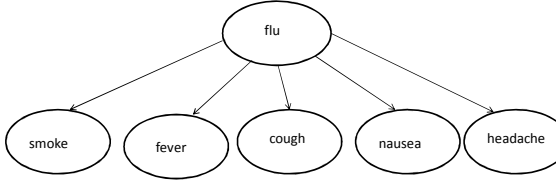


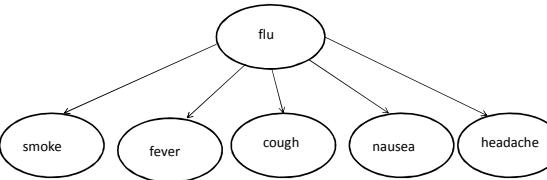
### Reasoning under uncertainty 3



A Bayesian network diagram with a root node 'flu' and five child nodes: 'smoke', 'fever', 'cough', 'nausea', and 'headache'. Arrows point from 'flu' to each of the child nodes.

$$P(\text{flu} \wedge \text{smoke} \wedge \text{fever} \wedge \text{cough} \wedge \text{nausea} \wedge \text{headache})$$

$$= P(\text{flu})P(\text{smoke} | \text{flu})P(\text{fever} | \text{flu})P(\text{cough} | \text{flu})$$

$$P(\text{nausea} | \text{flu})P(\text{headache} | \text{flu})$$


A Bayesian network diagram with a root node 'flu' and five child nodes: 'smoke', 'fever', 'cough', 'nausea', and 'headache'. Arrows point from 'flu' to each of the child nodes.

More generally, for classification :

$$P(C = c_j | X_1 = x_1, \dots, X_n = x_n)$$

$$= P(C = c_j) \prod_i P(X_i = x_i | C = c_j)$$

**"Naive Bayes"**

### Naive Bayes Classification Algorithm

1. From training data, estimate  $P(X_i = x_i | C = c_j)$  for each  $i, j$
2. From training data, estimate  $P(C = c_j)$  for each  $j$
3. For new instance  $X_i = x_i, i=1..n$ , use previously estimated probabilities to calculate:
 
$$\operatorname{argmax}_{\text{class}=j} P(C = c_j) \prod_i P(X_i = x_i | C = c_j)$$
4. Return  $c_j$ .

### Example of Naive Bayes Classification

Suppose you have the following training data concerning four patients, their symptoms, and whether they have the flu:

	Cough	Fever	Headache	Flu
Patient 1	yes	yes	yes	yes
Patient 2	yes	no	no	no
Patient 3	no	yes	yes	yes
Patient 4	no	yes	yes	no

How would a naive Bayes classifier, trained on this data, classify ("flu" or "not flu") a new patient who is coughing, tired, and has a fever?

### Issues in Bayesian Networks

- Building or learning network topology
- Assigning or learning conditional probability tables
- Approximate inference via sampling

### Learning network topology

- Many different approaches, including:
  - Heuristic search, with evaluation based on information theory measures
  - Genetic algorithms
  - Using “meta” Bayesian networks!

### Learning conditional probabilities

- In general, random variables are not binary, but real-valued
- Conditional probability tables  $\Rightarrow$  conditional probability distributions
- Estimate parameters of these distributions from data
- If data is missing on one or more variables, use “expectation maximization” algorithm

### Approximate inference via sampling

- Recall: We can calculate full joint probability distribution from network.

$$P(X_1, \dots, X_d) = \prod_{i=1}^d P(X_i | \text{parents}(X_i))$$

where  $\text{parents}(X_i)$  denotes specific values of parents of  $X_i$ .

- We can do diagnostic, causal, and inter-causal inference
- But if there are a lot of nodes in the network, this can be very slow!

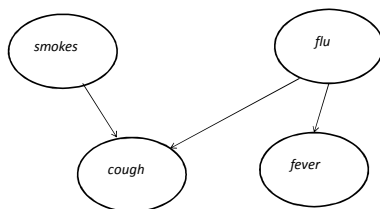
Need efficient algorithms to do approximate calculations!

### Markov Chain Monte Carlo Sampling

- One of most common methods used in real applications.
- **Recall that:** By construction of Bayesian network, a node is conditionally independent of its non-descendants, given its parents.
- **Also recall that:** a node can be conditionally dependent on its children and on the other parents of its children. (Why?)
- **Definition:** The *Markov blanket* of a variable  $X_i$  is  $X_i$ 's parents, children, and children's other parents.

### Example

- What is the Markov blanket of *cough*? of *flu*?



- **Theorem:** A node  $X_i$  is conditionally independent of all other nodes in the network, given its Markov blanket.

### Markov Chain Monte Carlo (MCMC) Sampling

- Start with random sample from variables:  $(x_1, \dots, x_n)$ . This is the current "state" of the algorithm.
- Next state: Randomly sample value for one non-evidence variable  $X_i$ , conditioned on current values in "Markov Blanket" of  $X_i$ .

### Example

- Query:** What is  $P(\text{cough} | \text{smoke})$ ?
- MCMC:**
  - Random sample, with evidence variables fixed:
 

<i>flu</i>	<b>smoke</b>	<i>fever</i>	<i>cough</i>
true	<b>true</b>	false	true
  - Repeat:
    - Sample *flu* probabilistically, given current values of its Markov blanket: *smoke* = true, *fever* = false, *cough* = true
 

Suppose result is *false*. New state:

<i>flu</i>	<b>smoke</b>	<i>fever</i>	<i>cough</i>
false	<b>true</b>	false	true

- Sample *cough*, given current values of its Markov blanket:
 

*smoke* = true, *flu* = false

Suppose result is *true*.

New state:

<i>flu</i>	<b>smoke</b>	<i>fever</i>	<i>cough</i>
false	<b>true</b>	false	true
- Sample *fever*, given current values of its Markov blanket:
 

*flu* = false

Suppose result is *true*.

New state:

<i>flu</i>	<b>smoke</b>	<i>fever</i>	<i>cough</i>
false	<b>true</b>	true	true

- Each sample contributes to estimate for query  $P(\text{cough} | \text{smoke})$
- Suppose we perform 100 such samples, 20 with *cough* = true and 80 with *cough* = false.
- Then answer to the query is  $P(\text{cough} | \text{smoke}) = .20$
- Theorem:** MCMC settles into behavior in which each state is sampled exactly according to its posterior probability, given the evidence.

### Applying Bayesian Reasoning to Speech Recognition

- Task:** Identify sequence of words uttered by speaker, given acoustic signal.
- Uncertainty introduced by noise, speaker error, variation in pronunciation, homonyms, etc.
- Thus speech recognition is viewed as problem of probabilistic inference.

- At each time step, we want to compute  $P(\text{Words} | \text{Signal } \mathbf{S})$ .
- We know from Bayes rule:
 
$$P(\text{Words} | \mathbf{S}) = \frac{P(\mathbf{S} | \text{Words})P(\text{Words})}{P(\mathbf{S})}$$
- $P(\mathbf{S} | \text{Words})$ : **acoustic model**
  - E.g. For each word, probability distribution over phones, and for each phone, probability distribution over acoustic signals (which can vary in pitch, speed, volume).
- $P(\text{Words})$ : **language model**, which specifies prior probability of each utterance.
  - E.g. **bigram model**: probability of each word following each other word.

- Speech recognition typically makes assumptions:
  1. Process underlying change is itself “stationary” i.e., state transition probabilities don’t change
  2. Current state *Words* depends on only a finite history of previous states (“**Markov assumption**”).
  - Typically assumed that current state depends only on immediately previous state.

## Acoustic model

### Phones

All human speech is composed from 40-50 phones, determined by the configuration of articulators (lips, teeth, tongue, vocal cords, air flow)

Form an intermediate level of hidden states between words and signal  
 ⇒ acoustic model = pronunciation model + phone model

ARPAbet designed for American English

[iy]	beat	[b]	bet	[p]	pet
[ih]	bit	[ch]	chet	[r]	rat
[ey]	bet	[d]	debt	[s]	set
[ao]	bought	[hh]	hat	[th]	thick
[ow]	boat	[hw]	high	[dh]	that
[er]	Bert	[l]	let	[w]	wet
[ix]	roses	[ng]	sing	[en]	button
:	:	:	:	:	:

E.g., “ceiling” is [s iy l ih ng] / [s iy l ix ng] / [s iy l en]

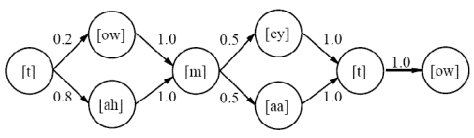
From Russell and Norvig, *Artificial Intelligence*

- ### Hidden Markov Models
- **Markov model:** Given state  $X_t$ , what is probability of transitioning to next state  $X_{t+1}$ ?
    - E.g., word bigram probabilities give  $P(word_{t+1} | word_t)$
  - **Hidden Markov model:** There are observable states (e.g., signal  $S$ ) and “hidden” states (e.g., *Words*). **HMM** represents probabilities of hidden states given observable states.

### Word pronunciation models

Each word is described as a distribution over phone sequences

Distribution represented as an HMM transition model



$P(\{towmeytow\} | \text{“tomato”}) = P(\{towaatow\} | \text{“tomato”}) = 0.1$   
 $P(\{tahmeytow\} | \text{“tomato”}) = P(\{tahmaatow\} | \text{“tomato”}) = 0.4$

Structure is created manually, transition probabilities learned from data

From Russell and Norvig, *Artificial Intelligence*

## Language Model

### Language model

Prior probability of a word sequence is given by chain rule:

$$P(w_1 \cdots w_n) = \prod_{i=1}^n P(w_i | w_1 \cdots w_{i-1})$$

Bigram model:

$$P(w_i | w_1 \cdots w_{i-1}) \approx P(w_i | w_{i-1})$$

Train by counting all word pairs in a large text corpus

More sophisticated models (trigrams, grammars, etc.) help a little bit

From Russell and Norvig, *Artificial Intelligence*