1	0.228997237
y1	x2	0.043861069	y2	z1	0.017749163	y3	z1	-0.119	y4	z1	-0.405502292
y1	y2	0.138219068	y2	x2	0.977115735	y3	x2	-0.1405	y4	x2	-0.23983578
y1	z2	-0.0301789	y2	z2	0.918648041	y3	y2	-0.096	y4	y2	-0.15437806
y1	x3	-0.181573944	y2	x3	0.006358901	y3	z2	-0.2002	y4	z2	-0.389086436
y1	y3	0.191617913	y2	y3	-0.095987852	y3	x3	0.32843	y4	x3	0.042718189
y1	z3	0.109626835	y2	z3	0.120384491	y3	z3	0.67093	y4	y3	0.3239336
y1	x4	-0.29045831	y2	x4	0.081715165	y3	x4	-0.1112	y4	z3	0.076057385
y1	y4	0.228997237	y2	y4	-0.15437806	y3	y4	0.32393	y4	x4	-0.600982199
y1	z4	0.186791537	y2	z4	0.165709979	y3	z4	-0.0364	y4	z4	-0.117144404
z1	x1	0.030417499	z2	x1	0.251138704	z3	x1	0.00417	z4	x1	-0.025592835
z1	y1	-0.5159614	z2	y1	-0.0301789	z3	y1	0.10963	z4	y1	0.186791537
z1	x2	0.109970588	z2	z1	0.2172898	z3	z1	0.12427	z4	z1	-0.197813141
z1	y2	0.017749163	z2	x2	0.953083727	z3	x2	0.10752	z4	x2	0.164505939
z1	z2	0.2172898	z2	y2	0.918648041	z3	y2	0.12038	z4	y2	0.165709979
z1	x3	0.130282473	z2	x3	0.006650803	z3	z2	0.07906	z4	z2	0.160719903
z1	y3	-0.118961498	z2	y3	-0.200237814	z3	x3	0.27106	z4	x3	-0.202266103
z1	z3	0.124272757	z2	z3	0.07905866	z3	y3	0.67093	z4	y3	-0.036427754
z1	x4	0.50628732	z2	x4	0.287828127	z3	x4	0.03536	z4	z3	0.031079739
z1	y4	-0.405502292	z2	y4	-0.389086436	z3	y4	0.07606	z4	x4	-0.068008246
z1	z4	-0.197813141	z2	z4	0.160719903	z3	z4	0.03108	z4	y4	-0.117144404

Figura 5: Indice di correlazione di Paerson calcolato su tutte le le combinazioni di componenti

Successivamente abbiamo stimato la precisione del modello calcolando il rapporto tra le inferenze corrette su quelle totali. Il metodo utilizzato, *query*, si occupa di effettuare la singola previsione. Abbiamo notato che a parità di modello, raddoppiando il numero delle classi per ogni variabile, il tempo impiegato dalla funzione aumenta vertiginosamente, perciò abbiamo stimato la precisione del modello più complesso possibile ma con tempi di esecuzione nella norma.
Il modello consiste in cinque classi per ogni variabile e le seguenti dipendenze tra le variabili:
[('x1', 'class'),('x3', 'class'),('y4', 'class'), ('z1', 'class'), ('z2', 'class'), ('z3', 'class'), ('z4', 'class'), ('y1', 'z1'), ('x2', 'y2'), ('x2', 'z2'), ('y2', 'z2'), ('y3', 'z3'), ('x4', 'z1'), ('x4', 'y4')].
La precisione stimata è del 54%, un valore troppo basso per giustificare la computazione così lunga e dispendiosa di risorse. Abbiamo deciso così di tentare con un modello differente.

3.1.2 Generazione del modello

Dato l'insuccesso del modello correlato abbiamo cercato di generare la miglior configurazione di dipendenze secondo la libreria. Dopo il lancio della funzione che si

occupa di chiamare l'API per la generazione abbiamo atteso circa 8 ore e successivamente interrotto l'esecuzione. Forse per la mole di dati, forse per la discretizzazione del dataset o forse per la natura stessa dell'algoritmo, non vi è stato alcun risultato.

3.2 pomegranate

Alla luce dei test effettuati abbiamo deciso di utilizzare un'altra libreria: Pomegranate [3], secondo gli utenti di alcuni forum, dovrebbe essere performante anche nei casi in cui *pgmpy* non lo è. La libreria è nota inoltre per avere API molto simili a quelle di scikit-learn [4] (uno tra i software leader nel campo del Machine Learning)

3.2.1 Modello generato

Il modello proposto viene generato automaticamente dalla libreria tramite la funzione *from_samples* la quale genera gli stati della rete e le dipendenze basandosi sul training-set passato, dando all'utente la possibilità di esportare un file JSON (poco comprensibile) contente tutti i dati del modello. Oltre a questa non vi sono altre API per ottenere informazioni riguardo alla rete.
Nonostante la lunga attesa per poter ottenere i dati riguardo all'accuratezza del modello, i risultati sono più che soddisfacenti 2, tra cui Accuracy del modello: 97%.
Un fatto che ci ha particolarmente colpito del modello generato da Pomegranate è

Tabella 2: Score del modello

Campo	Precision	Recall	F1 Score
Walking	97%	97%	97%
Standing	99%	98%	98%
Standingup	90%	87%	90%
Sitting	99%	99%	99%
Sittingdown	86%	92%	89%

che in alcuni casi le inferenze venivano restituite dall'API senza essere state risolte. Ipotizziamo che quando la rete non è in grado di effettuare una decisione, restituisce *None* invece di assegnare una valore casuale al risultato. Queste particolari inferenze sono state ignorate nel calcolo delle performance dato che il numero di esse è estremamente ridotto, circa lo 0.09% delle totali.
Date le notevoli performance del modello, è stato sviluppato un software che permette di istanziare il modello e lanciare una o più inferenze, eseguibile tramite '*python inference.py "x1=-5;y1=30;z1=-20;x2=..."*' e viene restituito in output la predizione. Nel caso il parametro stringa venga omesso, il software elabora un array di stringhe in "inference/in.txt", le interpreta come query alla variabile *class* del modello e stampa in "inference/out.txt" i risultati.

4 Conclusioni

L'obiettivo di questo lavoro è inferire l'azione compiuta da un soggetto tramite i dati rilevati da un sistema di 4 accelerometri. La previsione è stata ottenuta mediante una rete Bayesiana che modella le dipendenze tra i sensori, visti come variabili casuali.

L'obiettivo è stato raggiunto tramite l'approccio *learning structure by data*, inizialmente tentando di individuare la struttura tramite uno studio di correlazione manuale tra i dati con esito negativo, successivamente generando automaticamente il modello grazie alle API delle librerie. Il modello finale riesce a inferire l'azione compiuta dal soggetto con una precisione molto alta, di circa 97%. Inoltre, i sample del dataset sono ben distribuiti tra le varie attività, quindi quando il modello viene generato, la fase di shuffle e di campionamento dei sample non influiscono sulla precisione nell'inferenza. Una difficoltà importante riportata nella progettazione è la mancata visualizzazione della struttura della rete. Infatti i risultati riportati sono stati ottenuti con la libreria [3] che si è scoperto essere sprovvista delle API per la visualizzazione dello scheletro e in generale poco documentata e supportata. L'unica funzione disponibile è l'esportazione tramite JSON che pero' risulta in un file difficilmente interpretabile.
Un altro fattore inaspettato che ci ha colpito negativamente sono i tempi di esecuzione delle due librerie utilizzate che, nonostante Pomegranate sia notevolmente più ottimizzato di Pgmpy, sono decisamente alti.

Riferimenti bibliografici

[1] http://groupware.les.inf.puc-rio.br/har.

[2] http://pgmpy.org/.

[3] https://pomegranate.readthedocs.io/.

[4] https://scikit-learn.org.

[5] Katia Vega Eduardo Velloso Ruy Milidiú Wallace Ugulino, Débora Cardador and Hugo Fuks. Wearable computing: Accelerometers' data classification of body postures and movements, 2012.