Andrea Augello
Department of Engineering, University of Palermo, Italy

Reti Bayesiane 1

Università degli Studi di Palermo

Una rete bayesiana è un modello grafico probabilistico che rappresenta un insieme di variabili stocastiche con le loro dipendenze condizionali attraverso l'uso di un grafo aciclico diretto (DAG)

- I nodi del grafo rappresentano le variabili stocastiche
- Gli archi rappresentano le relazioni di dipendenza statistica tra le variabili e le distribuzioni locali di probabilità dei nodi figlio rispetto ai valori dei nodi padre.

bnlearn

bnlearn

- `bnlearn` è un pacchetto Python per l'apprendimento di reti bayesiane
- Consente di apprendere sia i parametri che la struttura di una rete bayesiana
  - Imparare i parametri di una rete bayesiana significa apprendere le distribuzioni di probabilità condizionale dei nodi figlio rispetto ai valori dei nodi padre
  - Imparare la struttura di una rete bayesiana significa apprendere le relazioni di dipendenza statistica tra le variabili stocastiche

https://erdogant.github.io/bnlearn/pages/html/index.html

Esempio 1

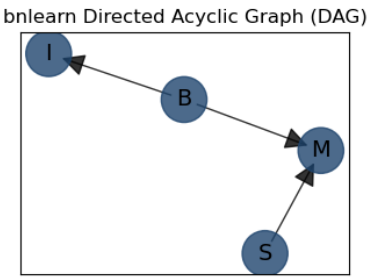
Costruire un grafo

Se si possiede già conoscenza a priori sulla struttura della rete bayesiana (o se stiamo usando un modello naive bayes), è possibile costruire il grafo a mano.

Esempio

Braccio robotico che tenta di sollevare un blocco:

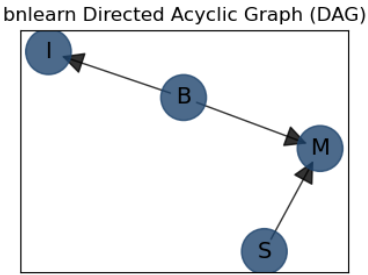
- M = c'è movimento
- B = la batteria è carica
- S = il blocco è sollevabile
- I = l'indicatore è acceso



Costruire un grafo

Per prima cosa dobbiamo definire le relazioni uno-ad-uno tra le variabili.
Definiamo gli archi:

- $S \to M$
- $B \to M$
- $B \to I$




```
import bnlearn as bn
from pgmpy.factors.discrete import
    TabularCPD

# Define the structure.

edges = [('S','M'), ('B','M'), ('B','I')]
DAG = bn.make_DAG(edges)
bn.plot(DAG)
```

Aggiungere le distribuzioni di probabilità

Al momento il grafo non contiene nessuna distribuzione condizionale di probabilità, quindi non può essere usato per fare inferenza.
Specifichiamo manualmente le tabelle delle distribuzioni di probabilità condizionale per ogni nodo figlio rispetto ai valori dei nodi padre.

$p(I\|\neg B)$	$0,1$
$p(I\|B)$	0.95

$p(M\|\neg B, \neg S)$	0
$p(M\|\neg B, S)$	0
$p(M\|B, \neg S)$	0.05
$p(M\|B, S)$	0.9

$p(S)$	0.95

$p(B)$	0.9

```
cpt_i = TabularCPD(variable='I', variable_card=2,
                   values=[[0.9, 0.05],
                           [0.1, 0.95]],
                   evidence=['B'], evidence_card=[2])

cpt_m = TabularCPD(variable='M', variable_card=2,
                   values=[[1, 1, 0.95, 0.1],
                           [0, 0, 0.05, 0.9]],
                   evidence=['B', 'S'], evidence_card=[2, 2])

cpt_s = TabularCPD(variable='S', variable_card=2,
                   values=[[0.05],[0.95]])
cpt_b = TabularCPD(variable='B', variable_card=2,
                   values=[[0.1],[0.9]])
```

Inferenza sul grafo

Per fare inferenza sul grafo, dobbiamo specificare le evidenze e quali variabili vogliamo calcolare.

Esempi di inferenza:

$p(M\|I) = 0.85$	`bn.inference.fit(DAG, variables=['M'], evidence={'I':1})`
$p(M\|B) = 0.86$	`bn.inference.fit(DAG, variables=['M'], evidence={'B':1})`
$p(M\|S) = 0.81$	`bn.inference.fit(DAG, variables=['M'], evidence={'S':1})`
$p(M\|S, I) = 0.89$	`bn.inference.fit(DAG, variables=['M'], evidence={'S':1, 'I':1})`
$p(M\|S, \neg I, B) = 0.90$	`bn.inference.fit(DAG, variables=['M'], evidence={'S':1, 'I':0, 'B':1})`

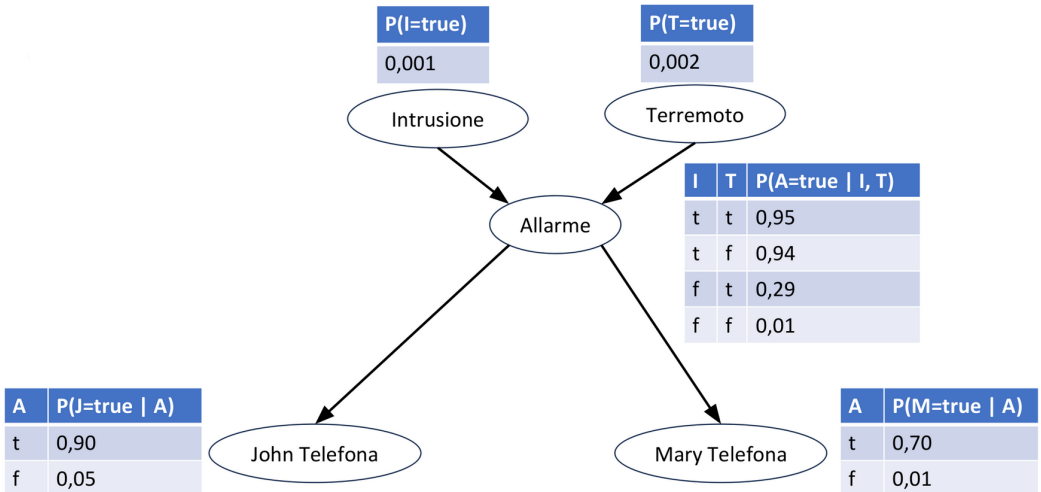
Esempio 2

Esempio 2

- Abbiamo installato un nuovo antifurto, che è abbastanza affidabile ma occasionalmente scatta anche per piccoli terremoti
- Abbiamo due vicini, John e Mary, che hanno promesso di telefonare se scatta l'allarme
- John chiama quasi sempre quando scatta l'allarme, ma a volte scambia lo squillo del telefono per l'allarme
- Mary non sempre sente l'allarme

Variabili: `['Intrusione', 'Terremoto', 'Allarme', 'John', 'Mary']`
Inferenza: `bn.inference.fit(DAG, variables=['Intrusione'], evidence={'John':1, 'Mary':0})`

Esempio 2


P(I=true)
0,001
P(T=true)
0,002
Intrusione
Terremoto
Allarme
I T P(A=true | I, T)
t t 0,95
t f 0,94
f t 0,29
f f 0,01
A P(J=true | A)
t 0,90
f 0,05
John Telefona
Mary Telefona
A P(M=true | A)
t 0,70
f 0,01

Soluzione

```
edges = [('Intrusione', 'Allarme'), ('Terremoto', 'Allarme'), ('Allarme', 'John'),
    ('Allarme', 'Mary')]

cpt_intrusione = TabularCPD(variable='Intrusione', variable_card=2,
                            values=[[0.999], [0.001]])
cpt_terremoto  = TabularCPD(variable='Terremoto', variable_card=2,
                            values=[[0.998], [0.002]])

cpt_allarme    = TabularCPD(variable='Allarme', variable_card=2,
                            values=[[0.999, 0.71, 0.06, 0.05],
                                    [0.001, 0.29, 0.94, 0.95]],
                            evidence=['Intrusione', 'Terremoto'],
                            evidence_card=[2, 2])

cpt_john       = TabularCPD(variable='John', variable_card=2,
                            values=[[0.95, 0.1],
                                    [0.05, 0.9]],
                            evidence=['Allarme'], evidence_card=[2])

cpt_mary       = TabularCPD(variable='Mary', variable_card=2,
                            values=[[0.99, 0.3],
                                    [0.01, 0.7]],
                            evidence=['Allarme'], evidence_card=[2])

dag = bn.make_DAG(edges, CPD=[cpt_intrusione, cpt_terremoto, cpt_allarme, cpt_john,
    cpt_mary])
```

Apprendere le CPD

Apprendere le CPD

Abbiamo potuto vedere che specificare manualmente le distribuzioni di probabilità condizionale è un processo lungo, noioso, e soggetto a errori.

Siamo quindi interessati ad un metodo automatico per apprendere le distribuzioni di probabilità condizionale a partire da un dataset.

Fortunatamente, una volta che abbiamo specificato la struttura del grafo, `bnlearn` può apprendere le distribuzioni di probabilità condizionale a partire da un dataset in modo estremamente semplice.

```
bn.parameter_learning.fit(DAG, data, methodtype='maximumlikelihood')
```

Esempio 3

Esempio 3

Abbiamo un dataset contenente informazioni sulle condizioni meteo e se si sia giocato a tennis o meno.

Rain	Humid	Wind	Play
False	False	False	True
False	True	False	False
False	False	True	False
False	True	True	False
False	True	False	True
		…	

```
def create_tennis_data(rows=100):
    dir_path = os.path.dirname(os.path.abspath(__file__))
    with open(dir_path+"/tennis.csv", "w") as f:
        f.write("rain,humid,wind,play\n")
        for _ in range(rows):
            rain  = random.choice([0, 0, 0, 1])
            humid = random.choice([0, 1, 1])
            wind  = random.choice([0, 1])

            if rain == 1:
                play = 1 if random.random() < 0.01 else 0
            elif humid == 1 and wind == 0:
                play = 1 if random.random() < 0.35 else 0
            elif humid == 0 and wind == 0:
                play = 1 if random.random() < 0.9 else 0
            elif humid == 1 and wind == 1:
                play = 1 if random.random() < 0.5 else 0
            else:
                play = 1 if random.random() < 0.7 else 0
            f.write(f"{rain},{humid},{wind},{play}\n")
```

Esempio 3

Nota bene

`bnlearn` si aspetta che il dataset sia fornito come dataframe pandas. Salviamo quindi il file in formato CSV, con la prima riga che contiene i nomi delle colonne.

```
import pandas as pd

def read_data():
    dir_path = os.path.dirname(os.path.abspath(__file__))
    data = pd.read_csv(dir_path+"/tennis.csv")
    return data
```

Esempio 3

Per prima cosa dobbiamo definire la struttura del grafo.

Dato che vogliamo usare un modello naive bayes, gli unici archi che dobbiamo definire sono quelli che collegano il nodo target (Play) ai nodi delle feature (Rain, Humid, Wind).

Tutti gli archi dovranno partire dal nodo target e arrivare ai nodi delle feature (questo può sembrare controintuitivo, ma la dipendenza statistica non è necessariamente una relazione causale).

Esempio 3

Per prima cosa dobbiamo definire la struttura del grafo.

Dato che vogliamo usare un modello naive bayes, gli unici archi che dobbiamo definire sono quelli che collegano il nodo target (Play) ai nodi delle feature (Rain, Humid, Wind).

Tutti gli archi dovranno partire dal nodo target e arrivare ai nodi delle feature (questo può sembrare controintuitivo, ma la dipendenza statistica non è necessariamente una relazione causale).

```
edges = [('play', 'rain'), ('play', 'humid'), ('play', 'wind')]
DAG = bn.make_DAG(edges)
```

Apprendere le CPD

Una volta che abbiamo definito la struttura del grafo, possiamo apprendere le tabelle di probabilità condizionale necessarie per fare inferenza a partire dal dataset.

- `bn.parameter_learning.fit(DAG, data, methodtype='maximumlikelihood')`
- `bn.parameter_learning.fit(DAG, data, methodtype='bayes', scoretype='k2')`

Per visualizzare le tabelle di probabilità condizionale apprese:

- `bn.print_CPD(DAG)`

Adesso possiamo fare inferenza sul grafo.