Apprendere Struttura di un Grafo Bayesiano

Guglielmo Fratticioli

Giugno 2020

Abstract

Il progetto si propone di sviluppare un codice per l'apprendimento della struttura di un grafo , in secondo luogo si procede al campionamento di un modello noto e quindi all'apprendimento della basato sul dataset estratto

1 Sviluppo Codice

1.1 Teoria

Un modello grafico probabilistico modellizza un particolare contesto di eventi stocastici legati fra loro da dipendenze condizionali . Si può quindi pensare ad un grafo diretto che codifica gli eventi come nodi e le dipendenze condizionali come archi fra i nodi. è importante che il grafo non sia Ciclico (è assurdo pensare ad un evento che condiziona se stesso) .
Ad ogni particolare configurazione della rete è associata una probabilità congiunta degli eventi che si può fattorizzare secondo Bayes per ogni dipendenza condizionale, avremo quindi :

$$P(E_1, E_2, ...E_n) = \prod_{i=1}^{n} P(E_i|Pa(E_i)) = \prod_{i=1}^{n} \prod_{k=1}^{d_i} \prod_{j} P(E_i = k|Pa(E_i) = j)$$

Supposto che i parametri si dispongano secondo distrubuzione Multinomiale

$$P(E_i|\theta_1, ...\theta_{d_i}) = \theta_1^{k_1}\theta_2^{k_2}... \ \theta_N^{k_N}$$

Possiamo stimare tramite conteggio nel dataset le probabilità

$$P(E_i = k|Pa(E_i) = j) = n_{ij} = \frac{\{\#\text{of examples where } E_i = k \text{ while } Pa(E_i) \text{ in config } j\}}{\{\#\text{of examples where} Pa(E_i) \text{in config} j\}}$$

assumendo che anche la Struttura sia una variabile aleatoria dove probabilità è totalmente concentrata nel massimo $S^\star$

$$P(E_i|D) = P(S^\star|D) \int P(E_i|\theta, D, S^\star)P(\theta|D, S^\star)d\theta \quad (*)$$

Secondo Bayes

$$P(S|D) = \frac{P(S)P(D|S)}{P(D)} \ =$$

Siamo interessati a massimizzare la marginial likehood $P(D|S) \propto P(S|D)$ Cooper e Herskoviz hanno ricavato la seguente espressione dalla relazione (*)

$$P(D|S) \propto \prod_{1}^{N} \prod_{k=1}^{d_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + n_{ij})} \prod_{j}^{q_i} \frac{\Gamma(\alpha_{ijk} + n_{ijk})}{\Gamma(\alpha_{ijk})}$$

che può essere quindi usata come Scoring per una ricerca della struttura

1.2 Modellizzazione

Linguaggio : python

- Node : ogni nodo memorizza una tabella dei parametri di probabilità, ha una lista di Nodi padri e Figli , un etichetta ed un nome
- Graph :

```
class Graph:
    def __init__(self, nodes):
        self.nodes = nodes
```

  il grafico è memorizzato come una lista di Nodi, sono implementate varie funzioni

  - AddEdge()
  - RemoveEdge()
  - invertEdge()
  - isCyclic()

- Dataset :

  è una lista di Esempi associati ad una lista di nodi , ogni Esempio è una particolare configurazione di valori assunta dai nodi

2 Campionamento

ho campionato gli esempi secondo il modello ALARM secondo un ordinamento topologico, quindi si inizia dalle sorgenti :

```
lvfail =  random.choices(population=[TRUE,FALSE], weights= [
    LVFAILURE.table.get('SELF')[0], LVFAILURE.table.get('SELF')[1] ] )

hypovelmia = random.choices(population=[TRUE,FALSE], weights= [
    HYPOVOLEMIA.table.get('SELF')[0], HYPOVOLEMIA.table.get('SELF')[1] ] )
```

e poi si procede a campionare i Nodi Figlio in base ai valori estratti precedentemente

```
lveovolume =  random.choices(population=[LOW3,NORMAL3,HIGH3],weights= [
                    LVEOVOLUME.table.get((hypovelmia[0],lvfail[0]))[0],
                    LVEOVOLUME.table.get((hypovelmia[0],lvfail[0]))[1],
                    LVEOVOLUME.table.get((hypovelmia[0],lvfail[0]))[2]  ] )
```

3 Apprendimento

Si inizializza un DAG casuale sulle variabili e si esegue la ricerca alterando un arco nel grafo , a patto che cambi la V-STructure. Si inserisce quindi in una lista tutti i possibili successori validi e si sceglie quello che ha scoring migliore

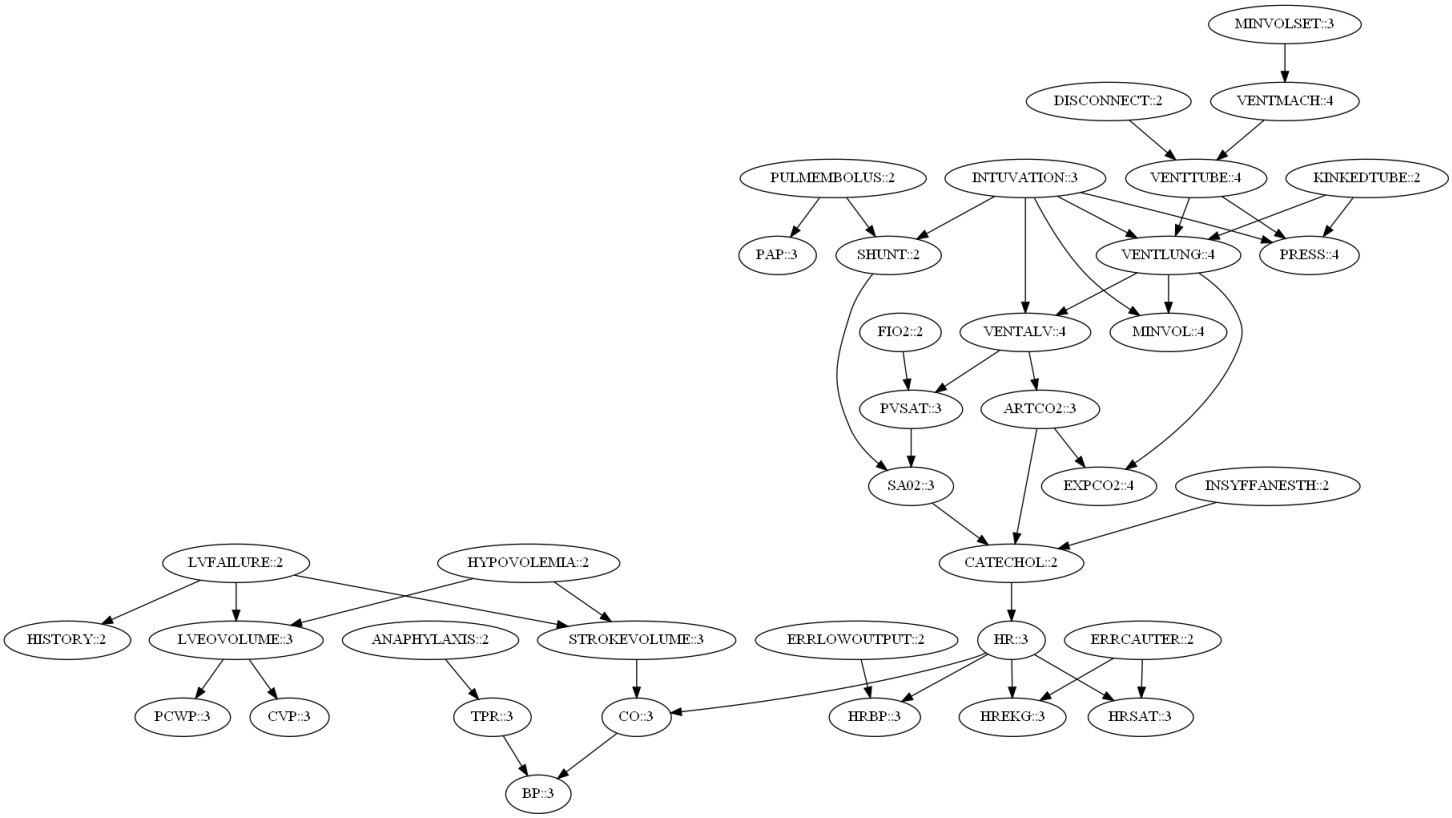
```
def Learn(graph, dataset):
    graph.initDAG()
    current = Score( graph , dataset)
    ...
    while(run):
        run = False
        vstruct = graph.VStruct()
        G = []
        S = []
        for i in range(len(graph.nodes)):
            for j in range(i, len(graph.nodes)) :
                if i != j:
                    g1 = copy.deepcopy(graph)
                    ...
                    g1.addEdge(g1.nodes[i], g1.nodes[j])
                    g2.removeEdge(g2.nodes[i], g2.nodes[j])
                    g3.invertEdge(g3.nodes[i], g3.nodes[j])

                    if g1.VStruct() != vstruct and not g1.isCyclic():
                        S.append(Score(g1, dataset))
                        G.append(g1)
                    if g2.VStruct() != vstruct and not g2.isCyclic():
                        ...
                    ...
        if len(S) > 0:
            score = Score(G[S.index(max(S))], dataset)
            if  score > current :
                graph = G[S.index(max(S))]
                current = score
                run = True
```

Lo Scoring secondo Cooper e Herskoviz :

```
def Score(graph, data):
score = 0
for node in graph.nodes:
    if len(node.fathers) != 0:
        j = make_j(node)
        for comb in j[0]:
            score = score + math.log(math.gamma(alphaij(node, comb))) -
                    math.log( math.gamma(alphaij(node, comb) +
                        Dataset.Nij(data, node, [comb, j[1]])) )
            for k in range(node.domine):
                score = score + math.log(math.gamma(alphaijk(node, comb, k) +
                    Dataset.Nijk(data, node, [comb, j[1]], k)) -
                       math.log(math.gamma(alphaijk(node, comb, k))))
return score
```

4 Results



- The ALARM Model; Sampled with 500 examples gives Score : -4152.68

- The Learnt Structure; Learned with 500 examples gives Score : -5176.78