Apprendimento di Parametri in modelli Grafici

Edoardo Ramalli (5940744)

8 Gennaio 2018

1 Introduzione

L'elaborato si divide principalmente in due parti: Nella prima parte si sviluppa del codice in Python 2.7 per l'apprendimento di parametri in reti orientate utilizzando l'approccio a massima verosimiglianza con l'accortezza di usare laplace smoothing per evitare stime degeneri. Nella seconda parte si implementa, sempre in Python 2.7, una tecnica per generare datasets di n elementi partendo da una rete nota con distribuzione p. Dunque si utilizza il codice della prima parte per apprendere i parametri della rete partendo dal datasets di dimensione n ed ottenendo una distribuzione appresa q_n. Infine si misura quanto si discosta la distribuzione appresa da quella reale utilizzando la divergenza di Kullback-Liebler.

1.1 Rete Bayesiana

Una rete bayesiana è un DAG in cui ogni nodo è etichettato con un'informazione probabilistica quantitativa per rappresentare le dipendenze (Attraverso gli archi del grafo) tra varibili (Discrete o Continue) e fornire una specifica concisa di qualsiasi distribuzione di probabilità congiunta completa. Ogni nodo x_i ha una distribuzione di probabilità condizionata $P(x_i|Genitori(x_i))$, che quantifica gli effetti dei genitori sul nodo. Dunque ogni nodo possiede una CPT (Tabella della probabilità condizionata), in cui ogni rifga contiene la probabilità condizionata di tutti i valori del nodo per ogni singolo caso condizionante (Una possibile combinazione dei valori dei nodi dei genitori).
Una rete bayesiana fornisce una descrizione completa del dominio. Ogni elemento della distribuzione della probabilità congiunta può essere calcolato a partire dalle informazione nella rete:

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i|Genitori(x_i))$$

In questo elaborato viene utilizzata una rete con 12 nodi, le cui dipendenze sono riportarte attraverso il grafo, con le relative CPT nella figura sottostante.

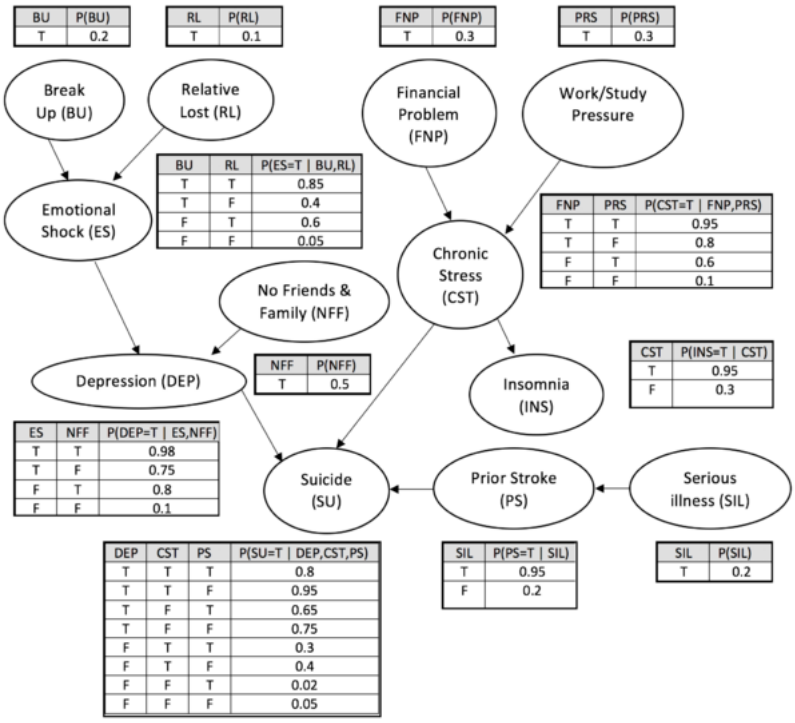

Figure 1: Rete Utilizzata

1.2 Massima Verosimiglianza e Laplace Smoothing

Il metodo della massima verosimiglianza è un procedimento per determinare uno stimatore. In particolare, in questo caso, si richiede di trovare i parametri numerici di un modello di probabilità che ha come struttura la rete bayesiana. In questo elaborato si assume che i dati forniti in ingresso per apprendere la rete siano completi e che le variabili in oggetto siano discrete.
Il principio della massima verosimiglianza vuole massimizzare la funzione di verosimiglianza dato un datasets D, in cui si indica con n i casi favorevoli e N il numero di campioni osservati: $P(D;\theta) = \prod_{i=1}^{n} P(x^{(i)};\theta) = \theta^n(1-\theta)^{N-n}$. Si ottiene dunque passando attraverso la verosimiglianza logaritmica e derivando rispetto ad ogni singolo parametro (i.i.d.) per massimizzare il tutto, ottenendo:

$$\theta_{ijk} = P(x_i = j | P_a(i) = k) = \frac{Nijk+1}{\sum_l N_{ilk}+2}$$

Si osserva che al numeratore è stato aggiunto 1 e al denominatore 2, in questa maniera si applica la tecnica del laplace smoothing per evitare stime degeneri su campioni piccoli di dati osservati, dunque si aggiunge 1 per far finta che ogni caso sia comparso almeno una volta e divido per due in quanto le variabili, in questo caso possono assumere valori, 0 o 1, assenza o presenza.

1.3 Divergenza di Kullback-Leibler

Viene utilizzata la divergenza di Kullback-Leibler per misurare quanto si discosta la distribuzione appresa su un datasets di dimensione n, da quella reale. Essa è definita come :

$$KL(p||q_n) = \sum_U p(U) \log \frac{p(U)}{q_n(U)}$$

2 Esperimento

Dopo aver inserito la struttura della rete in un'apposita classe Python, si eseguono 10 test per ogni dimensione in ingresso. Le dimensioni in ingresso sono : [100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800, 900, 1000, 1250, 1500, 2000]. Ogni test consiste di generare un datasets di dimensione n a partire dalla rete nota, rispettando ogni CPT di ogni nodo. Dopodichè partendo dal datasets si apprendono i parametri della rete ottenendo come risultato delle CPT "learned". Infine si misura la distanza delle distribuzioni con la divergenza KL iterando su tutte le possibili configurazioni di ingresso U. Per ogni test si ottiene 10 misure di divergenza di cui se ne fa la media e si calcola lo scarto quadratico medio per riportare il tutto in un grafico.

3 Implementazione

Sono stati usati tre file Pyhton: Function,BayesianNetwork, Test. Il primo file contiene funzioni di supporto per gli altri due file.
Il secondo file contiene al suo interno due classi: La prima "Node" rappresenta i nodi della rete ed ha come attributi il nome del nodo, il suo nickname, i suoi padri (Rappresentati come una lista di indici), la CPT e La CPT Learned. Questa classe ha alcune funzioni di supporto tra cui la addCPT che da un file .csv, importa in una matrice la CPT, e la funzione getCPT, che data un configurazione in ingresso dei nodi a cui il nodo stesso è condizionato, resistutisce il valore di probabilità congiunta corrispondente. La secondo classe è la classe Network. La classe Network non è altro che una collezione di nodi che acquisiscono un indice man mano che vengono inseriti nella rete. I nodi vanno inseriti nella rete con un certo ordine: prima i nodi che non hanno padri e poi a seguire i nodi che dipendono da altri nodi già inseriti nella rete. Essa contiene diverse funzioni. La funzione CreateDatset, riceve in ingresso un numero che rappresenta la dimensione del datasets da creare e utilizzando la funzione createRowofDatset, inserisce di volta in volta un tupla con tanti elementi quanti sono i nodi della rete e li memorizza riga per riga in un array. La funzione createRowofDataset, parte calcolando il valore (0 o 1 in questo caso) dei nodi senza padri rispettando la loro CPT, dopodichè per i nodi che hanno dei padri, si seleziona la probabilità condizionata del nodo a secondo dei valori assunti dai padri e si calcola quindi il valore di quel nodo. Ad esempio se un nodo con una

certa configurazione ha un probabilità condizionata θ, si estrae con probabilità uniforme tra 0 e 1 un valore; se questo è minore di θ allora il valore di quel nodo è 1 altrimenti 0. (Nelle CPT vengono riportati solo i valori "positivi" per calcolare il corrispettivo si applica $1 - \theta$). Una volta creato il datasets, questo viene dato in ingresso, come file .csv, alla funzione Learning di NetWork che per semplicità, visto che l'apprendimento dei parametri in questo caso deve conoscere la struttura della rete, è la stessa istanza di prima. Dunque seguendo sempre lo stesso ragionamento, ovvero partire dai nodi senza padre per poi ad arrivare a quelli con, si stima i paramtri delle CPT di ogni nodo applicando la massima verosimiglianza con laplace smoothing. Infine la funzione divergenza calcola la divergenza tra la CPT reale e quella appresa andando a sommare su tutte le possibili configurazioni di ingresso utilizzando la funzione CPD,che data una configurazione in ingresso calcola la distribuzione di probabilità congiunnta della rete.
Il terzo file crea l'istanza della classe Network, vi aggiunge i nodi con i loro parametri ed infine avvia i test e ne mostra i risultati in un grafico.

4 Risultati

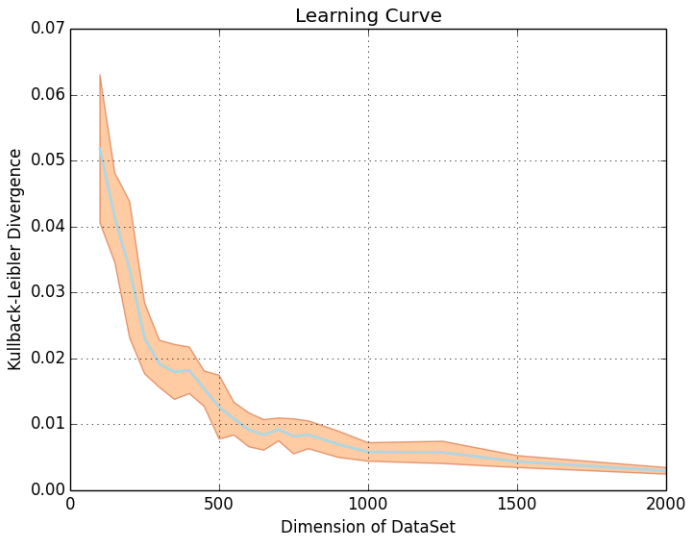

Figure 2: LearningCurve

Si mostra il risultato di una possibile esecuzione dei test. Dal Grafico si osserva, come preventivato, che la divergenza si riduce al crescere di n. Vengono riportati il valor medio, in azzurro, e la banda dello scarto quadratico medio in arancione.