

# Hidden Markov Models

Based on

“Foundations of Statistical NLP” by C. Manning & H. Schütze, ch. 9  
MIT Press, 2002

# PLAN

- 1 Markov Models  
Markov assumptions
- 2 Hidden Markov Models
- 3 Fundamental questions for HMMs
  - 3.1 Probability of an observation sequence:  
the Forward algorithm, the Backward algorithm
  - 3.2 Finding the “best” sequence: the Viterbi algorithm
  - 3.3 HMM parameter estimation:  
the Forward-Backward (EM) algorithm
- 4 HMM extensions
- 5 Applications

## 1 Markov Models (generally)

Markov Models are used to model a sequence of random variables in which each element depends on previous elements.

$$X = \langle X_1 \dots X_T \rangle \quad X_t \in S = \{s_1, \dots, s_N\}$$

$X$  is also called a **Markov Process** or **Markov Chain**.

$S$  = set of states

$\Pi$  = initial state probabilities

$$\pi_i = P(X_1 = s_i); \sum_{i=1}^N \pi_i = 1$$

$A$  = transition probabilities:

$$a_{ij} = P(X_{t+1} = s_j | X_t = s_i); \sum_{j=1}^N a_{ij} = 1 \quad \forall i$$

## Markov assumptions

- **Limited Horizon:**

$$P(X_{t+1} = s_i | X_1 \dots X_t) = P(X_{t+1} = s_i | X_t)$$

(first-order Markov model)

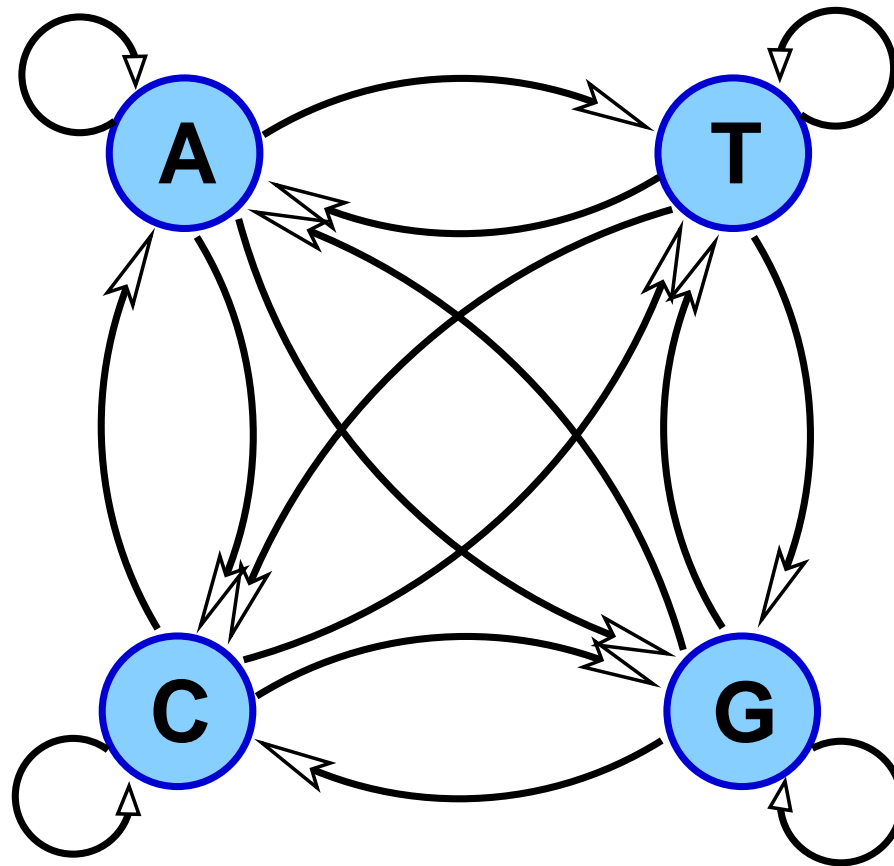
- **Time Invariance:**  $P(X_{t+1} = s_j | X_t = s_i) = p_{ij} \quad \forall t$

## Probability of a Markov Chain

$$\begin{aligned} P(X_1 \dots X_T) &= P(X_1)P(X_2|X_1)P(X_3|X_1X_2) \dots \\ &\quad P(X_T|X_1X_2 \dots X_{T-1}) \\ &= P(X_1)P(X_2|X_1)P(X_3|X_2) \dots P(X_T|X_{T-1}) \\ &= \pi_{X_1} \prod_{t=1}^{T-1} a_{X_t X_{t+1}} \end{aligned}$$

## A 1st Markov chain example: DNA

(from [Durbin et al., 1998])



Note:  
Here we leave  
transition  
probabilities  
unspecified.

## A 2nd Markov chain example: CpG islands in DNA sequences

Maximum Likelihood estimation of parameters using real data (+ and -)

$$a_{st}^+ = \frac{c_{st}^+}{\sum_{t'} c_{st'}^+} \quad a_{st}^- = \frac{c_{st}^-}{\sum_{t'} c_{st'}^-}$$

+	A	C	G	T	-	A	C	G	T
A	0.180	0.274	0.426	0.120	A	0.300	0.205	0.285	0.210
C	0.171	0.368	0.274	0.188	C	0.322	0.298	0.078	0.302
G	0.161	0.339	0.375	0.125	G	0.248	0.246	0.298	0.208
T	0.079	0.355	0.384	0.182	T	0.177	0.239	0.292	0.292

## Using log likelihood (*log-odds*) ratios for discrimination

$$S(x) = \log_2 \frac{P(x \mid \text{model } +)}{P(x \mid \text{model } -)} = \sum_{i=1}^L \log_2 \frac{a_{x_{i-1}x_i}^+}{a_{x_{i-1}x_i}^-} = \sum_{i=1}^L \beta_{x_{i-1}x_i}$$

$\beta$	<i>A</i>	<i>C</i>	<i>G</i>	<i>T</i>
<i>A</i>	−0.740	0.419	0.580	−0.803
<i>C</i>	−0.913	0.302	1.812	−0.685
<i>G</i>	−0.624	0.461	0.331	−0.730
<i>T</i>	−1.169	0.573	0.393	−0.679

## 2 Hidden Markov Models

$\mathbf{K}$  = output alphabet =  $\{k_1, \dots, k_M\}$

$\mathbf{B}$  = output emission probabilities:

$$b_{ijk} = P(O_t = k | X_t = s_i, X_{t+1} = s_j)$$

Notice that  $b_{ijk}$  does not depend on  $t$ .

In HMMs we only observe a probabilistic function of the state sequence:  $\langle O_1 \dots O_T \rangle$

When the state sequence  $\langle X_1 \dots X_T \rangle$  is also observable:  
Visible Markov Model (VMM)

### Remark:

In all our subsequent examples  $b_{ijk}$  is independent of  $j$ .

## A program for a HMM

$t = 1;$

start in state  $s_i$  with probability  $\pi_i$  (i.e.,  $X_1 = i$ );

forever do

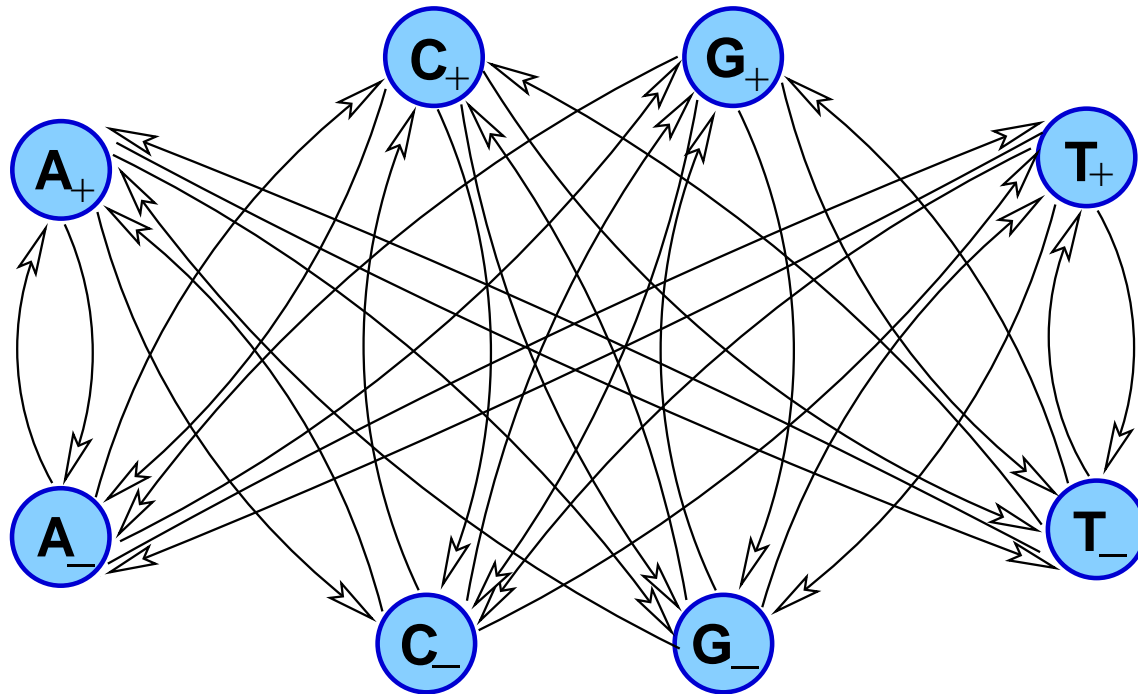
    move from state  $s_i$  to state  $s_j$  with prob.  $a_{ij}$  (i.e.,  $X_{t+1} = j$ );

    emit observation symbol  $O_t = k$  with probability  $b_{ijk}$ ;

$t = t + 1;$

## A 1st HMM example: CpG islands

(from [Durbin et al., 1998])

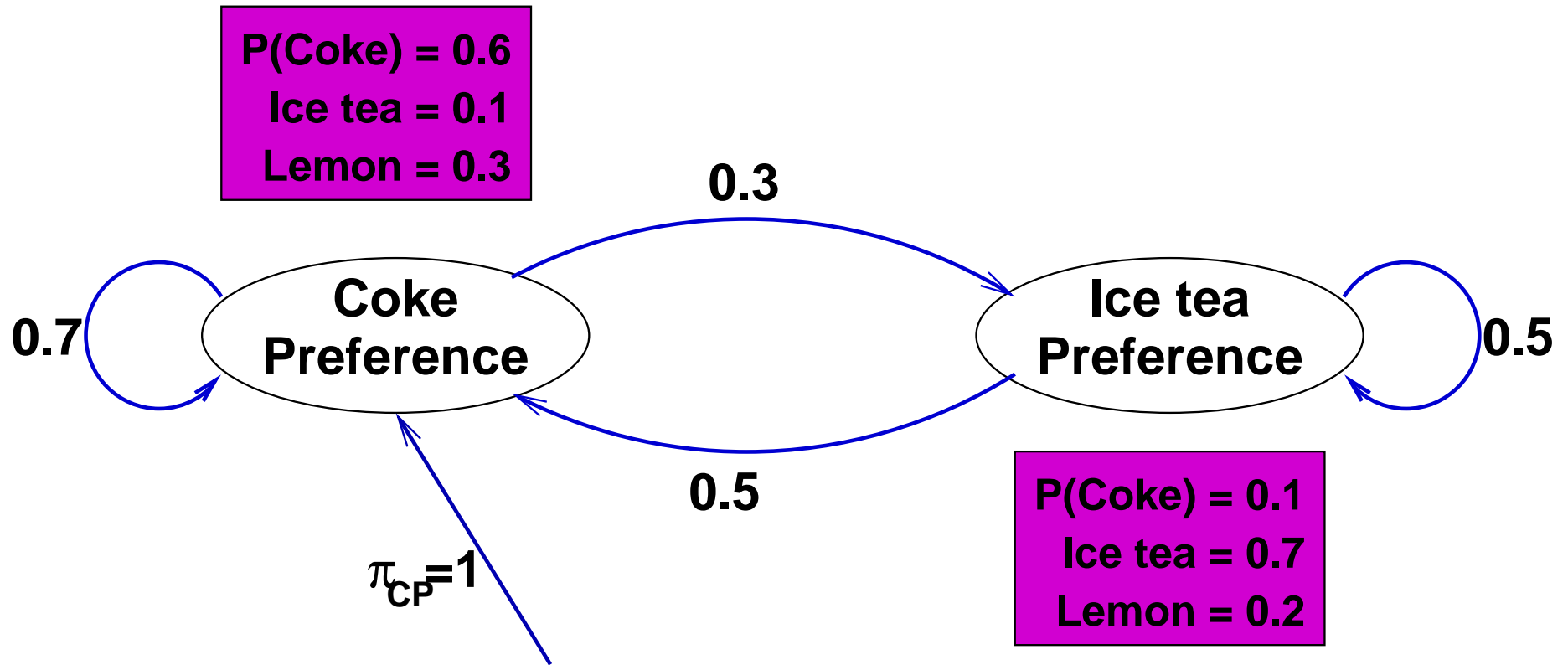


*Notes:*

1. In addition to the transitions shown, there is also a complete set of transitions within each set (+ respectively -).
2. Transition probabilities in this model are set so that within each group they are close to the transition probabilities of the original model, but there is also a small chance of switching into the other component. Overall, there is more chance of switching from '+' to '-' than viceversa.

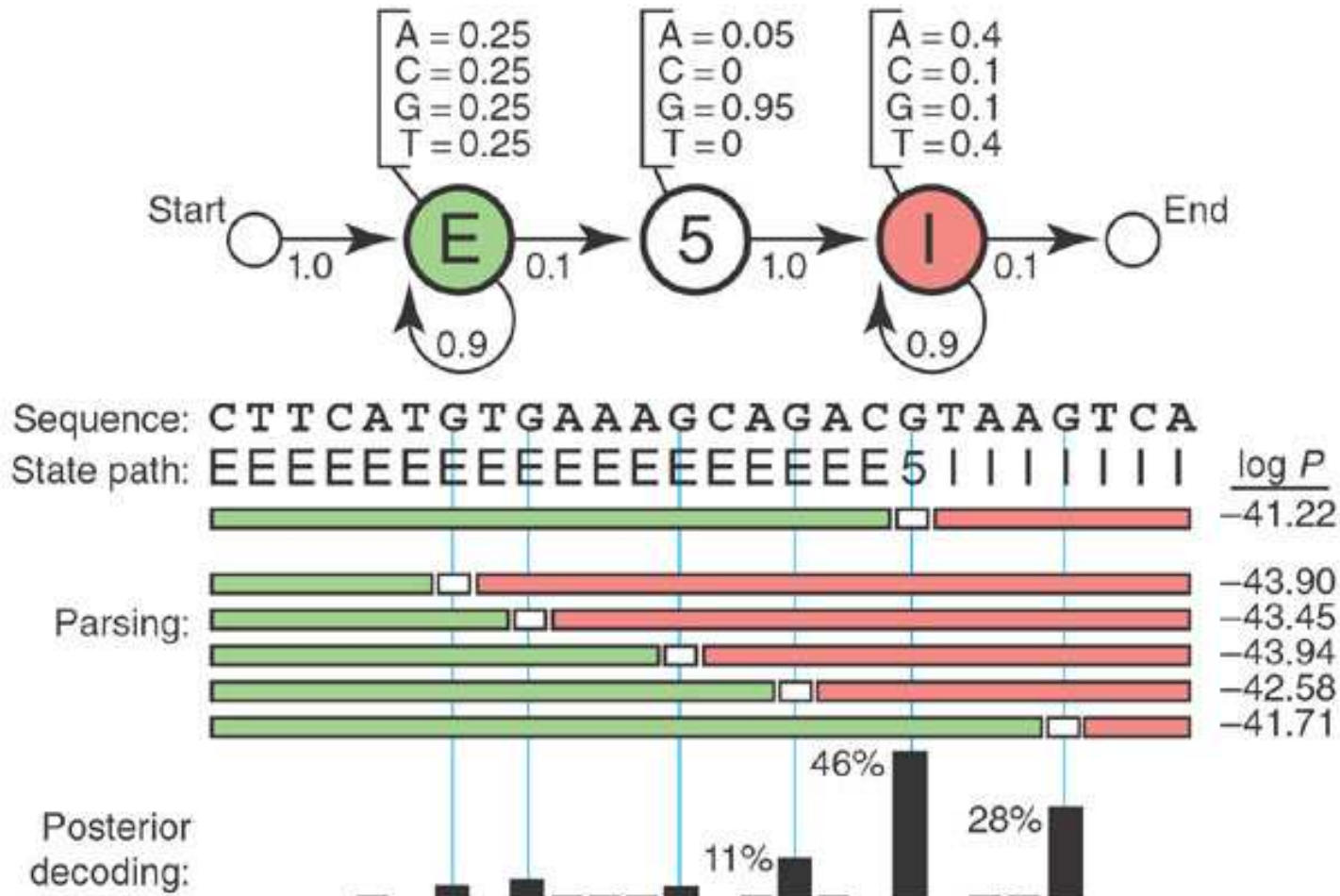
## A 2nd HMM example: The crazy soft drink machine

(from [Manning & Schütze, 2000])



# A 3rd example: A tiny HMM for 5' splice site recognition

(from [Eddy, 2004])



## **3** Three fundamental questions for HMMs

### 1. Probability of an Observation Sequence:

Given a model  $\mu = (A, B, \Pi)$  over  $S, K$ , how do we (efficiently) compute the likelihood of a particular sequence,  $P(O|\mu)$ ?

### 2. Finding the “Best” State Sequence:

Given an observation sequence and a model, how do we choose a state sequence  $(X_1, \dots, X_{T+1})$  to best explain the observation sequence?

### 3. HMM Parameter Estimation:

Given an observation sequence (or corpus thereof), how do we acquire a model  $\mu = (A, B, \Pi)$  that best explains the data?

## 3.1 Probability of an observation sequence

$$P(O|X, \mu) = \prod_{t=1}^T P(O_t|X_t, X_{t+1}, \mu) = b_{X_1 X_2 O_1} b_{X_2 X_3 O_2} \dots b_{X_T X_{T+1} O_T}$$

$$P(O, \mu) = \sum_X P(O|X, \mu) P(X, \mu) = \sum_{X_1 \dots X_{T+1}} \pi_{X_1} \prod_{t=1}^T a_{X_t X_{t+1}} b_{X_t X_{t+1} O_t}$$

**Complexity** :  $(2T + 1)N^{T+1}$ , too inefficient

**better** : use dynamic prog. to store partial results

$$\alpha_i(t) = P(O_1 O_2 \dots O_{t-1}, X_t = s_i | \mu).$$

## 3.1.1 Probability of an observation Sequence: The Forward algorithm

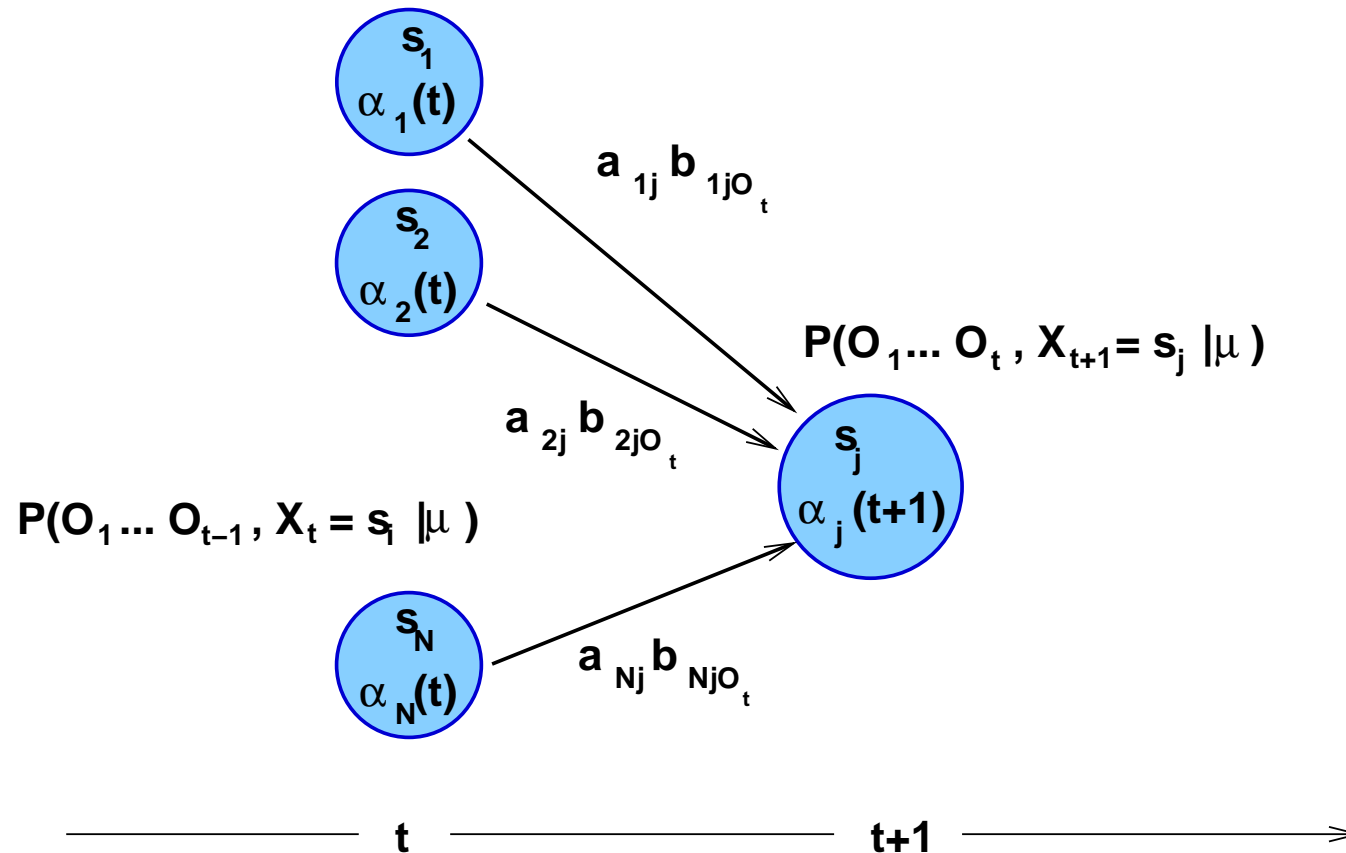
14.

1. **Initialization:**  $\alpha_i(1) = \pi_i$ , for  $1 \leq i \leq N$
2. **Induction:**  $\alpha_j(t+1) = \sum_{i=1}^N \alpha_i(t) a_{ij} b_{ij} O_t$ ,  $1 \leq t \leq T$ ,  $1 \leq j \leq N$
3. **Total:**  $P(O|\mu) = \sum_{i=1}^N \alpha_i(T+1)$ . **Complexity:**  $2N^2T$

**Proof of induction step:**

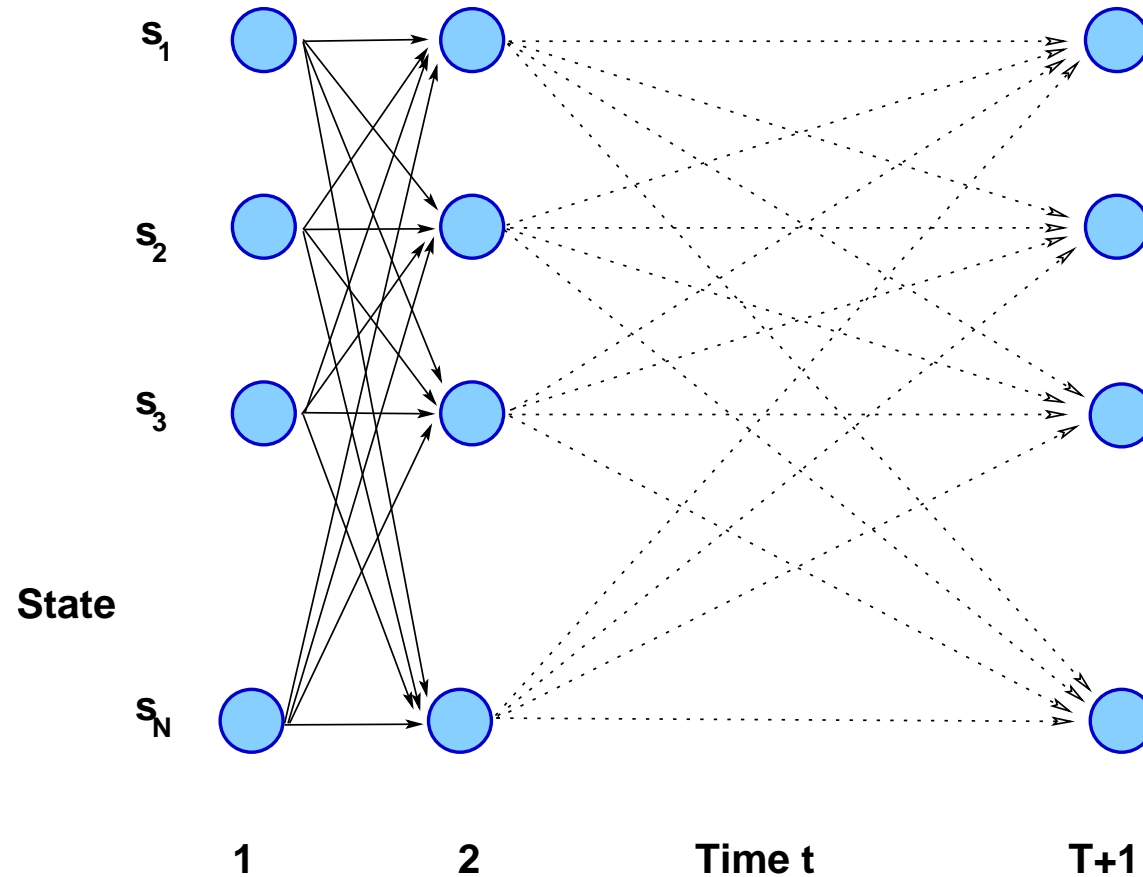
$$\begin{aligned}\alpha_j(t+1) &= P(O_1 O_2 \dots O_{t-1} O_t, X_{t+1} = j | \mu) \\ &= \sum_{i=1}^N P(O_1 O_2 \dots O_{t-1}, X_t = i | \mu) P(O_t, X_{t+1} = j | O_1 O_2 \dots O_{t-1}, X_t = i, \mu) \\ &= \sum_{i=1}^N \alpha_i(t) P(O_t, X_{t+1} = j | X_t = i, \mu) \\ &= \sum_{i=1}^N \alpha_i(t) P(O_t | X_t = i, X_{t+1} = j, \mu) P(X_{t+1} = j | X_t = i, \mu) = \sum_{i=1}^N \alpha_i(t) b_{ij} O_t a_{ij}\end{aligned}$$

## Closeup of the Forward update step



## Trellis

Each node  $(s_i, t)$  stores information about paths through  $s_i$  at time  $t$ .



### 3.1.2 Probability of an observation sequence: The Backward algorithm

$$\beta_i(t) = P(O_t \dots O_T | X_t = i, \mu)$$

1. **Initialization:**  $\beta_i(T + 1) = 1$ , for  $1 \leq i \leq N$
2. **Induction:**  $\beta_i(t) = \sum_{j=1}^N a_{ij} b_{ij} O_t \beta_j(t + 1)$ ,  $1 \leq t \leq T$ ,  $1 \leq i \leq N$
3. **Total:**  $P(O | \mu) = \sum_{i=1}^N \pi_i \beta_i(1)$

**Complexity:**  $2N^2T$

## The Backward algorithm: Proofs

**Induction:**

$$\begin{aligned}
 \beta_i(t) &= P(O_t O_{t+1} \dots O_T | X_t = i, \mu) \\
 &= \sum_{j=1}^N P(O_t O_{t+1} \dots O_T | X_t = i, X_{t+1} = j, \mu) P(X_{t+1} = j | X_t = i, \mu) \\
 &= \sum_{j=1}^N P(O_{t+1} \dots O_T | O_t, X_t = i, X_{t+1} = j, \mu) P(O_t | X_t = i, X_{t+1} = j, \mu) a_{ij} \\
 &= \sum_{j=1}^N P(O_{t+1} \dots O_T | X_{t+1} = j, \mu) b_{ij O_t} a_{ij} = \sum_{j=1}^N \beta_j(t+1) b_{ij O_t} a_{ij}
 \end{aligned}$$

**Total:**

$$P(O|\mu) = \sum_{i=1}^N P(O_1 O_2 \dots O_T | X_1 = i, \mu) P(X_1 = i | \mu) = \sum_{i=1}^N \beta_i(1) \pi_i$$

## Combining Forward and Backward probabilities

$$P(O, X_t = i | \mu) = \alpha_i(t)\beta_i(t)$$

$$P(O | \mu) = \sum_{i=1}^N \alpha_i(t)\beta_i(t) \text{ for } 1 \leq t \leq T + 1$$

### Proofs:

$$\begin{aligned} P(O, X_t = i | \mu) &= P(O_1 \dots O_T, X_t = i | \mu) = \\ &= P(O_1 \dots O_{t-1}, X_t = i, O_t \dots O_T | \mu) \\ &= P(O_1 \dots O_{t-1}, X_t = i | \mu) P(O_t \dots O_T | O_1 \dots O_{t-1}, X_t = i, \mu) \\ &= \alpha_i(t) P(O_t \dots O_T | X_t = i, \mu) \\ &= \alpha_i(t)\beta_i(t) \end{aligned}$$

$$P(O | \mu) = \sum_{i=1}^N P(O, X_t = i | \mu) = \sum_{i=1}^N \alpha_i(t)\beta_i(t)$$

**Note:** The forward/backward formulae are special cases of the above one.

## 3.2 Finding the “best” state sequence

### 3.2.1 Posterior decoding

One way to find the most likely state sequence underlying the observation sequence: **choose the states individually**

$$\gamma_i(t) = P(X_t = i | O, \mu)$$

$$\hat{X}_t = \operatorname{argmax}_{1 \leq i \leq N} \gamma_i(t) \text{ for } 1 \leq t \leq T + 1$$

**Computing  $\gamma_i(t)$ :**

$$\gamma_i(t) = P(X_t = i | O, \mu) = \frac{P(X_t = i, O | \mu)}{P(O | \mu)} = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^N \alpha_j(t)\beta_j(t)}$$

**Remark:**

$\hat{X}$  maximizes the expected number of states that will be guessed correctly. However, it may yield a quite unlikely/unnatural state sequence.

## Note

Sometimes not the state itself is of interest, but some other property derived from it.

For instance, in the **CpG islands** example, let  $g$  be a function defined on the set of states:  $g$  takes the value 1 for  $A_+, C_+, G_+, T_+$  and 0 for  $A_-, C_-, G_-, T_-$ .

Then

$$\sum_j P(\pi_t = s_j \mid O)g(s_j)$$

designates the posterior probability that the symbol  $O_t$  come from a state in the  $+$  set.

Thus it is possible to find the most probable label of the state at each position in the output sequence  $O$ .

## 3.2.2 Finding the “best” state sequence

### The Viterbi algorithm

Compute the probability of the most likely path

$$\operatorname{argmax}_X P(X|O, \mu) = \operatorname{argmax}_X P(X, O|\mu)$$

through a node in the trellis

$$\delta_i(t) = \max_{X_1 \dots X_{t-1}} P(X_1 \dots X_{t-1}, O_1 \dots O_{t-1}, X_t = s_i | \mu)$$

1. **Initialization:**  $\delta_j(1) = \pi_j$ , for  $1 \leq j \leq N$
2. **Induction:** (see the similarity with the Forward algorithm)

$$\delta_j(t+1) = \max_{1 \leq i \leq N} \delta_i(t) a_{ij} b_{ij} O_t, \quad 1 \leq t \leq T, \quad 1 \leq j \leq N$$

$$\psi_j(t+1) = \operatorname{argmax}_{1 \leq i \leq N} \delta_i(t) a_{ij} b_{ij} O_t, \quad 1 \leq t \leq T, \quad 1 \leq j \leq N$$

3. **Termination and readout of best path:**

$$P(\hat{X}|O, \mu) = \max_{1 \leq i \leq N} \delta_i(T+1)$$

$$\hat{X}_{T+1} = \operatorname{argmax}_{1 \leq i \leq N} \delta_i(T+1), \quad \hat{X}_t = \psi_{\hat{X}_{t+1}}(t+1)$$

**Example:**

Variable calculations for  
the *crazy soft drink ma-*  
*chine* HMM

<i>Output</i> <i>t</i>	<i>lemon</i> 1	<i>ice_tea</i> 2	<i>cola</i> 3	4
$\alpha_{CP}(t)$	1.0	0.21	0.0462	0.021294
$\alpha_{IP}(t)$	0.0	0.09	0.0378	0.010206
$P(o_1 \dots o_{t-1})$	1.0	0.3	0.084	0.0315
$\beta_{CP}(t)$	0.0315	0.045	0.6	1.0
$\beta_{IP}(t)$	0.029	0.245	0.1	1.0
$P(o_1 \dots o_T)$	0.0315			
$\gamma_{CP}(t)$	1.0	0.3	0.88	0.676
$\gamma_{IP}(t)$	0.0	0.7	0.12	0.324
$\hat{X}_t$	<i>CP</i>	<i>IP</i>	<i>CP</i>	<i>CP</i>
$\delta_{CP}(t)$	1.0	0.21	0.0315	0.01323
$\delta_{IP}(t)$	0.0	0.09	0.0315	0.00567
$\psi_{CP}(t)$		<i>CP</i>	<i>IP</i>	<i>CP</i>
$\psi_{IP}(t)$		<i>CP</i>	<i>IP</i>	<i>CP</i>
$\hat{X}_t$	<i>CP</i>	<i>IP</i>	<i>CP</i>	<i>CP</i>
$P(\hat{X})$	0.019404			

### 3.3 HMM parameter estimation

Given a single observation sequence for training, we want to find the model (parameters)  $\mu = (A, B, \pi)$  that best explains the observed data.

Under Maximum Likelihood Estimation, this means:

$$\operatorname{argmax}_{\mu} P(O_{\text{training}}|\mu)$$

There is no known analytic method for doing this.

However we can choose  $\mu$  so as to **locally maximize**  $P(O_{\text{training}}|\mu)$  by an iterative hill-climbing algorithm: **Forward-Backward** (or: **Baum-Welch**), which is a special case of the **EM algorithm**.

## 3.3.1 The Forward-Backward algorithm

### The basic intuition

- Assume some (perhaps randomly chosen) model parameters. Calculate the probability of the observed data.
- Using the above calculation, we can see which transitions and signal emissions were probably used the most; by increasing the probability of these, we will get a higher probability of the observed sequence.
- Iterate, hopefully arriving at an optimal parameter setting.

## The Forward-Backward algorithm: Expectations

Define the probability of traversing a certain arc at time  $t$ , given the observation sequence  $O$

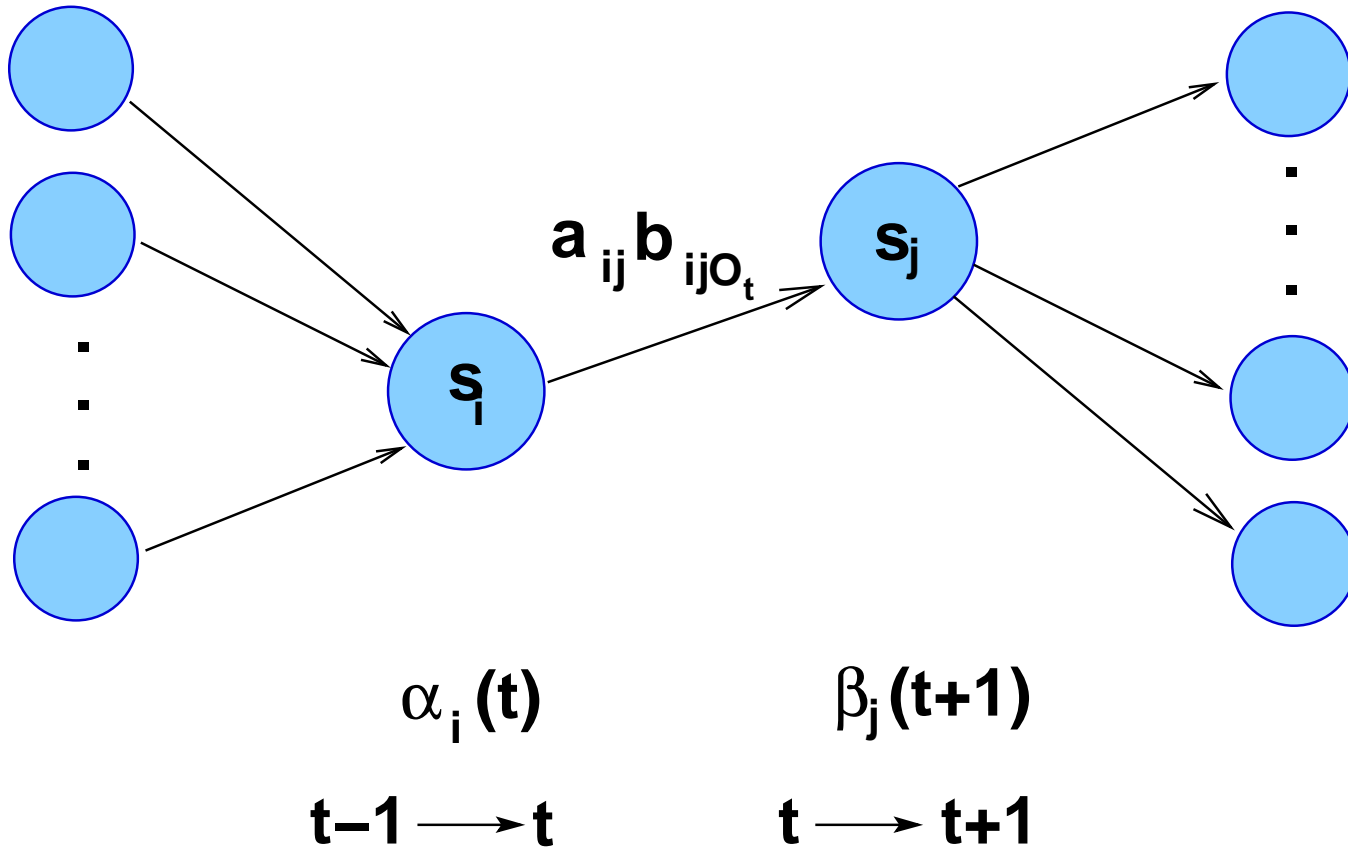
$$p_t(i, j) = P(X_t = i, X_{t+1} = j | O, \mu)$$

$$\begin{aligned} p_t(i, j) &= \frac{P(X_t = i, X_{t+1} = j, O | \mu)}{P(O | \mu)} = \frac{\alpha_i(t) a_{ij} b_{ij O_t} \beta_j(t+1)}{\sum_{m=1}^N \alpha_m(t) \beta_m(t)} \\ &= \frac{\alpha_i(t) a_{ij} b_{ij O_t} \beta_j(t+1)}{\sum_{m=1}^N \sum_{n=1}^N \alpha_m(t) a_{mn} b_{mn O_t} \beta_n(t+1)} \end{aligned}$$

Summing over  $t$ :

$\sum_{t=1}^T p_t(i, j) =$  expected number of transitions from  $s_i$  to  $s_j$  in  $O$

$\sum_{j=1}^N \sum_{t=1}^T p_t(i, j) =$  expected number of transitions from  $s_i$  in  $O$



## The Forward-Backward algorithm: Re-estimation

From  $\mu = (A, B, \Pi)$ , derive  $\hat{\mu} = (\hat{A}, \hat{B}, \hat{\Pi})$ :

$$\hat{\pi}_i = \frac{\sum_{j=1}^N p_1(i, j)}{\sum_{l=1}^N \sum_{j=1}^N p_1(l, j)} = \sum_{j=1}^N p_1(i, j) = \gamma_i(1)$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^T p_t(i, j)}{\sum_{l=1}^N \sum_{t=1}^T p_t(i, l)}$$

$$\hat{b}_{ijk} = \frac{\sum_{t: O_t=k, 1 \leq t \leq T} p_t(i, j)}{\sum_{t=1}^T p_t(i, j)}$$

## The Forward-Backward algorithm: Justification

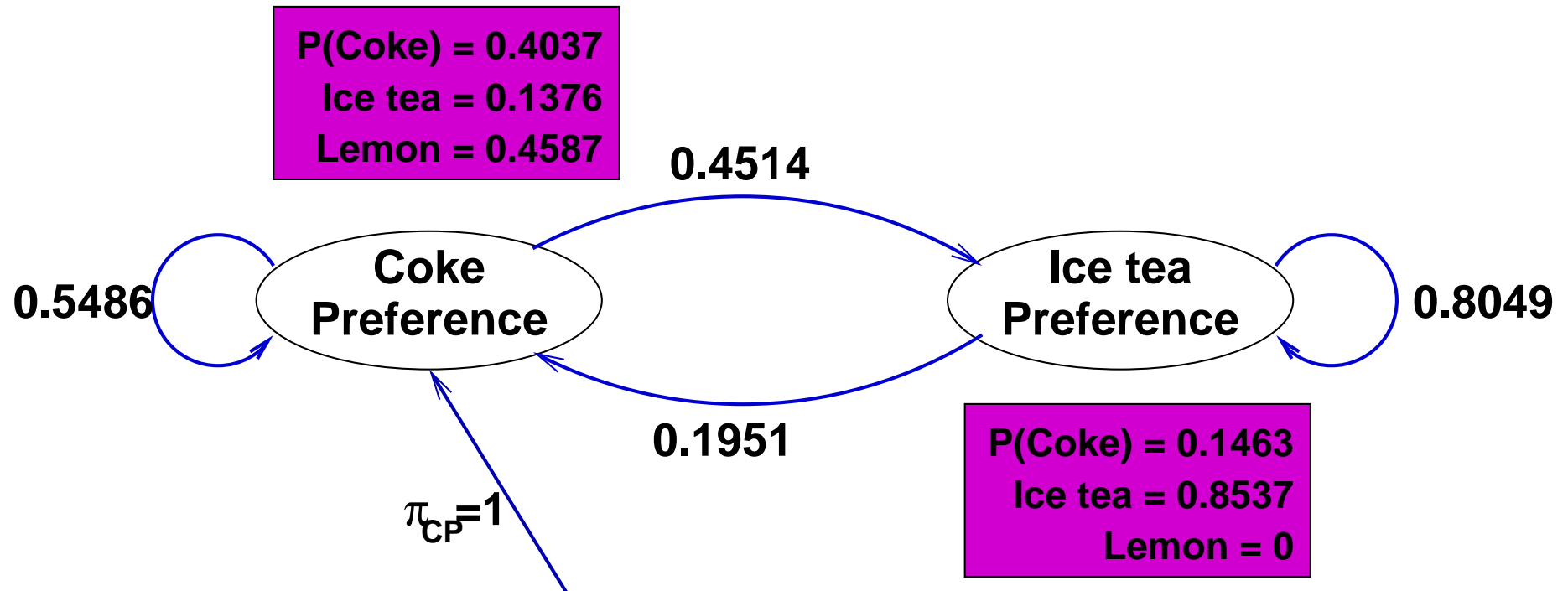
**Theorem (Baum):**  $P(O|\hat{\mu}) \geq P(O|\mu)$

**Note1:** However, it does not necessarily converge to a global optimum.

**Note2:** There is a straightforward *extension* of the algorithm that deals with multiple observation sequences (i.e., a corpus).

## Example: Re-estimation of HMM parameters

The crazy soft drink machine, after one EM iteration on the Sequence  $O = (\text{Lemon}, \text{Ice-tea}, \text{Coke})$



### 3.3.2 HMM parameter estimation: Viterbi version

**Objective:** *maximize*  $P(O \mid \Pi^*(O), \mu)$ , where  $\Pi^*(O)$  is the Viterbi path for the sequence  $O$

**Idea:**

Instead of estimating the parameters  $a_{ij}$ ,  $b_{ijk}$  using the expected values of hidden variables ( $p_t(i, j)$ ), estimate them (as Maximum Likelihood), based on the computed Viterbi path.

**Note:**

In practice, this method performs poorer than the Forward-Backward (Baum-Welch) main version. However it is widely used, especially when the HMM used is primarily intended to produce Viterbi paths.

## 4 HMM extensions

- Null (epsilon) emissions
- Initialization of parameters: improve chances of reaching global optimum
- Parameter tying: help coping with data sparseness
- Linear interpolation of HMMs
- Variable-Memory HMMs
- Acquiring HMM topologies from data

## 5 Some applications of HMMs

- Speech Recognition
- Text Processing: Part Of Speech Tagging
- Probabilistic Information Retrieval
- Bioinformatics: genetic sequence analysis

## 5.1 Part Of Speech (POS) Tagging

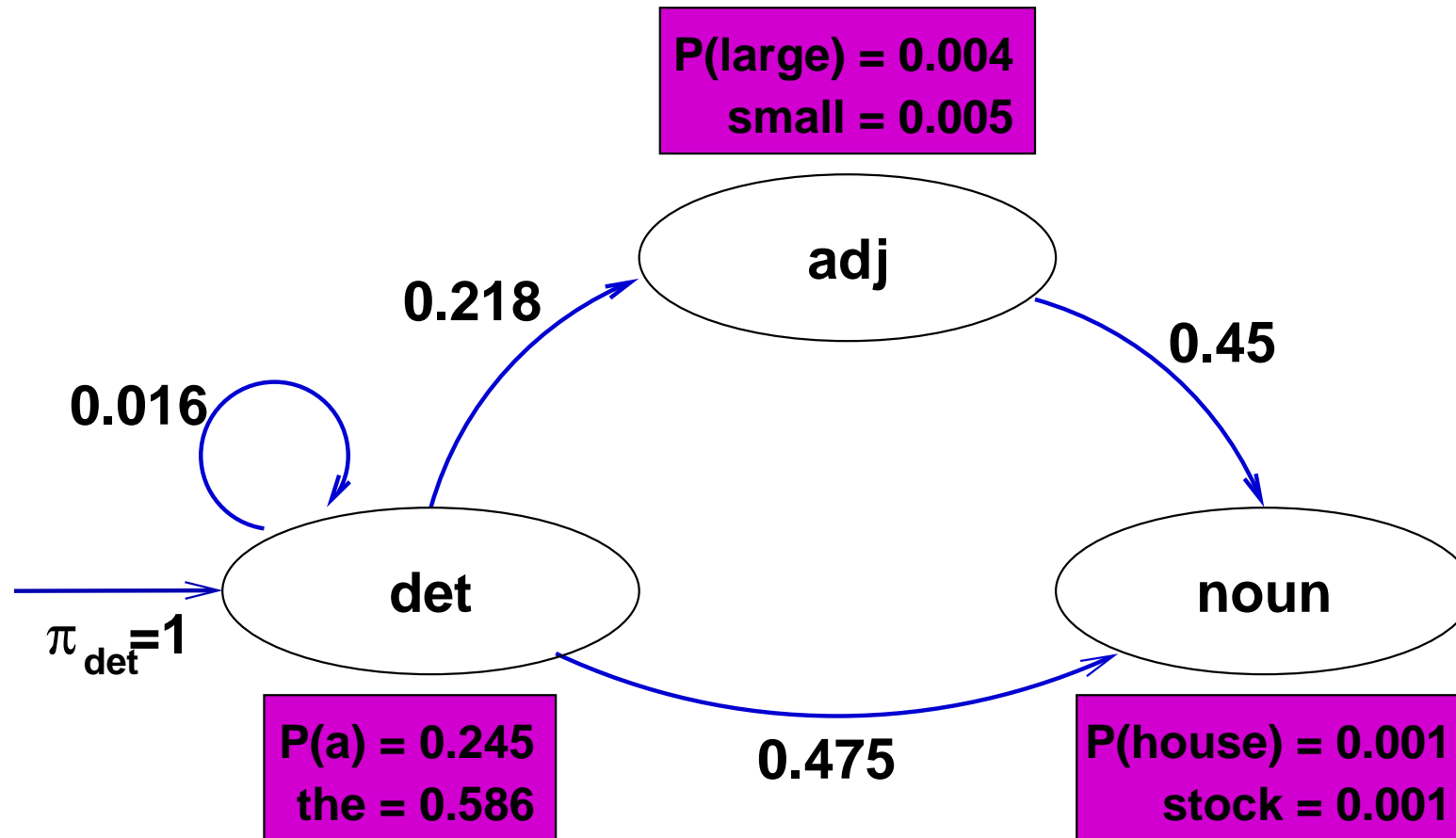
### Sample POS tags for the Brown/Penn Corpora

AT	article	RB	adverb
BEZ	<i>is</i>	RBR	adverb: comparative
IN	preposition	TO	<i>to</i>
JJ	adjective	VB	verb: base form
JJR	adjective: comparative	VBD	verb: past tense
MD	modal	VBG	verb: present participle, gerund
NN	noun: singular or mass	VBN	verb: past participle
NNP	noun: singular proper	VBP	verb: non-3rd singular present
PERIOD	.:?!	VBZ	verb: 3rd singular present
PN	personal pronoun	WDT	<i>wh</i> -determiner ( <i>what, which</i> )

## POS Tagging: Methods

- [Charniak, 1993] Frequency-based: 90% accuracy  
now considered baseline performance
- [Schmid, 1994] Decision lists; artificial neural networks
- [Brill, 1995] Transformation-based learning
- [Brants, 1998] Hidden Markov Models
- [Chelba &  
Jelinek, 1998] lexicalized probabilistic parsing (the best!)

**A fragment of a HMM for POS tagging**  
 (from [Charniak, 1997])



## Using HMMs for POS tagging

$$\operatorname{argmax}_{t_{1\dots n}} P(t_{1\dots n} | w_{1\dots n}) = \operatorname{argmax}_{t_{1\dots n}} \frac{P(w_{1\dots n} | t_{1\dots n}) P(t_{1\dots n})}{P(w_{1\dots n})}$$

$$= \operatorname{argmax}_{t_{1\dots n}} P(w_{1\dots n} | t_{1\dots n}) P(t_{1\dots n})$$

*using the two Markov assumptions*

$$= \operatorname{argmax}_{t_{1\dots n}} \prod_{i=1}^n P(w_i | t_i) \prod_{i=1}^n P(t_i | t_{i-1})$$

### Supervised POS Tagging:

$$\text{MLE estimations: } P(w|t) = \frac{C(w,t)}{C(t)}, \quad P(t''|t') = \frac{C(t',t'')}{C(t')}$$

## The Treatment of Unknown Words:

- use apriori uniform distribution over all tags:  
error rate 40%  $\Rightarrow$  20%
- feature-based estimation [ Weishedel et al., 1993 ]:  
 $P((w|t) = \frac{1}{Z}P(\text{unknown word} | t)P(\text{Capitalized} | t)P(\text{Ending} | t)$
- using both roots and suffixes [Charniak, 1993]

## Smoothing:

$$P(t|w) = \frac{C(t,w)+1}{C(w)+k_w} \quad [\text{Church, 1988}]$$

where  $k_w$  is the number of possible tags for  $w$

$$P(t''|t') = (1 - \epsilon) \frac{C(t',t'')}{C(t')} + \epsilon \quad [\text{Charniak et al., 1993}]$$

## Fine-tuning HMMs for POS tagging

See [ Brants, 1998 ]

## 5.2 The Google PageRank Algorithm

A Markov Chain worth no. 5 on *Forbes* list!

( $2 \times 18.5$  billion USD, as of November 2007)

“Sergey Brin and Lawrence Page introduced Google in 1998, a time when the pace at which the web was growing began to outstrip the ability of current search engines to yield usable results.

In developing Google, they wanted to improve the design of search engines by moving it into a more open, academic environment.

In addition, they felt that the usage of statistics for their search engine would provide an interesting data set for research.”

From David Austin, “How Google finds your needle in the web’s haystack”, Monthly Essays on Mathematical Topics, 2006.

## Notations

Let  $n =$  the number of pages on Internet, and  $H$  and  $A$  two  $n \times n$  matrices defined by

$$h_{ij} = \begin{cases} 1 & \text{if page } j \text{ points to page } i \text{ (notation: } P_j \in B_i) \\ 0 & \text{otherwise} \end{cases}$$

$$a_{ij} = \begin{cases} 1 & \text{if page } i \text{ contains no outgoing links} \\ 0 & \text{otherwise} \end{cases}$$

$$\alpha \in [0; 1] \text{ (this is a parameter that was initially set to 0.85)}$$

The **transition matrix** of the Google Markov Chain is

$$G = \alpha(H + A) + \frac{1 - \alpha}{n} \cdot \mathbf{1}$$

where  $\mathbf{1}$  is the  $n \times n$  matrix whose entries are all 1

The **significance of  $G$**  is derived from:

- the **Random Surfer** model
- the definition the (relative) **importance** of a page: combining votes from the pages that point to it

$$I(P_i) = \sum_{P_j \in B_i} \frac{I(P_j)}{l_j}$$

where  $l_j$  is the number of links pointing out from  $P_j$ .

# The PageRank algorithm

[Brin & Page, 1998]

$G$  is a **stochastic** matrix ( $g_{ij} \in [0; 1]$ ,  $\sum_{i=1}^n g_{ij} = 1$ ),  
therefore  $\lambda_1$  the greatest eigenvalue of  $G$  is 1, and  
 $G$  has a stationary vector  $I$  (i.e.,  $GI = I$ ).

$G$  is also **primitive** ( $|\lambda_2| < 1$ , where  $\lambda_2$  is the second eigenvalue of  $G$ )  
and **irreducible** ( $I > 0$ ).

From the matrix calculus it follows that

$I$  can be computed using the **power method**:  
if  $I^1 = GI^0$ ,  $I^2 = GI^1$ , ...,  $I^k = GI^{k-1}$  then  $I^k \rightarrow I$ .

$I$  gives the relative importance of pages.

## Suggested readings

“Using Google’s PageRank algorithm to identify important attributes of genes”

G.M. Osmani, S.M. Rahman, 2006

“MicroRNA prediction with a novel ranking algorithm based on random walks”

Y. Xu, X. Zhou, W. Zhang, 2008