Modelling

November 10, 2021

[13]:
```
import networkx as nx
from matplotlib import pyplot as plt
```

1 Import raw data as ...

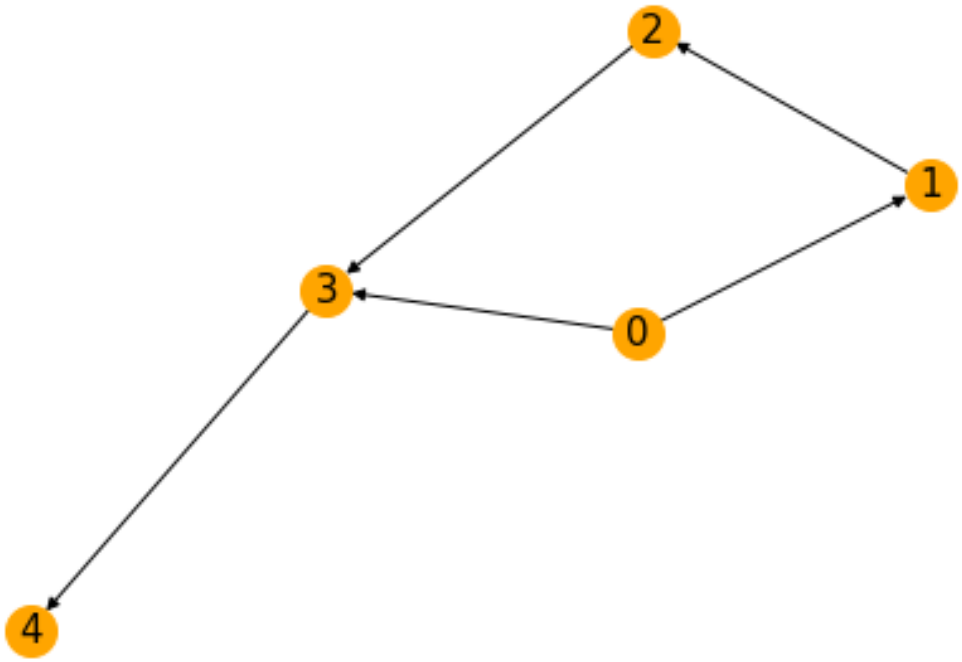
[1]:
```
import numpy as np

from pybbn.generator.bbngenerator import generate_multi_bbn,␣
↪convert_for_exact_inference, convert_for_drawing

# very important to set the seed for reproducible results
np.random.seed(37)

# this method generates the graph, g, and probabilities, p
# note we are generating a multi-connected graph
g, p = generate_multi_bbn(5, max_iter=5)
# you have to convert g and p to a BBN
bbn = convert_for_exact_inference(g, p)
# you can convert the BBN to a nx graph for visualization
nx_graph = convert_for_drawing(bbn)
```

[17]:
```
my_pos = nx.spring_layout(nx_graph, seed = 100)
nx.draw(nx_graph, pos = my_pos, with_labels=True, node_color='orange',␣
↪node_size=400, edge_color='black', linewidths=1, font_size=15)
plt.show()
```



[7]:
```
nx_graph.nodes
```

[7]:
```
NodeView((0, 1, 2, 3, 4))
```

[8]:
```
nx_graph.edges
```

[8]:
```
OutEdgeView([(0, 1), (0, 3), (1, 2), (2, 3), (3, 4)])
```

[22]:
```
from pybbn.graph.dag import Bbn
from pybbn.graph.edge import Edge, EdgeType
from pybbn.graph.jointree import EvidenceBuilder
from pybbn.graph.node import BbnNode
from pybbn.graph.variable import Variable
from pybbn.pptc.inferencecontroller import InferenceController
```

[23]:
```
join_tree = InferenceController.apply(bbn)
```

[31]:
```
# Define a function for printing marginal probabilities
def print_probs():
    for node in join_tree.get_bbn_nodes():
        potential = join_tree.get_bbn_potential(node)
        print("Node:", node)
        print("Values:")
```

```
        print(potential)
        print('----------------')

# Use the above function to print marginal probabilities
print_probs()
```

```
Node: 3|3|state0,state1
Values:
3=state0|0.30921
3=state1|0.69079
----------------
Node: 4|4|state0,state1
Values:
4=state0|0.32575
4=state1|0.67425
----------------
Node: 0|0|state0,state1
Values:
0=state0|0.05776
0=state1|0.94224
----------------
Node: 2|2|state0,state1
Values:
2=state0|0.35749
2=state1|0.64251
----------------
Node: 1|1|state0,state1
Values:
1=state0|0.60777
1=state1|0.39223
----------------
```

[32]:
```
# To add evidence of events that happened so probability distribution can be␣
 ↪recalculated
def evidence(ev, nod, cat, val):
    ev = EvidenceBuilder() \
    .with_node(join_tree.get_bbn_node_by_name(nod)) \
    .with_evidence(cat, val) \
    .build()
    join_tree.set_observation(ev)

# Use above function to add evidence


# Print marginal probabilities
print_probs()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipykernel_2571/2435206888.py in <module>
      8
      9 # Use above function to add evidence
---> 10 evidence('ev1', 'H9am', '>60', 1.0)
     11
     12 # Print marginal probabilities

/tmp/ipykernel_2571/2435206888.py in evidence(ev, nod, cat, val)
      5     .with_evidence(cat, val) \
      6     .build()
----> 7     join_tree.set_observation(ev)
      8
      9 # Use above function to add evidence

/mnt/d/Python/Projects/optimal_learning_path/.env/lib/python3.8/site-packages/
 ↪pybbn/graph/jointree.py in set_observation(self, evidence)
    356         :return: This join tree.
    357         """
--> 358         potentials = self.evidences[evidence.node.id]
    359
    360         pvalues = []

AttributeError: 'NoneType' object has no attribute 'id'
```