solution11

November 10, 2020

Exercise Sheet 11 **Bayesian Networks for Inference**

[1]:
```
from pgmpy.models import BayesianModel
from pgmpy.factors.discrete import TabularCPD
from pgmpy.inference import VariableElimination

import warnings
warnings.filterwarnings("ignore")
```

11.2: Software

NOTE
in the probability tables, probabilities ending with "_ 0" are true and those ending with "_ 1" are considered false...

[2]:
```
# Starting with defining the network structure
wet_model = BayesianModel([('Cloudy', 'Sprinkler'),
                           ('Cloudy', 'Rain'),
                           ('Sprinkler', 'WetGrass'),
                           ('Rain', 'WetGrass')])
```

[3]:
```
# defining the parameters
cpd_cloudy = TabularCPD(variable='Cloudy',
                        variable_card=2,
                        values=[[0.5], [0.5]])

cpd_sprinkler_cloudy = TabularCPD(variable='Sprinkler',
                        variable_card=2,
                        values=[[0.1, 0.5],[0.9, 0.5]],
                        evidence=['Cloudy'],
                        evidence_card=[2])

cpd_rain_cloudy = TabularCPD(variable='Rain',
                        variable_card=2,
                        values=[[0.8, 0.2],[0.2, 0.8]],
                        evidence=['Cloudy'],
                        evidence_card=[2])

cpd_wetgrass_sprinkler_rain = TabularCPD(variable='WetGrass',
```

```
                        variable_card=2,
                        values=[[0.99, 0.9, 0.9, 0], [0.01, 0.1, 0.1, 1]],
                        evidence=['Sprinkler','Rain'],
                        evidence_card=[2,2])
```

[4]:
```
# Checking the model
wet_model.add_cpds(cpd_cloudy,
                   cpd_sprinkler_cloudy,
                   cpd_rain_cloudy,
                   cpd_wetgrass_sprinkler_rain)

print(wet_model.check_model())

wet_model.get_independencies()
```

```
True
```

[4]:
```
(Cloudy _|_ WetGrass | Sprinkler, Rain)
(Sprinkler _|_ Rain | Cloudy)
(Rain _|_ Sprinkler | Cloudy)
(WetGrass _|_ Cloudy | Sprinkler, Rain)
```

[5]:
```
# inference
wet_infer = VariableElimination(wet_model)
```

[21]:
```
# probability distributions
p_rain = wet_infer.query(variables=['WetGrass'])
print('prob. wet grass:\n',p_rain['WetGrass'])
```

```
prob. wet grass:
 +------------+-----------------+
| WetGrass   |   phi(WetGrass) |
+============+=================+
| WetGrass_0 |          0.6471 |
+------------+-----------------+
| WetGrass_1 |          0.3529 |
+------------+-----------------+
```

[22]:
```
p_sprinkler_if_wet = wet_infer.query(variables=['Sprinkler'],
                                     evidence={'WetGrass': 0})
print('prob. sprinkler if grass is wet:\n',
      p_sprinkler_if_wet['Sprinkler'])
```

```
prob. sprinkler if grass is wet:
 +-------------+------------------+
| Sprinkler   |   phi(Sprinkler) |
+=============+==================+
| Sprinkler_0 |           0.4298 |
+-------------+------------------+
```

```
| Sprinkler_1 |           0.5702 |
+-------------+------------------+
```

[23]:
```
p_sprinkler_if_wet = wet_infer.query(variables=['Rain'],
                                     evidence={'WetGrass': 0})
print('prob. rain if grass is wet:\n',
      p_sprinkler_if_wet['Rain'])
```

```
prob. rain if grass is wet:
 +--------+-------------+
| Rain   |   phi(Rain) |
+========+=============+
| Rain_0 |      0.7079 |
+--------+-------------+
| Rain_1 |      0.2921 |
+--------+-------------+
```

Exercise H11.3: Construction of a DAG

(b)

[11]:
```
# defining the model
dag_model = BayesianModel([('Burglary', 'Alarm'),
                           ('Earthquake', 'Alarm'),
                           ('Earthquake', 'Radio')])

# defining the parameters
cpd_B = TabularCPD(variable='Burglary',
                   variable_card=2,
                   values=[[0.01], [0.99]])

cpd_E = TabularCPD(variable='Earthquake',
                   variable_card=2,
                   values=[[1e-6], [1-1e-6]])

cpd_R_E = TabularCPD(variable='Radio',
                   variable_card=2,
                   values=[[1, 0],[0, 1]],
                   evidence=['Earthquake'],
                   evidence_card=[2])

cpd_A_B_E = TabularCPD(variable='Alarm',
                   variable_card=2,
                   values=[[0.98, 0.95, 0.41, 0.001], [0.02, 0.05, 0.59, 0.
 ↪999]],
                   evidence=['Burglary', 'Earthquake'],
                   evidence_card=[2,2])
```

[12]:
```
# Checking the model
dag_model.add_cpds(cpd_B,
                   cpd_E,
                   cpd_R_E,
                   cpd_A_B_E)

print(dag_model.check_model())

dag_model.get_independencies()
```

```
True
```

[12]:
```
(Burglary _|_ Radio, Earthquake)
(Burglary _|_ Earthquake | Radio)
(Burglary _|_ Radio | Earthquake)
(Burglary _|_ Radio | Alarm, Earthquake)
(Alarm _|_ Radio | Earthquake)
(Alarm _|_ Radio | Earthquake, Burglary)
(Earthquake _|_ Burglary)
(Earthquake _|_ Burglary | Radio)
(Radio _|_ Burglary)
(Radio _|_ Alarm, Burglary | Earthquake)
(Radio _|_ Burglary | Alarm, Earthquake)
(Radio _|_ Alarm | Earthquake, Burglary)
```

[15]:
```
# inference
dag_infer = VariableElimination(dag_model)
```

[24]:
```
# probability distributions
p_alarm = dag_infer.query(variables=['Alarm'])
print('prob. of alarm:\n',
      p_alarm['Alarm'])
```

```
prob. of alarm:
 +---------+--------------+
| Alarm   |   phi(Alarm) |
+=========+==============+
| Alarm_0 |       0.0105 |
+---------+--------------+
| Alarm_1 |       0.9895 |
+---------+--------------+
```

[25]:
```
p_A_if_R = dag_infer.query(variables=['Alarm'],
                           evidence={'Radio': 0})
print('prob. alarm if radio:\n',
      p_A_if_R['Alarm'])
```

```
prob. alarm if radio:
 +---------+--------------+
```

```
| Alarm   |   phi(Alarm) |
+=========+==============+
| Alarm_0 |       0.4157 |
+---------+--------------+
| Alarm_1 |       0.5843 |
+---------+--------------+
```

[26]:
```
p_B_if_A = dag_infer.query(variables=['Burglary'],
                           evidence={'Alarm': 0})
print('prob. burglary if alarm:\n',
      p_B_if_A['Burglary'])
```

```
prob. burglary if alarm:
 +------------+-----------------+
| Burglary   |   phi(Burglary) |
+============+=================+
| Burglary_0 |          0.9056 |
+------------+-----------------+
| Burglary_1 |          0.0944 |
+------------+-----------------+
```

[27]:
```
p_B_if_A_and_R = dag_infer.query(variables=['Burglary'],
                           evidence={'Alarm': 0, 'Radio': 0})
print('prob. burglary if alarm and radio:\n',
      p_B_if_A_and_R['Burglary'])
```

```
prob. burglary if alarm and radio:
 +------------+-----------------+
| Burglary   |   phi(Burglary) |
+============+=================+
| Burglary_0 |          0.0236 |
+------------+-----------------+
| Burglary_1 |          0.9764 |
+------------+-----------------+
```

11.3 (C) 'Explaining Away' stipulates that confirmation of one posterior event A reduces the need to look for the occurrence of other events, which alongside A are parents to an observed event.

[ ]:
```

```