Lecture 3

Cezar Ionescu

Bayesian networks

Joint probability distributions

The mathematical model of probability consists of a triple `(Ω, Event, p)` as in the previous lecture. Very frequently, however, the set of possible outcomes `Ω` has the structure of a *cartesian product*, i.e., a tuple of possible results. For instance, the diagnostic problem we discussed last time could also have been modelled by taking `Omega = (H, R)`, i.e., the set of tuples `{(healthy, ⊖), (healthy, ⊕), (ill, ⊖), (ill, ⊕)`. The event "healthy person" would then be described by `Healthy = {(healthy, ⊖), (healthy, ⊕)}`, etc.

Exercise: Define the other events of interest: "ill", "tested positive", "tested negative".

We can now construct new probability spaces on each component of `Ω`. For example:

```
pₕ : ℙ(H) → [0, 1]
pₕ (healthy) = p({(healthy, ⊖), (healthy, ⊕)})
             = p (healthy, ⊖) + p (healthy, ⊕)
```

Remark: in general, if `Ω = (Ω₁, Ω₂, ..., Ωₙ)` and `ωᵢ ∈ Ωᵢ`, we will use `p(ω_i)` to denote

```
p(ωᵢ) = p({(x₁, x₂, ..., xₙ) | x₁ ∈ Ω₁, ..., xₙ ∈ Ωₙ, xᵢ = ωᵢ})
```

Remark: recall our convention that for `x ∈ Ω` `p(x)` is an abbreviation for `p({x})`. Above, we have taken this convention one step further, by using `p(x, y)` to denote `p({(x, y)})`, and `p(x)` to denote `p({(x, y) | y ∈ Y})`. We shall continue using this form of abbreviation throughout, as is standard usage.

The "complete" probability measure `p` is called the **joint probability distribution**, the probability measures over the individual components (or over tuples of individual components) are called **marginal probability distributions**.

Remark: it is trivial to derive the marginal probability distributions, given the joint probability distribution. However, even if we have all the marginal distributions, we cannot derive the joint distribution from them.

By definition, the joint distribution is sufficient to answer any query about the probabilities of events in `Ω`. The problem is that it requires exponential storage space with respect to the

number of components. For example, if Ω consists of a tuple of n boolean components, then the joint probability distribution can be represented as a table with 2^n rows.

Let $\Omega = (\Omega_1, \Omega_2, \ldots, \Omega_n)$ and $(\omega_1, \omega_2, \ldots, \omega_n) \in \Omega$. Then

```
p(ω₁, ω₂, ..., ωₙ)  =  p(ω₁ ∩  ω₂ ∩  ... ∩ ωₙ)                  -- recall convention above!
                    =  p(ω₁ | ω₂ ∩  ... ∩ ωₙ)*p(ω₂ ∩  ... ∩ ωₙ)  -- Bayes' theorem
```

It is easy to see that, therefore:

```
p(ω₁, ω₂, ..., ωₙ)  =  p(ω₁ |  ω₂ ∩  ... ∩ ωₙ) *
                       p(ω₂ | ω₃ ∩ ... ∩ ωₙ) * ...
                       p(ωₙ)
```

Independence, conditional independence

Definition: Two events `X, Y ⊆ Ω` are called **independent** if

```
p(X ∩ Y) = p(X) * p(Y)
```

Example: Consider a fair standard die, and the events `even = {2, 4, 6}`, `big = {5, 6}`. Then `even` and `big` are independent.

Exercise: If `X, Y` are independent, then so are `X, ¬Y`.

Example: The events `big` and `divBy3 = {3, 6}` are **not** independent.

Definition: Consider events `X, Y, Z ⊆ Ω` such that `p(Z) ≠ 0`. Events `X, Y` are called **conditionally independent given** `Z` if

```
p(X ∩ Y | Z) = P(X | Z) * p(Y | Z)
```

Example:

- The events `even, big` are conditionally dependent given `divBy3`. Intuition: knowing the result is `divBy3` means that knowledge of it being `big` informs knowledge of it being `even`.
- The events `divBy3, big` are conditionally independent given `{2, 3, 5, 6}`. Intuition: knowing that the result is in `{2, 3, 5, 6}` means that knowledge of it being `big` doesn't tell me anything more about it being `divBy3`

Bayesian networks

A Bayesian network is a graphical device for recording conditional independence information for a finite $\Omega = (\Omega_1, \Omega_2, \ldots, \Omega_n)$, in order to make the computation of entries of the joint probability distribution more efficient. The network is a *directed acyclic graph (DAG)*, with one node for each component Ω_i. For example:

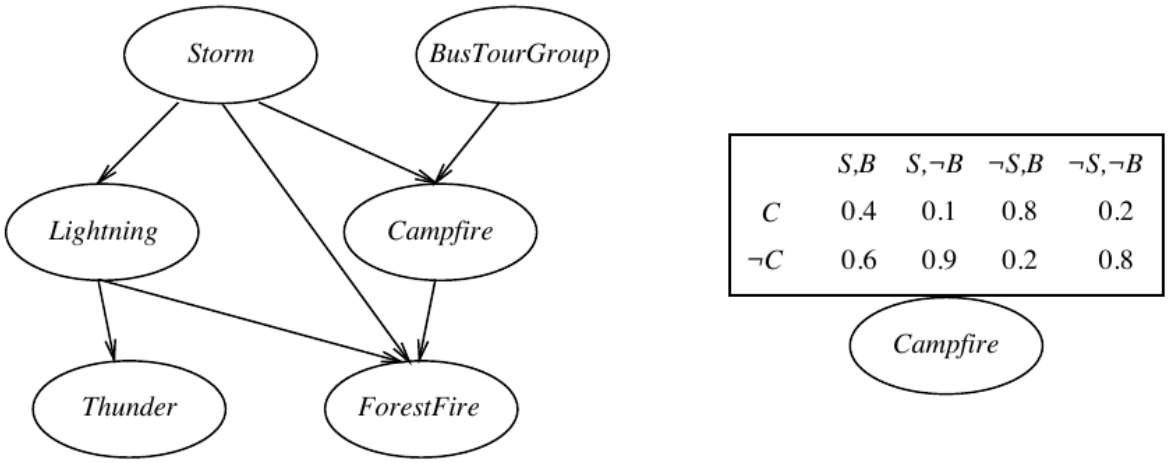

Figure 1: Mitchell, page 186

The network is assumed to be constructed in such a way that the probability of an `ω_i` is conditionally independent of any non-descendants given its parents.

**Example*: In the network above, we have

```
p(thunder | lightning ∩ bustourgroup) = p(thunder | lightning)
```

but in general

```
p(lightning | thunder ∩ storm) ≠ p(lightning | storm)
```

In this way, the Bayesian network allows us to compute any entry in the joint distribution table. For example:

```
  p(thunder, forestfire, campfire, lightning, storm, bustourgroup)
=
  p(thunder | forestfire ∩ campfire ∩ lightning ∩ storm ∩ bustourgroup) *
  p(forestfire ∩ campfire ∩ lightning ∩ storm ∩ bustourgroup)
=
  p(thunder | lightning) *
  p(forestfire ∩ campfire ∩ lightning ∩ storm ∩ bustourgroup)
=
  p(thunder | lightning) *
  p(forestfire | campfire ∩ lightning ∩ storm ∩ bustourgroup) *
  p(campfire ∩ lightning ∩ storm ∩ bustourgroup)
=
  p(thunder | lightning) *
  p(forestfire | campfire ∩ lightning) *
  p(campfire ∩ lightning ∩ storm ∩ bustourgroup)
=
  p(thunder | lightning) *
```

```
  p(forestfire | campfire ∩ lightning) *
  p(campfire | lightning ∩ storm ∩ bustourgroup) *
  p(lightning ∩ storm ∩ bustourgroup)
=
  p(thunder | lightning) *
  p(forestfire | campfire ∩ lightning) *
  p(campfire | storm ∩ bustourgroup) *
  p(lightning | storm) *
  p(storm) * p(bustourgroup)
```

Assuming all the Ω_i are boolean, we need two tables 1x2 tables to store the probability values for `Storm`, `BusTourGroup`, two 2x2 tables to store the conditional table for `Lightning | Storm` and `Thunder | Lightning`, and two 4x2 tables to store the conditional table for `ForestFire | Campfire ∩ Lightning` and `Campfire | Storm ∩ BusTourGroup`, altogether 28 values. If we stored the full joint distribution table, we would need `2⁶ = 64` values.

Homework: (based on Nilsson, 19.4, page 340) Consider the following Bayesian network:

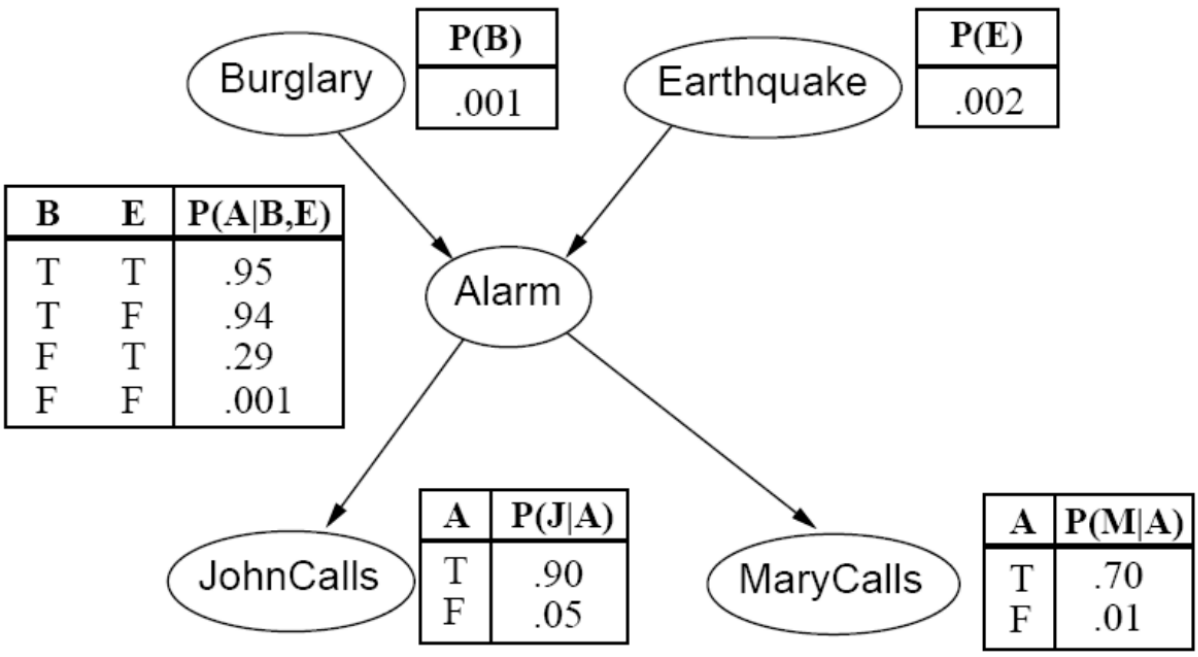

Figure 2: Nielsson, page 341

Compute `p(¬J, ¬M, A, B, E)`. This is the probability that there is both an earthquake and a burglary, the alarm rings, but neither John nor Mary call.

Exercise: Compute `p(¬J, ¬M, B, E)` (this is exercise 19.4 in Nielsson).

Minimum description length (MDL) principle

The ASCII code used to represent alphabetic characters of the English language (and largely replaced by Unicode nowadays) used one byte (eight bits) per character. Thus, to encode a string of 100 characters, 800 bits were needed. Quite often, this encoding was very wasteful, and substantial memory savings could be achieved by *compressing* the string (for example by using *zip* or *rar*). Compression uses a different, string-specific encoding, with the property that more frequent characters receive shorter codes. For example, to encode `nine` we could use

```
'n' → 0
'i' → 10
'e' → 11
```

so that

```
'nine' → 010011
```

It turns out that an optimal encoding scheme requires, for every character `c`, about $-\log_2 f(c)$ bits, where `f(c)` is the frequency with which the character `c` appears in the string.

More generally, in order to encode messages drawn at random from a finite set, the optimal encoding scheme will associate to message m_i, having probability p_i, about $-\log_2 p_i$ bits.

We these preliminaries, we return to the definition of the MAP hypothesis. We have

$$
\begin{aligned}
&h_{map} = \operatorname{argmax}_h\ p(d \mid h) * p(h) \\
&\text{iff} \\
&h_{map} = \operatorname{argmax}_h\ \log_2 (p(d \mid h) * p(h)) \\
&\text{iff} \\
&h_{map} = \operatorname{argmax}_h\ \log_2 p(d \mid h) + \log_2 p(h) \\
&\text{iff} \\
&h_{map} = \operatorname{argmin}_h\ -\log_2 p(d \mid h) - \log_2 p(h)
\end{aligned}
$$

Here, the expression $-\log_2 p(h)$ represents the length that we associate to hypothesis `h` under the optimal encoding. Thus, $-\log_2 p(h) =$ `length(optimal_encode(h))`.

Similarly, the expression $-\log_2 p(d \mid h)$ represents the length associated to the data `d` under the assumption that hypothesis `h` is the correct one. We write this as `length(optimal_encode(d | h))`. Therefore

$$h_{map} = \operatorname{argmin}_h\ \texttt{length(optimal\_encode(d | h))} + \texttt{length(optimal\_encode(h))}$$

In other words, the MAP hypothesis is the one that minimises the sum of two description lengths: that of the hypothesis itself, and that of the given data, assuming that hypothesis to be the correct one. This can be seen as a mathematical interpretation of Occam's razor.