Belief Networks in Julia with `BayesNets.jl`

In this notebook we will use the `BayesNets` julia package (https://github.com/sisl/BayesNets.jl) to build belief networks.

Example 1: Inspector Clouseau

Here we re-do the Inspector Clouseau example from Barber, but this time by constructing a belief network.

> **Barber Example 1.3 (Inspector Clouseau)**
>
> Inspector Clouseau arrives at the scene of a crime. The victim lies dead in the room and the inspector quickly finds the murder weapon, a Knife (`K`). The Butler (`B`) and Maid (`M`) are his main suspects. The inspector has a prior belief of 0.8 that the Butler is the murderer, and a prior belief of 0.2 that the Maid is the murderer. These probabilities are independent in the sense that $p(B, M) = p(B)p(M)$. (It is possible that both the Butler and the Maid murdered the victim or neither). The inspector's *prior* criminal knowledge can be formulated mathematically as follows.

(Here we use short-hand notation, e.g. $p(B)$ means the (*a priori*) probability that the butler is the murder, $p(K|B, \neg M$ means the probability that the knife was used given that the butler was the murder and the maid was not.)

> $$p(B) = 0.6$$
> $$p(M) = 0.2$$
> $$p(M, B) = p(M)p(B)$$
>
> $$p(K|\neg B, \neg M) = 0.3$$
> $$p(K|\neg B, M) = 0.2$$
> $$p(K|B, \neg M) = 0.6$$
> $$p(K|B, M) = 0.1$$
>
> Assuming the knife is the murder weapon, what is the probability that the Butler is the murderer?

Build the Network

A belief network is specified by variables and their conditional probabilities. To build the model, you describe the structure of network by defining nodes and links in a directed graph, and then assign the conditional probabilities (or priors) for each variable.

We have to be careful, because the convention for specifying the conditional probability (`DiscreteCPD`) is different from that in `pgmpy` and from (1.2.6) in Barber. The *first* variable changes the fastest. From the docs:

In [1]:

```
1 using BayesNets
```

In [2]:

```
1 ?DiscreteCPD
```

```
search: DiscreteCPD Discrete DiscreteUniform DiscreteBayesNet
```

Out[2]:

A categorical distribution, P(x|parents(x)) where all parents are discrete integers 1:N.

The ordering of `distributions` array follows the convention in Decision Making Under Uncertainty. Suppose a variable has three discrete parents. The first parental instantiation assigns all parents to their first bin. The second will assign the first parent (as defined in `parents`) to its second bin and the other parents to their first bin. The sequence continues until all parents are instantiated to their last bins.

This is equivalent to:

X,Y,Z 1,1,1 2,1,1 1,2,1 2,2,1 1,1,2 ...

In [3]:

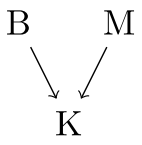
```
using BayesNets

model = DiscreteBayesNet() # uses variable values 1:N

# variable convention:  [False, True]
push!(model, DiscreteCPD(:B, [0.4, 0.6])) # :foo means symbol foo in Julia
push!(model, DiscreteCPD(:M, [0.8, 0.2]))

push!(model, DiscreteCPD(:K, [:B, :M], [2, 2],
    [Categorical([0.7, 0.3]),
     Categorical([0.4, 0.6]), # B=T, M=F
     Categorical([0.8, 0.2]), # B=F, M=T
     Categorical([0.9, 0.1]),
    ]))
```

Out[3]:



TODO: We can also make a more user-friendly model by using category names. (tried, but only partial success)

In [4]:

```
bn = BayesNet()
```

Out[4]:

Empty Graph

In [5]:

```
pB = NamedCategorical([:murderer, :not_murderer], [0.6, 0.4])
```

Out[5]:

```
NamedCategorical with entries:
          0.4000:  not_murderer
          0.6000:  murderer
```

In [6]:

```
pB = NamedCategorical(["murderer", "not murderer"], [0.6, 0.4])
```

Out[6]:

```
NamedCategorical with entries:
          0.4000:  not murderer
          0.6000:  murderer
```

In [7]:

```
1 CategoricalCPD(:B, pB)
```

Out[7]:

```
CategoricalCPD{NamedCategorical{String}}(:B, Symbol[], Int64[], NamedC
ategorical{String}[NamedCategorical with entries:
          0.4000:  not murderer
          0.6000:  murderer
])
```

In [8]:

```
1 pM = NamedCategorical(["murderer", "not murderer"], [0.2, 0.8])
```

Out[8]:

```
NamedCategorical with entries:
          0.8000:  not murderer
          0.2000:  murderer
```

In [9]:

```
1 push!(bn, CategoricalCPD(:B, pB), CategoricalCPD(:M, pM))
```

Out[9]:

In [10]:

```
bn = BayesNet()
push!(bn, DiscreteCPD(:K, [:B, :M], [2, 2], [
    NamedCategorical(["knife used", "knife not used"], [0.3, 0.7]),  # note ord
    NamedCategorical(["knife used", "knife not used"], [0.6, 0.4]),  # p(K | B=
    NamedCategorical(["knife used", "knife not used"], [0.2, 0.8]),  # p(K | B=
    NamedCategorical(["knife used", "knife not used"], [0.1, 0.9])
    ]))
```

```
MethodError: Cannot `convert` an object of type NamedCategorical{Strin
g} to an object of type Categorical{Float64}
Closest candidates are:
  convert(::Type{Categorical{T<:Real}}, !Matched::Array{S<:Real,1}) wh
ere {T<:Real, S<:Real} at /Users/mike/.julia/packages/Distributions/WH
jOk/src/univariate/discrete/categorical.jl:45
  convert(::Type{Categorical{T<:Real}}, !Matched::Categorical{S<:Real}
) where {T<:Real, S<:Real} at /Users/mike/.julia/packages/Distribution
s/WHjOk/src/univariate/discrete/categorical.jl:46
  convert(::Type{S}, !Matched::T<:(Union{CategoricalString{R}, Categor
icalValue{T,R} where T} where R)) where {S, T<:(Union{CategoricalStrin
g{R}, CategoricalValue{T,R} where T} where R)} at /Users/mike/.julia/p
ackages/CategoricalArrays/ucKV2/src/value.jl:91
  ...

Stacktrace:
 [1] setindex!(::Array{Categorical{Float64},1}, ::NamedCategorical{Str
ing}, ::Int64) at ./array.jl:767
 [2] copyto! at ./abstractarray.jl:753 [inlined]
 [3] copyto! at ./abstractarray.jl:745 [inlined]
 [4] Type at ./array.jl:482 [inlined]
 [5] convert at ./array.jl:474 [inlined]
 [6] CategoricalCPD{Categorical{Float64}}(::Symbol, ::Array{Symbol,1},
::Array{Int64,1}, ::Array{NamedCategorical{String},1}) at /Users/mike/
.julia/packages/BayesNets/B5P8k/src/CPDs/categorical_cpd.jl:22
 [7] top-level scope at In[10]:2
```

Inference in Belief Networks

From Barber, we can solve for a query pdf such as $p(B|K)$ using variable elimination as follows:

$$\begin{aligned} p(B|K) &= \sum_m p(B, m|K) \\ &= \sum_m \frac{p(B, m, K)}{p(K)} \\ &= \frac{p(B) \sum_m p(K|B, m)p(m)}{\sum_b p(b) \sum_m p(K|b, m)p(m)} \end{aligned}$$

Note how the summation variables are lowercase versions of the random variables to keep them distinct. The capital variables are set from $p(B|K)$, but the summation variables are b and m, which sum over the domains of B and M (which here are the same). Filling in the values above, we get

$$\begin{aligned} &= \frac{0.6 \times (0.2 \times 0.1 + 0.8 \times 0.6)}{0.6 \times (0.2 \times 0.1 + 0.8 \times 0.6) + 0.4 \times (0.2 \times 0.2 + 0.8 \times 0.3)} \\ &\approx 0.73 \end{aligned}$$

To solve for $p(B|K = \text{true})$:

In [11]:

```
1 infer(model, :B, evidence=Assignment(:K=>2)) # K=True
```

Out[11]:

2 rows × 2 columns

	B	potential
	Int64	Float64
1	1	0.271845
2	2	0.728155

We can easily compute the probability for the other suspect $P(M|K)$

In [12]:

```
1 infer(model, :M, evidence=Assignment(:K=>2)) # K=True
```

Out[12]:

2 rows × 2 columns

	M	potential
	Int64	Float64
1	1	0.932039
2	2	0.0679612

And also add evidence. For example if we know that the maid could not have been present, $P(B|K = \text{true}, M = \text{false})$

In [13]:

```
1 infer(model, :B, evidence=Assignment(:K=>2, :M=>1)) # K=True, M=False
```

Out[13]:

2 rows × 2 columns

	B	potential
	Int64	Float64
1	1	0.25
2	2	0.75

This is the same result as in `pgmpy` .

The probability that neither of the suspects is the murderer can be computed as

In [14]:

```
infer(model, [:B, :M], evidence=Assignment(:K=>2)) # K=True
```

Out[14]:

4 rows × 3 columns

	B	M	potential
	Int64	Int64	Float64
1	1	1	0.23301
2	2	1	0.699029
3	1	2	0.038835
4	2	2	0.0291262

Explaining away

Noisy fuel gauge

A simple example from Bishop (fig 8.21).

Explaining away is the concept that our beliefs about other potential causes can be "explained away" when we learn new information.

Consider a model of a noisy electric fuel gauge in an old and unreliable car. The state of the guage G indicates the fuel level F. The gauge is electric, so it depends on the fuel level F and the state of the battery B. For simplicity, assume the states are binary.

First we define the network.

$$p(B = 1) = 0.9$$
$$p(F = 1) = 0.9$$

In [16]:

```
using BayesNets
fg = DiscreteBayesNet()
```

Out[16]:

Empty Graph

In [17]:

```
pB = DiscreteCPD(:B, [0.1, 0.9]) # p(B = 0), p(B = 1)
pF = DiscreteCPD(:F, [0.1, 0.9]) # p(F)
push!(fg, pB, pF)
```

Out[17]:

F B

In [18]:

```
pB
```

Out[18]:

2 rows × 2 columns

	B	potential
	Int64	Float64
1	1	0.1
2	2	0.9

In [19]:

```
fg
```

Out[19]:

F B

Now we define the conditional probabilities:

$$p(G = 1|B = 0, F = 0) = 0.1$$
$$p(G = 1|B = 1, F = 0) = 0.2$$
$$p(G = 1|B = 0, F = 1) = 0.2$$
$$p(G = 1|B = 1, F = 1) = 0.8$$

In [20]:

```
pGgBF = DiscreteCPD(:G, [:B, :F], [2, 2],
    [Categorical([0.9, 0.1]), # p(G | B=0, F=0)
     Categorical([0.8, 0.2]), # p(G | B=1, F=0)
     Categorical([0.8, 0.2]), # p(G | B=0, F=1)
     Categorical([0.2, 0.8])  # P(G | B=1, F=1)
    ])

push!(fg, pGgBF)
```

Out[20]:

How does our belief about the state of the tank change when we observe that the gauge reads empty?

The prior probability of the tank being empty is $p(F = 0) = 0.1$. When we observe the state of the gauge, our updated belief is given by

$$p(F = 0|G = 0)$$

One method using conditional probability:

$$\begin{aligned} p(F|G) &= \frac{p(F, G)}{p(G)} \\ &= \frac{\sum_b p(b, F, G)}{\sum_{b,f} p(b, f, G)} \\ &= \frac{\sum_b p(b)p(F)p(G|b, F)}{\sum_{b,f} p(b)p(f)p(G|b, f)} \end{aligned}$$

We could also use Bayes' rule:

$$p(F = 0|G = 0) = \frac{p(G = 0|F = 0)p(F = 0)}{p(G = 0)}$$

Converting the tables above for reference:

$$
\begin{aligned}
p(B = 0) &= 0.1 \\
p(F = 0) &= 0.1 \\
p(G = 0|B = 0, F = 0) &= 0.9 \\
p(G = 0|B = 1, F = 0) &= 0.8 \\
p(G = 0|B = 0, F = 1) &= 0.8 \\
p(G = 0|B = 1, F = 1) &= 0.2
\end{aligned}
$$

$$
\begin{aligned}
p(G = 0) &= \sum_b \sum_f p(G = 0|b, f)p(b)p(f) \\
&= 0.9 \times 0.1 \times 0.1 = 0.009 + && \text{G=0, b=0, f=0} \\
&= 0.8 \times 0.1 \times 0.9 = 0.072 + && \text{G=0, b=0, f=1} \\
&= 0.8 \times 0.9 \times 0.1 = 0.072 + && \text{G=0, b=1, f=0} \\
&= 0.2 \times 0.9 \times 0.9 = 0.162 && \text{G=0, b=1, f=1} \\
&= 0.315
\end{aligned}
$$

$$
\begin{aligned}
p(G = 0|F = 0) &= \sum_b p(G = 0|b, F)p(b) \\
&= 0.9 \times 0.1 = 0.09 + && \text{G=0, b=0, f=0} \\
&= 0.8 \times 0.9 = 0.72 + && \text{G=0, b=1, f=0} \\
&= 0.81
\end{aligned}
$$

Plugging these numbers into the Bayes' rule formula we have:

$$
\begin{aligned}
p(F = 0|G = 0) &= \frac{p(G = 0|F = 0)p(F = 0)}{p(G = 0)} \\
&= \frac{0.81 \times 0.1}{0.315} \\
&= 0.257143
\end{aligned}
$$

And somewhat less tediously, we can get the same answer:

In [21]:

```
1 infer(fg, [:F], evidence=Assignment(:G=>1))
```

Out[21]:

2 rows × 2 columns

	F	potential
	Int64	Float64
1	1	0.257143
2	2	0.742857

Now, back to explaining "explaining away"... Our prior belief that the fuel tank was empty was $p(F = 0) = 0.1$. We just calculated that when we observe that the gauge reads empty, our belief changes to $p(F = 0|G = 0) \approx 0.25$. So seeing the gauge *increases* our belief that the fuel tank is empty.

We know, however, that the batter in the car is unreliable, so if we check the battery and see that it is dead, we see that

$$p(F = 0|G = 0, B = 0) = 0.111$$

In [22]:

```
1 infer(fg, [:F], evidence=Assignment(:G=>1, :B=>1))
```

Out[22]:

2 rows × 2 columns

	F	potential
	Int64	Float64
1	1	0.111111
2	2	0.888889

Summarizing our change in beliefs:

belief	probability	explanation
p(F = 0)	0.1	prior probability of empty fuel tank
p(F = 0 | G = 0)	0.257	observing the gauge reads empty
p(F = 0 | G = 0, B = 0)	0.111	finding a dead battery with an empty gauge reading

Seeing the dead battery *explains away* our belief that the empty fuel gauge reading was caused by an empty fuel tank.

If we had just observed the dead battery, then our beliefs are the same as our prior:

In [23]:

```
1 infer(fg, [:F], evidence=Assignment(:B=>1))
```

Out[23]:

2 rows × 2 columns

	F	potential
	Int64	Float64
1	1	0.1
2	2	0.9

Wet grass

Here is another example of explaining away: the wet grass example in Barber 3.1.1. Variables are `0 or 1` meaning `false` or `true`.

(Fix: Why is this right justified? The Markdown seems to be ignored.)

variable value	meaning
R = 1	It has been raining.
S = 1	Tracey forgot to turn off sprinkler.
J = 1	Jack’s grass is wet.
T = 1	Tracey’s Grass is wet

In [24]:

```
using BayesNets
wg = DiscreteBayesNet()
```

Out[24]:

Empty Graph

In [25]:

```
pR = DiscreteCPD(:R, [0.8, 0.2]) # p(R = false), p(R = true)
pS = DiscreteCPD(:S, [0.9, 0.1]) # p(S)
push!(wg, pR, pS)
```

Out[25]:

S R

In [26]:

```
pR
```

Out[26]:

2 rows × 2 columns

	R	potential
	Int64	Float64
1	1	0.8
2	2	0.2

In [27]:

```
# params: (target, parents, list parent_ncategories, list of distributions
pJgR = DiscreteCPD(:J, :R, [2],
    [Categorical([0.8, 0.2]), # p(J | R=F) (wet due to something else)
     Categorical([0.0, 1.0])  # p(J | R=T)
    ])
push!(wg, pJgR)
```

Out[27]:

R S

↓

J

In [ ]:

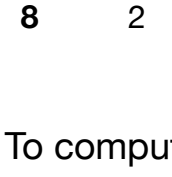
```
pJgR # Note: 1=false, 2=true
```

In [28]:

```
pTgRS = DiscreteCPD(:T, [:R, :S], [2, 2],
    [Categorical([1.0, 0.0]), # p(T | R=F, S=F)
     Categorical([0.0, 1.0]), # p(T | R=T, S=F)
     Categorical([0.1, 0.9]), # p(T | R=F, S=T)
     Categorical([0.0, 1.0])  # P(T | R=T, S=T)
    ])

push!(wg, pTgRS)
```

Out[28]:



In [29]:

```
pTgRS
```

Out[29]:

8 rows × 4 columns

	T	R	S	potential
	Int64	Int64	Int64	Float64
1	1	1	1	1.0
2	2	1	1	0.0
3	1	2	1	0.0
4	2	2	1	1.0
5	1	1	2	0.1
6	2	1	2	0.9
7	1	2	2	0.0
8	2	2	2	1.0

To compute $p(S|T = \text{true})$:

In [30]:

```
1 infer(wg, [:S], evidence=Assignment(:T=>2))
```

Out[30]:

2 rows × 2 columns

	S	potential
	Int64	Float64
1	1	0.661765
2	2	0.338235

which matches Barber (3.1.11). Now look what happens if Tracey also observes that her neighbor Jack's grass is also wet (Barber 3.1.15):

In [31]:

```
1 infer(wg, [:S], evidence=Assignment(:T=>2, :J=>2))
```

Out[31]:

2 rows × 2 columns

	S	potential
	Int64	Float64
1	1	0.839552
2	2	0.160448

The probability goes *down*. Another way of saying this is that the most likely explanation for why Jack's grass is wet is rain, which "*explains away*" the cause of the sprinkler.

If Tracey observes Jack's grass is not wet, then the only explanation (with this model) is that she forgot to turn off her sprinkler.

In [32]:

```
1  infer(wg, [:S], evidence=Assignment(:T=>2, :J=>1))
```

Out[32]:

2 rows × 2 columns

	S	potential
	Int64	Float64
1	1	0.0
2	2	1.0

In [ ]:

```
1
```