Bayesian Network date:24-06-2022

import pgmpy.models

import pgmpy.inference

import networkx as nx

import pylab as plt

# Create a bayesian network

model = pgmpy.models.BayesianModel([('Burglary', 'Alarm'),

('Earthquake', 'Alarm'),

('Alarm', 'JohnCalls'),

('Alarm', 'MaryCalls')])

# Define conditional probability distributions (CPD)

# Probability of burglary (True, False)

cpd\_burglary = pgmpy.factors.discrete.TabularCPD('Burglary', 2, [[0.001], [0.999]])

# Probability of earthquake (True, False)

cpd\_earthquake = pgmpy.factors.discrete.TabularCPD('Earthquake', 2, [[0.002], [0.998]])

# Probability of alarm going of (True, False) given a burglary and/or earthquake

cpd\_alarm = pgmpy.factors.discrete.TabularCPD('Alarm', 2, [[0.95, 0.94, 0.29, 0.001],

[0.05, 0.06, 0.71, 0.999]],

evidence=['Burglary', 'Earthquake'],

evidence\_card=[2, 2])

# Probability that John calls (True, False) given that the alarm has sounded

cpd\_john = pgmpy.factors.discrete.TabularCPD('JohnCalls', 2, [[0.90, 0.05],

[0.10, 0.95]],

evidence=['Alarm'],

evidence\_card=[2])

# Probability that Mary calls (True, False) given that the alarm has sounded

cpd\_mary = pgmpy.factors.discrete.TabularCPD('MaryCalls', 2, [[0.70, 0.01],

[0.30, 0.99]],

evidence=['Alarm'],

evidence\_card=[2])

# Add CPDs to the network structure

model.add\_cpds(cpd\_burglary, cpd\_earthquake, cpd\_alarm, cpd\_john, cpd\_mary)

# Check if the model is valid, throw an exception otherwise

model.check\_model()

# Print probability distributions

print('Probability distribution, P(Burglary)')

print(cpd\_burglary)

print()

print('Probability distribution, P(Earthquake)')

print(cpd\_earthquake)

print()

print('Joint probability distribution, P(Alarm | Burglary, Earthquake)')

print(cpd\_alarm)

print()

print('Joint probability distribution, P(JohnCalls | Alarm)')

print(cpd\_john)

print()

print('Joint probability distribution, P(MaryCalls | Alarm)')

print(cpd\_mary)

print()

# Plot the model

nx.draw(model, with\_labels=True)

plt.savefig('C:\\Users\\admin\\Desktop\\alarm.png')

plt.close()

# Perform variable elimination for inference

# Variable elimination (VE) is a an exact inference algorithm in bayesian networks

infer = pgmpy.inference.VariableElimination(model)

# Calculate the probability of a burglary if John and Mary calls (0: True, 1: False)

posterior\_probability = infer.query(['Burglary'], evidence={'JohnCalls': 0, 'MaryCalls': 0})

# Print posterior probability

print('Posterior probability of Burglary if JohnCalls(True) and MaryCalls(True)')

print(posterior\_probability)

print()

# Calculate the probability of alarm starting if there is a burglary and an earthquake (0: True, 1: False)

posterior\_probability = infer.query(['Alarm'], evidence={'Burglary': 0, 'Earthquake': 0})

# Print posterior probability

print('Posterior probability of Alarm sounding if Burglary(True) and Earthquake(True)')

print(posterior\_probability)

print()

output: 