Exact Inference with Probabilistic Graphical Models

Make exact inferences about probabilistic graphical models using the state-of-the-art graphical model packages in Python.

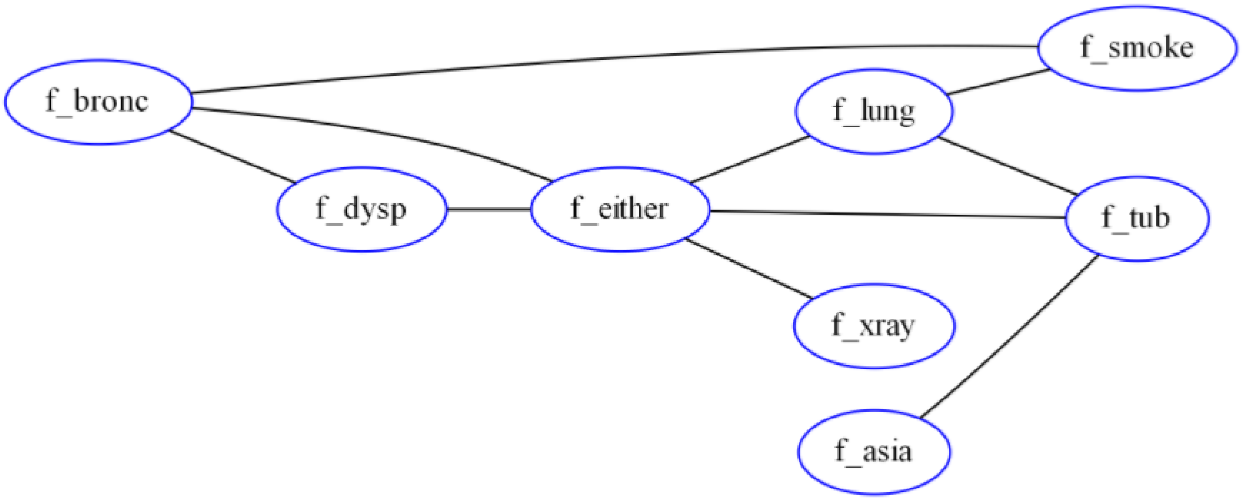
1. Draw the moral graph, triangulated graph and the junction tree. Explain why the "running intersection property" is satisfied in your junction tree.
2. Describe how the different terms are distributed among the different junction tree clusters. Write out the messages using these terms and verify that the message passing algorithm gives the cluster marginals.

Tools & Liblaries: Python 2, Bayesian Belief Networks

1 Drawing a graph

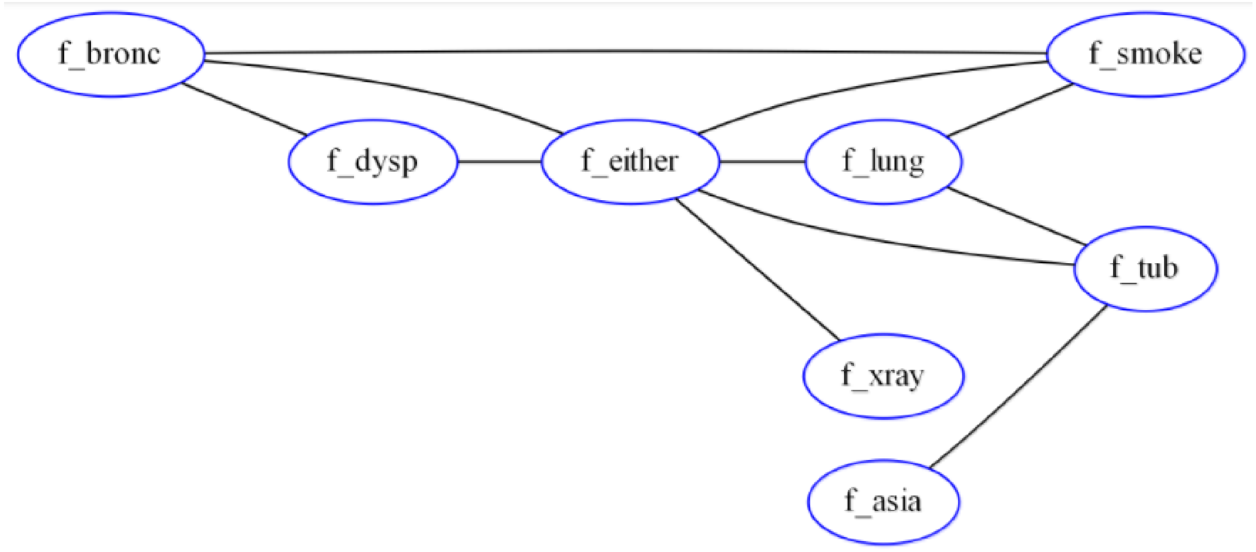
Moral Graph

Moralization converts a Bayesian network into an undirected graphical model and insert an undirected edge between the nodes, that have a child in common.



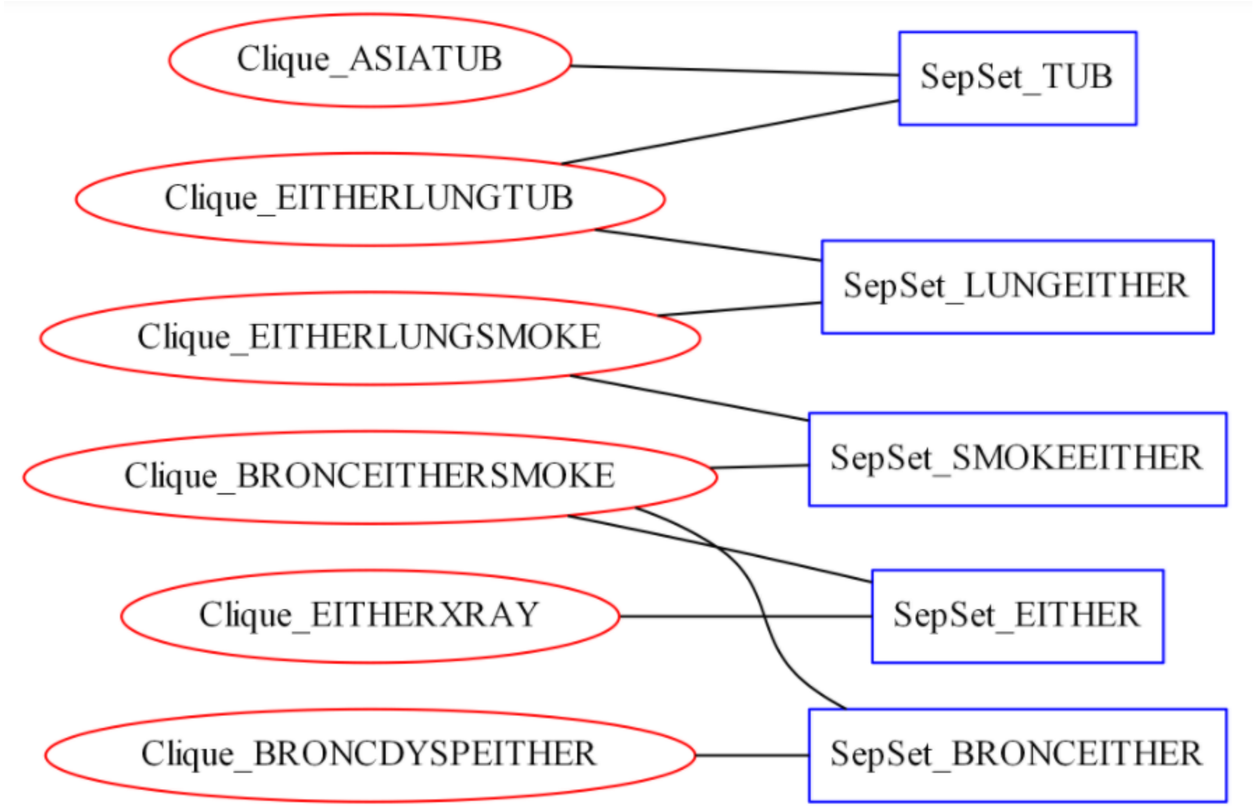
Triangulated graph

On this stage, the undirected graph is triangulated, which is the addition of edges between nodes if there is a cycle whose length is greater than 4. Thus we are adding edge "smoke-either", since the cycle bronc-smoke-lung-either is consist of four nodes.



Junction tree

On this stage, we create cliques according to triangulated graph and creates the sepsets which are the intersection points between every pair of cliques.



”Running intersection property” states that if X_i, X_j and X_k are nodes, and X_k is on the path from X_i to X_j, then $X_i \cap X_j \subseteq X_k$.

This property is satisfied in our junction tree, as follows:

$Asia - Tub \cap Either - Lung - Tub \subseteq Tub$

$Either - Lung - Tub \cap Either - Lung - Smoke \subseteq Lung - Either$

$Either - Lung - Smoke \cap Bronc - Either - Smoke \subseteq Smoke - Either$

$Bronc - Either - Smoke \cap Bronc - Dysp - Either \subseteq Bronc - Either$

$Either - XRay \cap Bronc - Dysp - Either \subseteq Either$

2 Message passing

Message passing stage ensure that the Junction Tree Algorithm sends messages between cliques and separators until consistency is reached. When convergence is reached clique potentials are marginals and separator potentials are submarginals.

$p(V) = p(a)p(t|a)p(s)p(l|s)p(b|s)p(e|t,l)p(d|e,b)p(x|e)$

Probability distributed among the different Junction tree clusters as follows:

$p(V) = \frac{1}{Z}\frac{\Pi_C \psi(x_C)}{\Pi_S \phi(x_S)}$

$p(V) = \frac{1}{Z}\frac{\psi(_{A,T})\psi(_{E,L,T})\psi(_{E,L,S})\psi(_{B,E,S})\psi(_{E,X})\psi(_{B,D,E})}{\phi(_T)\phi(_{EL})\phi(_{ES})\phi(_E)\phi(_{BE})}$

Message propagation:

Assume, that clique "BDE" is a root, then:

Forward Pass

$\phi^*_{(E)} = \sum_x \psi(_{EX})$

$\phi^*_{(BE)} = \sum_D \psi(_{BDE})$

$\psi^*_{(BES)} = \frac{\phi*_{(BE)}}{\phi_{(BE)}}\frac{\phi*_{(E)}}{\phi_{(E)}}\psi_{(BES)}$

$\phi*_{(ES)} = \sum_B \psi^*_{(BES)}$

$\psi^*_{(ELS)} = \frac{\phi*_{(ES)}}{\phi_{(ES)}}\psi_{(ELS)}$

$\phi^*_{(EL)} = \sum_S \psi^*_{(ELS)}$

$\psi^*_{(ELT)} = \frac{\phi*_{(EL)}}{\phi_{(EL)}}\psi_{(ELT)}$

$\phi^*_{(T)} = \sum_{E,T} \psi^*_{(ELT)}$

$\psi^*_{(AT)} = \frac{\phi*_{(T)}}{\phi_{(T)}}\psi_{(AT)}$

Reverse Pass:

$\phi^{**}_{(T)} = \sum_{A} \psi^*_{(AT)}$

$\psi^{**}_{(ELT)} = \frac{\phi^{**}_{(T)}}{\phi^*_{(T)}}\psi^*_{(ELT)}$

$\phi^*_{(EL)}* = \sum_{T} \psi^{**}_{(ELT)}$

$\psi^{**}_{(ELS)} = \frac{\phi^{**}_{(EL)}}{\phi^*_{(EL)}}\psi^*_{(ELS)}$

$\phi^*_{(ES)}* = \sum_{L} \psi^{**}_{(ELS)}$

$\psi^{**}_{(BES)} = \frac{\phi^{**}_{(ES)}}{\phi^*_{(ES)}}\psi^*_{(BES)}$

$\phi^{**}_{(BE)} = \sum_{S} \psi^{**}_{(BES)}$

$\psi^{**}_{(BDE)} = \frac{\phi^{**}_{(BE)}}{\phi^*_{(BE)}}\psi^*_{(BDE)}$

$\phi^{**}_{(E)} = \sum_{BS} \psi^{**}_{(BES)}$

$\psi^{**}_{(EX)} = \frac{\phi^{**}_{(E)}}{\phi^*_{(E)}}\psi^*_{(EX)}$

Numerical results of message passing:

Sepsetnote - ”E”

```
((’either’, ’no’),): 1.0
((’either’, ’yes’),): 1.0
```

Target potential - ”BDE”:

```
((’bronc’, ’yes’), (’dysp’, ’yes’), (’either’, ’no’)): 0.8
((’bronc’, ’no’), (’dysp’, ’yes’), (’either’, ’no’)): 0.1
((’bronc’, ’yes’), (’dysp’, ’no’), (’either’, ’yes’)): 0.1
((’bronc’, ’no’), (’dysp’, ’no’), (’either’, ’yes’)): 0.3
((’bronc’, ’no’), (’dysp’, ’no’), (’either’, ’no’)): 0.9
((’bronc’, ’yes’), (’dysp’, ’no’), (’either’, ’no’)): 0.2
((’bronc’, ’no’), (’dysp’, ’yes’), (’either’, ’yes’)): 0.7
((’bronc’, ’yes’), (’dysp’, ’yes’), (’either’, ’yes’)): 0.9
```

Sepsetnote - ”BE”

```
((’bronc’, ’no’), (’either’, ’no’)): 1.0
((’bronc’, ’yes’), (’either’, ’no’)): 1.0
((’bronc’, ’yes’), (’either’, ’yes’)): 1.0
((’bronc’, ’no’), (’either’, ’yes’)): 1.0
```

Target potential - ”BES”

```
((’bronc’, ’no’), (’either’, ’yes’), (’smoke’, ’no’)): 0.35
((’bronc’, ’yes’), (’either’, ’no’), (’smoke’, ’yes’)): 0.3
((’bronc’, ’no’), (’either’, ’no’), (’smoke’, ’yes’)): 0.2
((’bronc’, ’yes’), (’either’, ’yes’), (’smoke’, ’yes’)): 0.3
((’bronc’, ’yes’), (’either’, ’yes’), (’smoke’, ’no’)): 0.15
((’bronc’, ’no’), (’either’, ’no’), (’smoke’, ’no’)): 0.35
((’bronc’, ’yes’), (’either’, ’no’), (’smoke’, ’no’)): 0.15
((’bronc’, ’no’), (’either’, ’yes’), (’smoke’, ’yes’)): 0.2
```

Sepsetnote - ”ES”

```
((’either’, ’yes’), (’smoke’, ’no’)): 0.5
((’either’, ’no’), (’smoke’, ’no’)): 0.5
((’either’, ’no’), (’smoke’, ’yes’)): 0.5
((’either’, ’yes’), (’smoke’, ’yes’)): 0.5
```

Target potential - ”ELS”

```
((’either’, ’no’), (’lung’, ’no’), (’smoke’, ’no’)): 0.495
((’either’, ’yes’), (’lung’, ’yes’), (’smoke’, ’no’)): 0.005
((’either’, ’yes’), (’lung’, ’no’), (’smoke’, ’yes’)): 0.45
((’either’, ’no’), (’lung’, ’yes’), (’smoke’, ’no’)): 0.005
((’either’, ’no’), (’lung’, ’yes’), (’smoke’, ’yes’)): 0.05
((’either’, ’yes’), (’lung’, ’no’), (’smoke’, ’no’)): 0.495
((’either’, ’no’), (’lung’, ’no’), (’smoke’, ’yes’)): 0.45
((’either’, ’yes’), (’lung’, ’yes’), (’smoke’, ’yes’)): 0.05
```

Sepsetnote - ”EL”

```
((’either’, ’yes’), (’lung’, ’no’)): 0.9450000000000001
((’either’, ’no’), (’lung’, ’no’)): 0.9450000000000001
((’either’, ’no’), (’lung’, ’yes’)): 0.055
((’either’, ’yes’), (’lung’, ’yes’)): 0.055
```

Target potential - ”ELT”

```
((’either’, ’yes’), (’lung’, ’no’), (’tub’, ’yes’)): 0.9450000000000001
((’either’, ’no’), (’lung’, ’yes’), (’tub’, ’yes’)): 0
((’either’, ’yes’), (’lung’, ’yes’), (’tub’, ’yes’)): 0.055
((’either’, ’yes’), (’lung’, ’no’), (’tub’, ’no’)): 0
((’either’, ’no’), (’lung’, ’no’), (’tub’, ’yes’)): 0
((’either’, ’no’), (’lung’, ’no’), (’tub’, ’no’)): 0.9450000000000001
```

```
((’either’, ’yes’), (’lung’, ’yes’), (’tub’, ’no’)): 0.055
((’either’, ’no’), (’lung’, ’yes’), (’tub’, ’no’)): 0
```

Sepsetnote - ”T”

```
((’tub’, ’yes’),): 1.0
((’tub’, ’no’),): 1.0
```

Target potential - ”AT”

```
((’asia’, ’yes’), (’tub’, ’no’)): 0.0095
((’asia’, ’yes’), (’tub’, ’yes’)): 0.0005
((’asia’, ’no’), (’tub’, ’yes’)): 0.0099
((’asia’, ’no’), (’tub’, ’no’)): 0.9801
```

Reverse Pass

Sepsetnote - ”T”

```
((’tub’, ’yes’),): 0.010400000000000001
((’tub’, ’no’),): 0.9895999999999999
```

Target potential - ”ELT”

```
((’either’, ’no’), (’lung’, ’no’), (’tub’, ’no’)): 0.935172
((’either’, ’no’), (’lung’, ’yes’), (’tub’, ’yes’)): 0
((’either’, ’yes’), (’lung’, ’yes’), (’tub’, ’yes’)): 0.000572
((’either’, ’yes’), (’lung’, ’no’), (’tub’, ’no’)): 0
((’either’, ’no’), (’lung’, ’no’), (’tub’, ’yes’)): 0
((’either’, ’yes’), (’lung’, ’no’), (’tub’, ’yes’)): 0.009828000000000002
((’either’, ’yes’), (’lung’, ’yes’), (’tub’, ’no’)): 0.054428
((’either’, ’no’), (’lung’, ’yes’), (’tub’, ’no’)): 0
```

Sepsetnote - ”EL”

```
((’either’, ’yes’), (’lung’, ’no’)): 0.009828000000000002
((’either’, ’no’), (’lung’, ’no’)): 0.935172
((’either’, ’no’), (’lung’, ’yes’)): 0.0
((’either’, ’yes’), (’lung’, ’yes’)): 0.055
```

Target potential - ”ELS”

```
((’either’, ’no’), (’lung’, ’no’), (’smoke’, ’no’)): 0.48985199999999995
((’either’, ’yes’), (’lung’, ’yes’), (’smoke’, ’no’)): 0.005
((’either’, ’yes’), (’lung’, ’no’), (’smoke’, ’yes’)): 0.004680000000000001
((’either’, ’no’), (’lung’, ’yes’), (’smoke’, ’no’)): 0.0
((’either’, ’no’), (’lung’, ’yes’), (’smoke’, ’yes’)): 0.0
((’either’, ’no’), (’lung’, ’no’), (’smoke’, ’yes’)): 0.44532
((’either’, ’yes’), (’lung’, ’no’), (’smoke’, ’no’)): 0.005148000000000001
((’either’, ’yes’), (’lung’, ’yes’), (’smoke’, ’yes’)): 0.05
```

Sepsetnote - ”ES”

```
(('either', 'yes'), ('smoke', 'no')): 0.010148
(('either', 'no'), ('smoke', 'no')): 0.48985199999999995
(('either', 'no'), ('smoke', 'yes')): 0.44532
(('either', 'yes'), ('smoke', 'yes')): 0.054680000000000006
```

Target potential - ”BES”

```
(('bronc', 'no'), ('either', 'yes'), ('smoke', 'no')): 0.0071036
(('bronc', 'yes'), ('either', 'no'), ('smoke', 'yes')): 0.267192
(('bronc', 'no'), ('either', 'no'), ('smoke', 'yes')): 0.178128
(('bronc', 'yes'), ('either', 'yes'), ('smoke', 'yes')): 0.032808000000000004
(('bronc', 'yes'), ('either', 'yes'), ('smoke', 'no')): 0.0030444
(('bronc', 'no'), ('either', 'no'), ('smoke', 'no')): 0.34289639999999993
(('bronc', 'yes'), ('either', 'no'), ('smoke', 'no')): 0.1469556
(('bronc', 'no'), ('either', 'yes'), ('smoke', 'yes')): 0.021872000000000003
```

Sepsetnote - ”BE”

```
(('bronc', 'no'), ('either', 'no')): 0.5210243999999999
(('bronc', 'yes'), ('either', 'no')): 0.41414759999999995
(('bronc', 'yes'), ('either', 'yes')): 0.035852400000000006
(('bronc', 'no'), ('either', 'yes')): 0.028975600000000004
```

Target potential - ”BDE”

```
(('bronc', 'yes'), ('dysp', 'yes'), ('either', 'no')): 0.33131807999999996
(('bronc', 'no'), ('dysp', 'yes'), ('either', 'no')): 0.05210244
(('bronc', 'yes'), ('dysp', 'no'), ('either', 'yes')): 0.003585240000000001
(('bronc', 'no'), ('dysp', 'no'), ('either', 'yes')): 0.008692680000000001
(('bronc', 'no'), ('dysp', 'no'), ('either', 'no')): 0.46892195999999997
(('bronc', 'yes'), ('dysp', 'no'), ('either', 'no')): 0.08282951999999999
(('bronc', 'no'), ('dysp', 'yes'), ('either', 'yes')): 0.020282920000000003
(('bronc', 'yes'), ('dysp', 'yes'), ('either', 'yes')): 0.03226716000000001
```

Sepsetnote - ”E”

```
(('either', 'no'),): 0.9351719999999999
(('either', 'yes'),): 0.06482800000000002
```

Target potential - ”EX”

```
(('either', 'yes'), ('xray', 'yes')): 0.06353144000000002
(('either', 'yes'), ('xray', 'no')): 0.0012965600000000004
(('either', 'no'), ('xray', 'no')): 0.8884133999999999
(('either', 'no'), ('xray', 'yes')): 0.0467586
```

When done with all the message passing, we are left with a tree that has consistent beliefs in all its clusters.To verify it we can call the marginal probability for any variable.
Hence, the junction tree is consistent, a variable in any clique, where it belongs to, will have

the same marginals.
To prove that, we will analyze the marginal probability for variables, that are belonging to multiple cliques.

Either
Clique "Either-Xray"

Either: yes 0.064828 no 0.935172

Clique "Bronc-Dysp-Either"

Either: yes 0.064828 no 0.935172

Clique "Bronc-Either-Smoke"

Either: yes 0.064828 no 0.935172

Clique "Either-Lung-Tub"

Either: yes 0.064828 no 0.935172

Clique "Either-Lung-Smoke"

Either: yes 0.064828 no 0.935172

Bronc

Clique "Bronc-Dysp-Either"

Bronc: yes 0.45 no 0.55

Clique "Bronc-Either-Smoke"

Bronc: yes 0.45 no 0.55

Smoke
Clique "Bronc-Either-Smoke"

Smoke: yes 0.5 no 0.5

Clique "Either-Lung-Smoke"

Smoke: yes 0.5 no 0.5

Tub
Clique "Asia-Tub"

Tub: yes 0.0104 no 0.9896

Clique "Either-Lung-Tub"

Tub: yes 0.0104 no 0.9896

Lung
Clique "Either-Lung-Tub"

Lung: yes 0.055 no 0.945

Clique "Either-Lung-Smoke"

Lung: yes 0.055 no 0.945

Thus we confirmed that the Tree Junction Algorithm returns cluster marginals for each variable.