Assignment 1: Modeling with DAGs

Andrew Lemke

In [1]:
```
import pandas as pd
from pgmpy.models import BayesianModel, BayesianNetwork
from pgmpy.estimators import BayesianEstimator
```

Loading the data

The raw data comes in the form of a csv.

In [2]:
```
data_file = 'transportation_survey.txt'
```

In [3]:
```
raw_data = pd.read_csv(
    data_file
)
```

In [4]:
```
raw_data.head(10)
```

Out[4]:

	A	S	E	O	R	T
0	adult	F	high	emp	small	train
1	young	M	high	emp	big	car
2	adult	M	uni	emp	big	other
3	old	F	uni	emp	big	car
4	young	F	uni	emp	big	car
5	young	F	uni	emp	big	car
6	adult	F	high	emp	small	other
7	adult	F	high	emp	big	other
8	adult	M	high	emp	big	car
9	adult	M	high	emp	big	car

The default options for loading the csv worked well.

Building the DAG

The variables in the model are:

- **A**ge, young for those under 30, old for those over 60, and adult for those between young and old
- **S**ex, the sex of the individual, M or F

- **E**ducation, whether the individual completed high echool only (high) or has a university degree (uni)
- **O**ccupation, (self) employed or (emp)loyee
- **R**esidence, if the person lives in a (small) or (big) city
- **T**ravel, the preferred means of travel of the individual

DAG

In [5]:
```
# the relationships are put in as pairs of source node to destination node
model = BayesianNetwork(
    [('A', 'E'), ('S', 'E'), ('E', 'O'), ('E', 'R'), ('O', 'T'), ('R', 'T')]
)
```

Learning the conditional probabilities

Since we do not have any priors to distribute on, we will use the `K2` prior type, which is a dirichlet distrobution with every pseudo count set to 1. If we had some other data to indicate some prior to distribute on (for example a similar dataset from another similar country) we could use it as a prior.

In [6]:
```
estimator = BayesianEstimator(model, raw_data)
```

In [7]:
```
# T is the node we desire the conditional probability distrobution on
cpd_C = estimator.estimate_cpd('T', prior_type="K2")
```

In [8]:
```
print(cpd_C)
```

```
+----------+--------------------+-----+--------------------+----------+
| O        | O(emp)             | ... | O(self)            | O(self)  |
+----------+--------------------+-----+--------------------+----------+
| R        | R(big)             | ... | R(big)             | R(small) |
+----------+--------------------+-----+--------------------+----------+
| T(car)   | 0.7007299270072993 | ... | 0.4166666666666667 | 0.5      |
+----------+--------------------+-----+--------------------+----------+
| T(other) | 0.1362530413625304 | ... | 0.3333333333333333 | 0.25     |
+----------+--------------------+-----+--------------------+----------+
| T(train) | 0.1630170316301703 | ... | 0.25               | 0.25     |
+----------+--------------------+-----+--------------------+----------+
```

In [9]:
```
for x in estimator.get_parameters('K2'):
    print(x)
```

```
+----------+----------+
| A(adult) | 0.387674 |
+----------+----------+
| A(old)   | 0.139165 |
+----------+----------+
| A(young) | 0.473161 |
+----------+----------+
+---------+--------------------+-----+---------------------+
| A       | A(adult)           | ... | A(young)            |
+---------+--------------------+-----+---------------------+
| S       | S(F)               | ... | S(M)                |
+---------+--------------------+-----+---------------------+
| E(high) | 0.5185185185185185 | ... | 0.7642276422764228  |
+---------+--------------------+-----+---------------------+
| E(uni)  | 0.48148148148148145 | ... | 0.23577235772357724 |
+---------+--------------------+-----+---------------------+
+------+----------+
| S(F) | 0.521912 |
+------+----------+
| S(M) | 0.478088 |
+------+----------+
+---------+---------------------+--------------------+
| E       | E(high)             | E(uni)             |
+---------+---------------------+--------------------+
| O(emp)  | 0.9794520547945206  | 0.9716981132075472 |
+---------+---------------------+--------------------+
| O(self) | 0.02054794520547945 | 0.02830188679245283 |
+---------+---------------------+--------------------+
+----------+---------------------+--------------------+
| E        | E(high)             | E(uni)             |
+----------+---------------------+--------------------+
| R(big)   | 0.7568493150684932  | 0.9339622641509434 |
+----------+---------------------+--------------------+
| R(small) | 0.24315068493150685 | 0.0660377358490566 |
+----------+---------------------+--------------------+
+----------+--------------------+-----+--------------------+----------+
| O        | O(emp)             | ... | O(self)            | O(self)  |
+----------+--------------------+-----+--------------------+----------+
| R        | R(big)             | ... | R(big)             | R(small) |
+----------+--------------------+-----+--------------------+----------+
| T(car)   | 0.7007299270072993 | ... | 0.4166666666666667 | 0.5      |
+----------+--------------------+-----+--------------------+----------+
| T(other) | 0.1362530413625304 | ... | 0.3333333333333333 | 0.25     |
+----------+--------------------+-----+--------------------+----------+
| T(train) | 0.1630170316301703 | ... | 0.25               | 0.25     |
+----------+--------------------+-----+--------------------+----------+
```

The below cell is the last table in another form. The first row of the table above is the first of the three array sets below. The first line is `P(T=car | O=emp, R=big)`, `P(T=car | O=emp, R=Small)`, the next line is `P(T=car | O=self, R=big)`, `P(T=car | O=self, R=Small)`.

The next two blocks are similar for other and train.

In [10]:
```
cpd_C.values
```

Out[10]:
```
array([[[0.70072993, 0.51764706],
        [0.41666667, 0.5       ]],

       [[0.13625304, 0.09411765],
        [0.33333333, 0.25      ]],

       [[0.16301703, 0.38823529],
        [0.25      , 0.25      ]]])
```

In [11]:
```
cpd_C.variables
```

Out[11]:
```
['T', 'O', 'R']
```

Assessment Questions

1. Which factorization is factorized along the DAG (Markov factorization).

We look to the graph to find the answer.

`p(A)(S) p(E|A, S) p(O|E)p(R|E) p(T|O, R)`

2. Which is true about Node E (education):

Again, the graph shows us the answer. The parents of E are A and S. The children (the outward edges) are E and R.

3. Suppose we modify the network by removing the edge from E to O. Which local distributions (factors in the factorization) change?

With parameter modularity, changes to one node's distobution do not change other nodes distrobutions. Here, the distrobutions are affected by a structural change to the graph. By removing edge EO, O becomes a root node--it's probability is no longer conditioned on E. This does not change the distrobution on E, as E is not conditioned on O. Even though R is conditioned on O, the change to O does not alter `p(T|O, R)` due to the parameter modularity.

asdf

> A categorical variable as three outcomes with probabilities p1, p2, and p3. You place a Dirichlet prior on these probabilities with concentration parameters 1, 1, and 1. In data with 20 observations you observe 10 instances of class 1, 2 instances of class 2, and 8 instances of class 3. What are the concentration parameters of the posterior.

With a dirichlet prior of 1, 1, 1, we get a uniform prior. When we get new observations, we incorperate them.

priors:

```
p1 = 1
p2 = 1
p3 = 1
```

With the new observations distrobuted on this prior,

```
p1 = 1 + 10
p2 = 1 + 2
p3 = 1 + 8
```

Resources

- dirichlet
  - https://www.youtube.com/watch?v=nfBNOWv1pgE
  - https://www.youtube.com/watch?v=gWgsKyEjclw
- pgmpy
  - https://pgmpy.org/models/bayesiannetwork.html
  - https://pgmpy.org/param_estimator/bayesian_est.html
  - https://pgmpy.org/factors/discrete.html#module-pgmpy.factors.discrete.JointProbabilityDistribution
  - https://pgmpy.org/examples/Learning%20Parameters%20in%20Discrete%20Bayesian%20N