

There are many representations of conditional probabilities and factors:

- Tables
- Decision Trees
- Deterministic System with Noisy Inputs
 - ▶ Weighted Logical Formulae
 - ▶ Probabilistic Programs
- Noisy-or
- Logistic Function
- Neural Networks

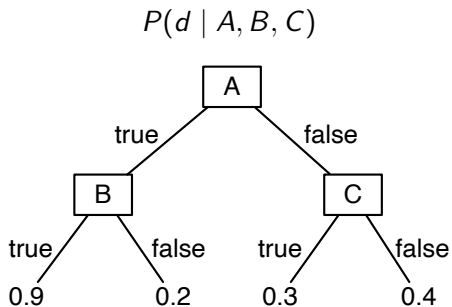
Tabular Representation

	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	Prob
	true	true	true	true	0.9
	true	true	true	false	0.1
	true	true	false	true	0.9
	true	true	false	false	0.1
	true	false	true	true	0.2
	true	false	true	false	0.8
	true	false	false	true	0.2
$P(D A, B, C) :$	true	false	false	false	0.8
	false	true	true	true	0.3
	false	true	true	false	0.7
	false	true	false	true	0.4
	false	true	false	false	0.6
	false	false	true	true	0.3
	false	false	true	false	0.7
	false	false	false	true	0.4
	false	false	false	false	0.6

Python:

```
pd = [[[[0.6,0.4],[0.7,0.3]],[[0.6,0.4],[0.7,0.3]]],  
      [[0.8,0.2],[0.8,0.2]],[[0.1,0.9],[0.1,0.9]]]  
pd[1][0][1][0]
```

Decision Tree Representation



Deterministic System with Noisy Inputs: Weighted Logical Formulae

$$d \leftrightarrow ((a \wedge b \wedge n_0) \\ \vee (a \wedge \neg b \wedge n_1) \\ \vee (\neg a \wedge c \wedge n_2) \\ \vee (\neg a \wedge \neg c \wedge n_3))$$

n_i are independent:

$$P(n_0) = 0.9$$

$$P(n_1) = 0.2$$

$$P(n_2) = 0.3$$

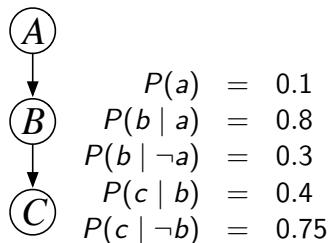
$$P(n_3) = 0.4$$

Deterministic System with Noisy Inputs: Probabilistic Programming

Probabilistic Program: a program with randomized inputs

```
if a:  
    if b:  
        d = flip(0.9)  
    else:  
        d = flip(0.2)  
else:  
    if c:  
        d = flip(0.3)  
    else:  
        d = flip(0.4)  
where  $flip(p) = (random() < p)$ 
```

Representing Belief Networks



$P(a) = 0.1,$
 $P(b|a) = 0.8, P(b|\neg a) = 0.3,$
 $P(c|b) = 0.4, P(c|\neg b) = 0.75.$

$b \leftrightarrow (a \wedge b|a) \vee (\neg a \wedge b|\neg a)$
 $c \leftrightarrow (b \wedge c|b) \vee (\neg b \wedge c|\neg b)$

```
a = flip(0.1)
if a:
    b = flip(0.8)
else:
    b = flip(0.3)
if b:
    c = flip(0.4)
else:
    c = flip(0.75)

flip(p) = (random() < p)
```

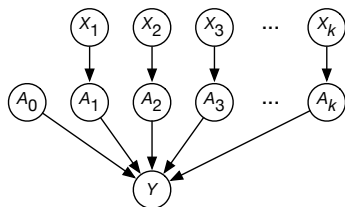
- The robot is wet if it gets wet from rain or coffee or sprinkler or another reason. They each have a probability of making the robot wet \rightarrow **noisy-or**.
- $P(Y | X_1, X_2, \dots, X_k)$, with all variables Boolean, is defined using:

$$y \leftrightarrow n_0 \vee (n_1 \wedge x_1) \vee \dots \vee (n_k \wedge x_k).$$

where n_i are unconditionally independent **noise variables**, with $P(n_i) = w_i$, and x_i means $X_i = \text{true}$.

Noisy-or — alternative definition

Noisy-or $P(Y | X_1, X_2, \dots, X_k)$ can be defined using $k + 1$ Boolean variables A_0, A_1, \dots, A_k , where for each $i > 0$, A_i has X_i as its only parent.



$$P(A_0) = w_0$$

$$P(A_i = \text{true} \mid X_i = \text{true}) = w_i \text{ for } i > 0$$

$$P(A_i = \text{true} \mid X_i = \text{false}) = 0 \text{ for } i > 0$$

$$P(Y \mid A_0, A_1, \dots, A_k) = \begin{cases} 1 & \text{if } \exists i A_i \text{ is true} \\ 0 & \text{if } \forall i A_i \text{ is false} \end{cases}$$

Noisy-or: Example

- Suppose the robot could get wet from rain or coffee.
- There is a probability that it gets wet from rain if it rains, and a probability that it gets wet from coffee if it has coffee, and a probability that it gets wet for other reasons.
- We could have:
 $P(\text{wet_from_rain} \mid \text{rain}) = 0.3,$
 $P(\text{wet_from_coffee} \mid \text{coffee}) = 0.2$
 $P(\text{wet_for_other_reasons}) = 0.1.$
- The robot is wet if it wet from rain, wet from coffee, or wet for other reasons.

$$\text{wet} \leftrightarrow \text{wet_from_rain} \vee \text{wet_from_coffee} \vee \text{wet_for_other_reasons}$$

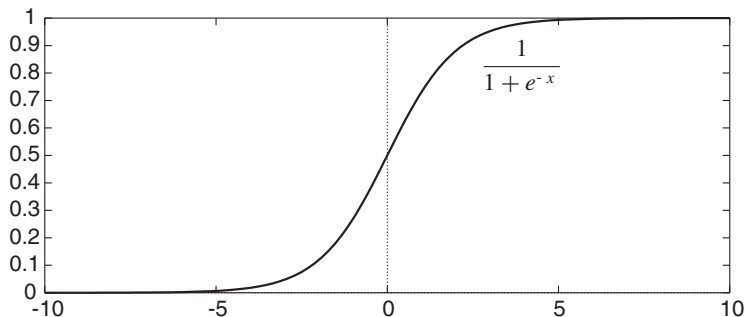
$$\begin{aligned}P(h | e) &= \frac{P(h \wedge e)}{P(e)} \\&= \frac{P(h \wedge e)}{P(h \wedge e) + P(\neg h \wedge e)} \\&= \frac{1}{1 + P(\neg h \wedge e)/P(h \wedge e)} \\&= \frac{1}{1 + e^{-\log P(h \wedge e)/P(\neg h \wedge e)}} \\&= \textit{sigmoid}(\log \textit{odds}(h | e))\end{aligned}$$

$$\textit{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

$$\textit{odds}(h | e) = \frac{P(h \wedge e)}{P(\neg h \wedge e)}$$

Logistic Functions

A conditional probability is the sigmoid of the log-odds.



$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

A **logistic function** is the sigmoid of a linear function.
If the odds form a product, the log odds give a sum.
What independence is assumed?

- $odds(h | e) = \frac{P(h \wedge e)}{P(\neg h \wedge e)} = \frac{P(h | e)}{P(\neg h | e)} = \frac{P(h | e)}{1 - P(h | e)}$
- Decomposing $P(h \wedge e)$ into $P(e | h) * P(h)$

$$odds(h | e) = \frac{P(e | h)}{P(e | \neg h)} * \frac{P(h)}{P(\neg h)}$$

- $\frac{P(h)}{P(\neg h)} = \frac{P(h)}{1-P(h)}$ is the **prior odds**
- $\frac{P(e|h)}{P(e|\neg h)}$ is the **likelihood ratio**.
- If $e = e_1 \wedge \dots \wedge e_k$, and e_i & e_j are independent given h

$$\frac{P(e | h)}{P(e | \neg h)} = \prod_{i=1}^k \frac{P(e_i | h)}{P(e_i | \neg h)}$$

$$\log odds(h | e) = \log \frac{P(h)}{P(\neg h)} + \sum_{i=1}^k \log \frac{P(e_i | h)}{P(e_i | \neg h)}$$

X_1	X_2	X_3	X_4	$Prob$		
False	False	False	False	$p_0=0.01$		
True	False	False	False	$p_1=0.05$		
False	True	False	False	$p_2=0.1$		
False	False	True	False	$p_3=0.2$		
False	False	False	True	$p_4=0.2$		
X_1	X_2	X_3	X_4	A	B	
False	False	True	True	0.353535	0.860870	
False	True	False	True	0.272727	0.733333	
False	True	True	True	0.412305	0.985520	
True	False	False	True	0.232323	0.565714	
True	False	True	True	0.379655	0.969916	
True	True	False	False	0.136364	0.366667	
True	True	False	True	0.302112	0.934764	
True	True	True	True	0.436050	0.997188	

A is **noisy-or**. B is **logistic regression**.

What if p_0 is increased to 0.05 with $p_1 \dots p_4$ fixed?

Logistic Representation of Conditional Probability

Can the running example be represented using a logistic function?

$$\begin{aligned} P(d \mid A, B, C) = & \text{sigmoid}(\text{logit}(0.9) * A * B \\ & + \text{logit}(0.2) * A * (1 - B) \\ & + \text{logit}(0.3) * (1 - A) * C \\ & + \text{logit}(0.4) * (1 - A) * (1 - C)) \end{aligned}$$

where *logit* is the inverse of *sigmoid*.

$$\begin{aligned} P(d \mid A, B, C) = & \text{sigmoid}(\text{logit}(0.4) \\ & + (\text{logit}(0.2) - \text{logit}(0.4)) * A \\ & + (\text{logit}(0.9) - \text{logit}(0.2)) * A * B \\ & + \dots \end{aligned}$$

- Allowing products in the features is **canonical representation**, which can represent any discrete conditional probability.
- This is the representation learned by **gradient boosted trees**.

- Build a neural network to predict D from A, B, C
- For Boolean outputs, typically use a sigmoid activation at the output, and optimize with log likelihood.
 - such a neural network is a logistic regression model with learned features
- For other discrete variables, **softmax** gives a probability distribution:

$$\text{softmax}((\alpha_1 \dots \alpha_k))_i = \frac{\exp(\alpha_i)}{\sum_{j=1}^k \exp(\alpha_j)}$$

- For other domains, a **Bayesian neural network** represents the distribution over the outputs (not just a point prediction).

Sigmoid and Softmax

- How are sigmoid and softmax related?

$$\begin{aligned}\text{sigmoid}(x) &= \frac{1}{\exp(-x) + 1} \\ &= \frac{\exp(x)}{\exp(0) + \exp(x)} \\ &= \text{softmax}((0, x))_2\end{aligned}$$

$(0, x)$ corresponds to the values $(\text{false}, \text{true})$

$\text{softmax}((0, x))_2$ is the second component of the result

- What happens in a softmax if a constant is added to each component?
- How can you avoid overflow or underflow in softmax? ($\exp(750)$ will overflow for most modern CPUs, and $\exp(-750)$ results in zero.)
- Why not use same “trick” as sigmoid – setting one value to 0 – to reduce the number of parameters in softmax?

