Tyler Starkus

5/1/2020

Artificial Intelligence

Spring 2020

Machine Problem 4

```
In [1]: import pyAgrum as gum
```

Part A

```
In [2]: bn=gum.BayesNet('FraudDetection')
        print(bn)

BN{nodes: 0, arcs: 0, domainSize: 1, dim: 0}
```

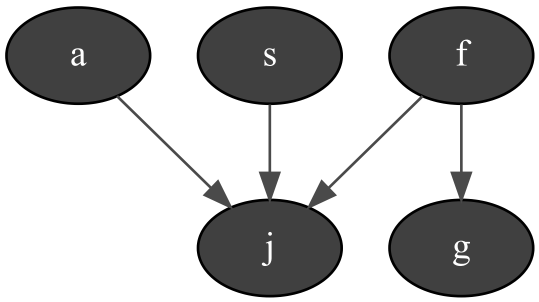
```
In [3]: #a=bn.add(gum.LabelizedVariable('a','age',3))
        #s, g, j, f = [ bn.add(name, 2) for name in "sgjf" ]
        #print (a,s,g,j,f)
        #print (bn)
```

```
In [4]: #for link in [('f','g'),('f','j'),('a','j'),('s','j')]:
        #    bn.addArc(*link)
        #print(bn)
```

```
In [5]: # For whatever reason the code blocks above don't work with inferencing, but thi
        bn=gum.fastBN("f->j;f->g;a[3]->j<-s;")
```

```
In [6]: import pyAgrum.lib.notebook as gnb
        bn
```

Out[6]:



```
In [7]: bn.cpt('j').var_names

Out[7]: ['s', 'a', 'f', 'j']
```

In [8]:
```
bn.cpt('f').fillWith([.99,.01])
```

Out[8]:

f	
0	1
0.9900	0.0100

In [9]:
```
bn.cpt('j')[{'a': 0, 's': 0, 'f': 0}] = [0.9999, 0.0001]
bn.cpt('j')[{'a': 0, 's': 0, 'f': 1}] = [0.95, 0.05]
bn.cpt('j')[{'a': 0, 's': 1, 'f': 0}] = [0.9995, 0.0005]
bn.cpt('j')[{'a': 0, 's': 1, 'f': 1}] = [0.95, 0.05]

bn.cpt('j')[{'a': 1, 's': 0, 'f': 0}] = [0.9996, 0.0004]
bn.cpt('j')[{'a': 1, 's': 0, 'f': 1}] = [0.95, 0.05]
bn.cpt('j')[{'a': 1, 's': 1, 'f': 0}] = [0.998, 0.002]
bn.cpt('j')[{'a': 1, 's': 1, 'f': 1}] = [0.95, 0.05]

bn.cpt('j')[{'a': 2, 's': 0, 'f': 0}] = [0.9998, 0.0002]
bn.cpt('j')[{'a': 2, 's': 0, 'f': 1}] = [0.95, 0.05]
bn.cpt('j')[{'a': 2, 's': 1, 'f': 0}] = [0.999, 0.001]
bn.cpt('j')[{'a': 2, 's': 1, 'f': 1}] = [0.95, 0.05]

bn.cpt('j')
```

Out[9]:

			j	
s	a	f	0	1
0	0	0	0.9999	0.0001
		1	0.9500	0.0500
	1	0	0.9996	0.0004
		1	0.9500	0.0500
	2	0	0.9998	0.0002
		1	0.9500	0.0500
1	0	0	0.9995	0.0005
		1	0.9500	0.0500
	1	0	0.9980	0.0020
		1	0.9500	0.0500
	2	0	0.9990	0.0010
		1	0.9500	0.0500

In [10]:
```
bn.cpt('g')[{'f': 0}] = [0.99, 0.01]
bn.cpt('g')[{'f': 1}] = [0.80, 0.20]
bn.cpt('g')
```

Out[10]:

	g	
f	**0**	**1**
0	0.9900	0.0100
1	0.8000	0.2000

Part B

In [11]:
```
print(gum.availableBNExts())
```

```
bif|dsl|net|bifxml|o3prm|uai
```

In [12]:
```
ie=gum.LazyPropagation(bn)
```

In [13]:
```
# Maximizing
ie.setEvidence({'a':0, 's':0, 'g':1, 'j': 1}) # hard evidence
ie.makeInference()
ie.posterior('f')
```

Out[13]:

f	
0	**1**
0.0098	0.9902

In [14]:
```
# Minimizing
ie.setEvidence({'a':2, 's':1, 'g':0, 'j': 0}) # hard evidence
ie.makeInference()
ie.posterior('f')
```

Out[14]:

f	
0	**1**
0.9923	0.0077

Part C

In [15]:
```
# Given gas and jewelry
ie.setEvidence({'g':1, 'j': 1}) # hard evidence
ie.makeInference()
ie.posterior('f')
```

Out[15]:

f	
0	**1**
0.0710	0.9290

In [16]:
```
# Given gas and not jewelry
ie.setEvidence({'g':1, 'j': 0}) # hard evidence
ie.makeInference()
ie.posterior('f')
```

Out[16]:

f	
0	1
0.8389	0.1611

In [17]:
```
# Given not gas but jewelry
ie.setEvidence({'g':0, 'j': 1}) # hard evidence
ie.makeInference()
ie.posterior('f')
```

Out[17]:

f	
0	1
0.6542	0.3458

In [19]:
```
# Given neither gas nor jewelry
ie.setEvidence({'g':0, 'j': 0}) # hard evidence
ie.makeInference()
ie.posterior('f')
```

Out[19]:

f	
0	1
0.9923	0.0077