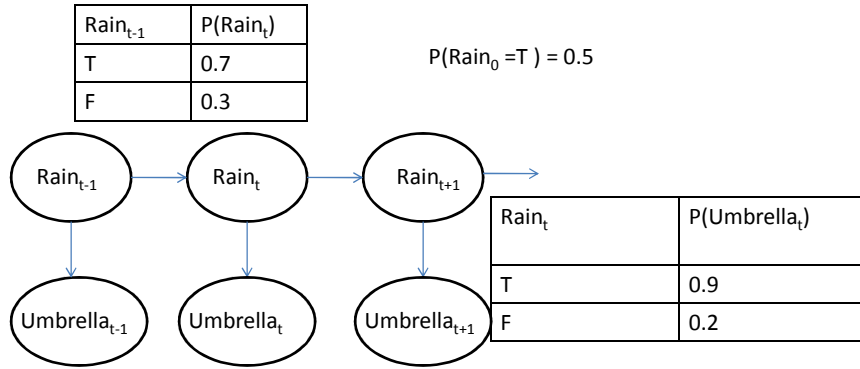
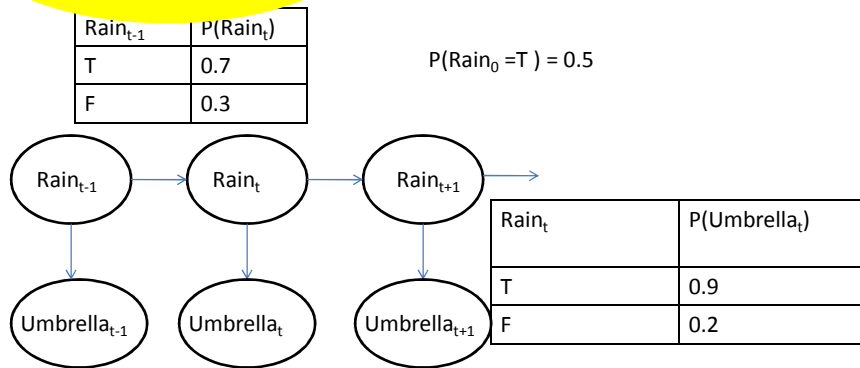


### Components of an HMM $\lambda$

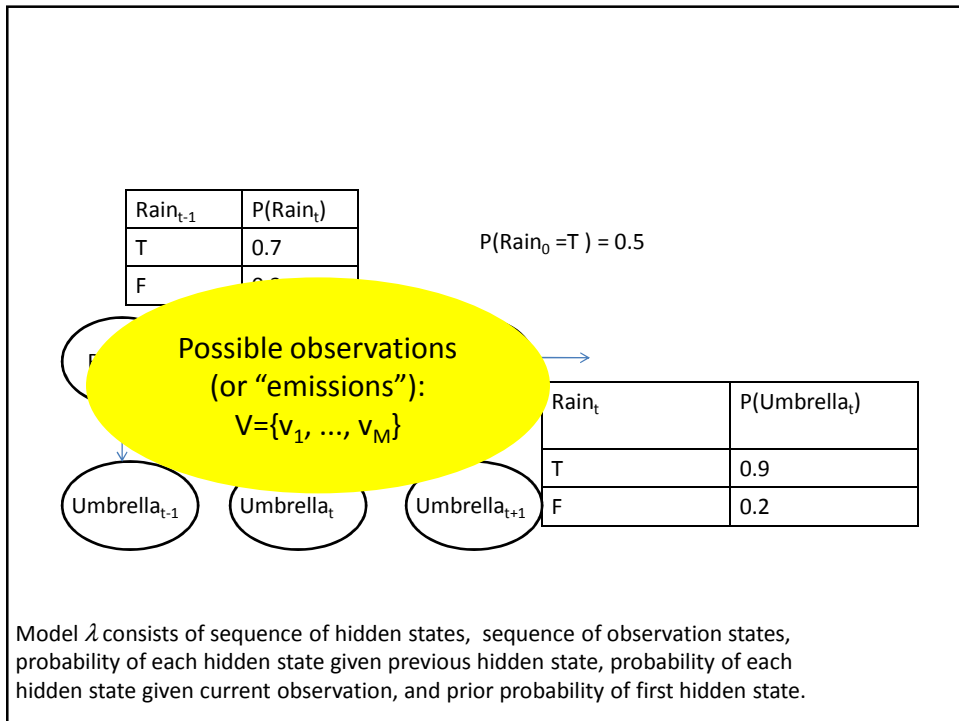
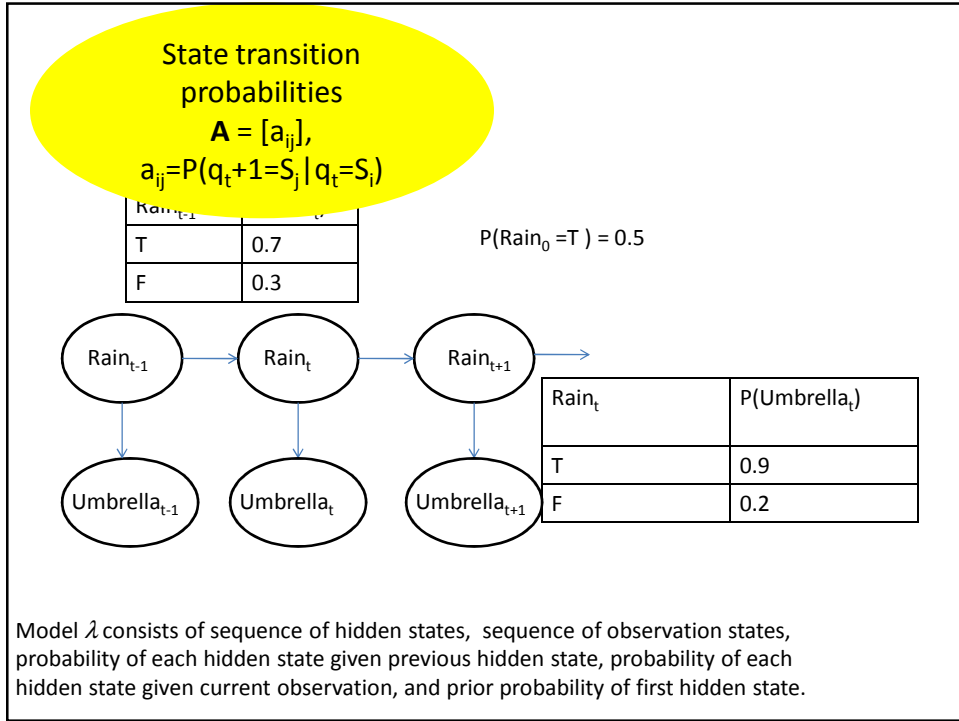


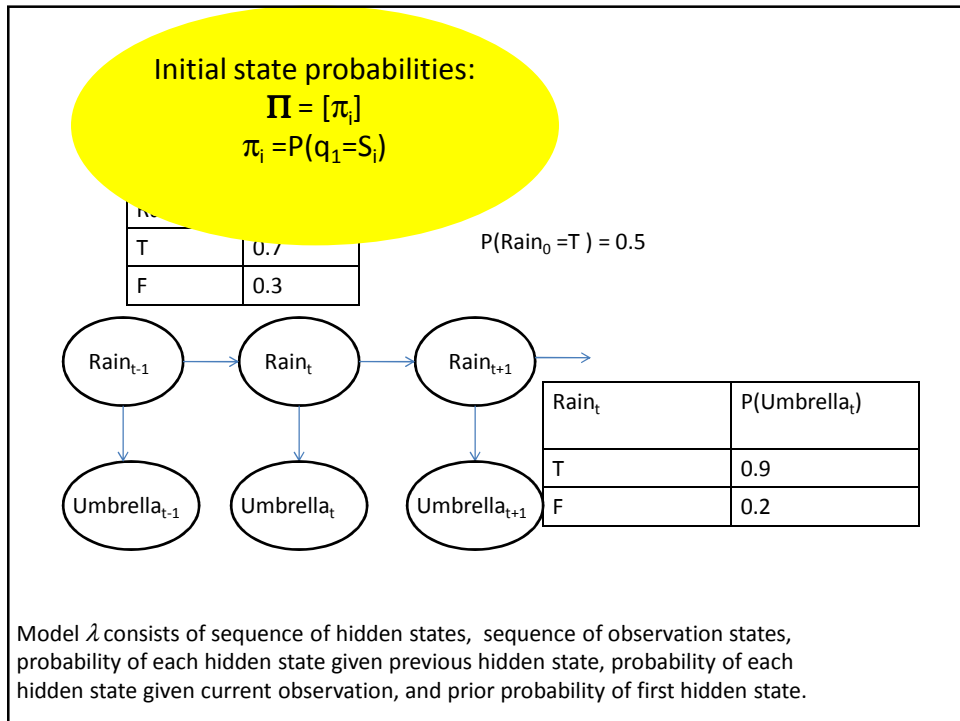
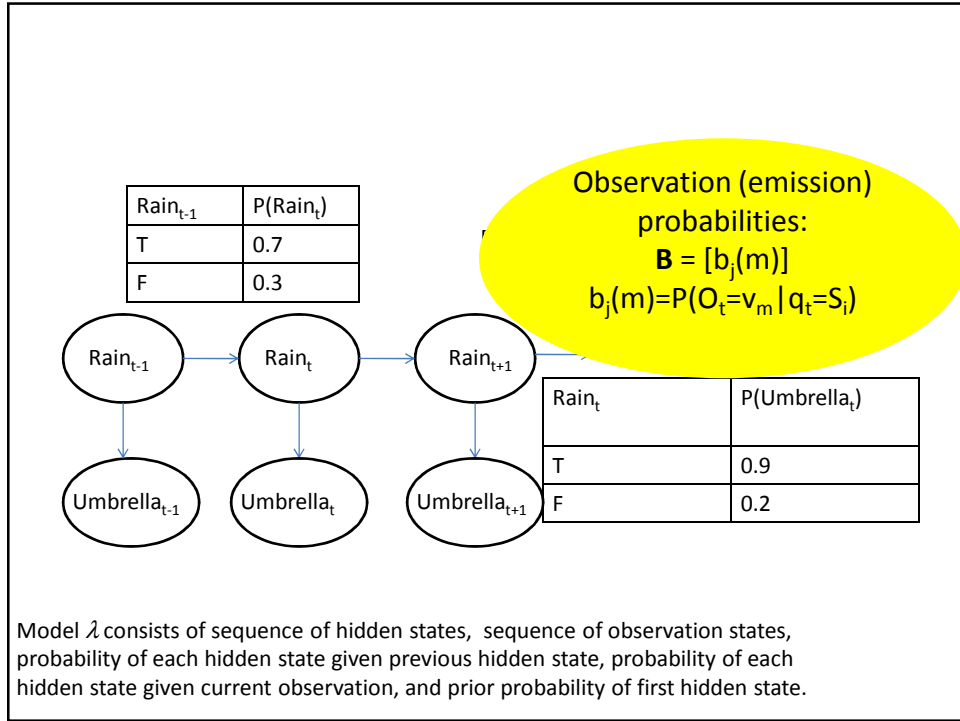
Model  $\lambda$  consists of sequence of hidden states, sequence of observation states, probability of each hidden state given previous hidden state, probability of each hidden state given current observation, and prior probability of first hidden state.

Possible states:  
 $S = \{S_1, \dots, S_N\}$



Model  $\lambda$  consists of sequence of hidden states, sequence of observation states, probability of each hidden state given previous hidden state, probability of each hidden state given current observation, and prior probability of first hidden state.





### Three problems/tasks for HMMs

1. Given model  $\lambda$ , what is probability of a given observation sequence  $O = \{O_1 O_2 \dots O_T\}$ ? Namely, what is  $P(O | \lambda)$ ?
2. Given model  $\lambda$  and observation sequence  $O$ , what state sequence  $Q = \{q_1 q_2 \dots q_T\}$  has highest probability of generating  $O$ ? Namely, what is  $\arg \max_Q P(Q | O, \lambda)$ ?
3. Given a training set of observations  $X = O$ , what is the model that maximizes probability of generating  $X$ ? Namely, what is  $\arg \max_{\lambda} P(X | \lambda)$ ?

### Problem 1: Evaluating a particular HMM

- Assume model  $\lambda$  with possible hidden state values  $S_i, 1 \leq i \leq N$ .
- What is probability of model given the data?
- In other words, given observation sequence  $O=O_1, \dots, O_T$  and the model  $\lambda$ , what is  $P(O | \lambda)$ ?
- Brute force method for calculating this: Calculate for every possible sequence of hidden states:  $Q = q_1, q_2, \dots, q_T$ .

$P(O | Q, \lambda) = P(O_1 | q_1)P(O_2 | q_2) \dots P(O_T | q_T)$  (observation at time  $t$  depends only on hidden state at time  $t$ )

$P(Q | \lambda) = P(q_1)P(q_2 | q_1) \dots P(q_{T-1} | q_{T-2})$  (hidden state at time  $t$  depends only on hidden state at time  $t-1$ )

$P(O, Q | \lambda) = P(O | Q, \lambda)P(Q | \lambda)$

Thus,  $P(O | \lambda) = \sum_{\text{all } Q} P(O | Q, \lambda)P(Q | \lambda)$

$= \sum_{\text{all } Q} P(O | Q, \lambda)P(Q | \lambda)$

$= \sum_{\text{all } Q} P(q_0) P(O_1 | q_1)P(q_1 | q_0) \dots P(O_T | q_T)P(q_T | q_{T-1})$

### **Umbrella/Rain example for $T=2$**

In general, impractical to calculate this in a brute-force way.

(There are  $N^T$  possible values of  $Q$ )

More efficient way to calculate this:

*forward-backward algorithm*  
(part of “*Baum-Walsh algorithm*”)

- General idea: divide observation sequence into two parts: from time 1 to  $t$  and from time  $t+1$  to  $T$ .
- Define *forward variable*  $\alpha_t(i)$  as probability of observing partial sequence  $\{O_1, O_2, \dots, O_t\}$  and being in state  $s_i$  at time  $t$ , given  $\lambda$ :

$$\alpha_t(i) = P(O_1, O_2, \dots, O_t, q_t = S_i | \lambda)$$

## Forward part of Forward-Backward Algorithm

Can solve for  $\alpha_t(i)$  inductively:

### 1. Solve for $t=1$ :

$$\begin{aligned}\alpha_1(i) &= P(O_1, q_1 = S_i | \lambda) \\ &= P(O_1 | q_1 = S_i, \lambda) P(q_1 = S_i | \lambda)\end{aligned}$$

### 2. Induction step:

$$\alpha_{t+1}(j) = \sum_{i=1}^N \alpha_t(i) P(q_{t+1} = S_j | q_t = S_i) \quad 1 \leq j \leq N$$

$\alpha_t(i)$  explains first  $t$  observations and ends in state  $s_i$ .

### Evaluation:

$$P(\mathbf{O} | \lambda) = \sum_{i=1}^N P(O, q_T = S_i | \lambda) = \sum_{i=1}^N \alpha_T(i)$$

- Complexity of calculating  $\alpha_t(i)$ :  $O(N^2T)$

**Umbrella/Rain example for  $T=2$**

- Can also define *backward variable*  $\beta_t(i)$ :

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \dots, O_T, q_t = S_i | \lambda)$$

- Can show that

$$\beta_t(i) = \sum_{j=1}^N \alpha_t(i) P(O_{t+1} | q_{t+1} = S_j) \beta_{t+1}(j)$$

## Problem 2

- Find  $\arg \max_Q P(Q | O, \lambda)$
- Use Viterbi algorithm

### Problem 3

- Find  $\arg \max_{\lambda} P(X | \lambda)$ ?
- Use Baum-Walsh algorithm (Uses forward-backward algorithm and EM)

### Outline of Baum-Walsh Algorithm

1. Define  $\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda)$
2. Derive formula for  $\xi$  in terms of  $\alpha_t(i)$  and  $\beta_{t+1}(j)$
3. Express probability of being in state  $S_i$  at time  $t$  in terms of  $\xi$ :
 
$$\gamma_t(i) = P(q_t = S_i | O, \lambda) = \sum_{j=1}^N \xi_t(i, j)$$
4. Guess parameters of model  $\lambda$
5. EM: repeat:
  - Compute  $\xi_t(i, j)$  and  $\gamma_t(i)$  given current parameters for  $\lambda$
  - Recalculate parameters for  $\lambda$  given current values of  $\xi_t(i, j)$  and  $\gamma_t(i)$

- Demo

## Beyond N-grams: Latent Semantic Analysis

- Problem: How to capture semantic similarity between documents in a natural corpus (e.g., problems of homonymy, polysemy, synonymy, etc.)
- In general, N-grams often fail to capture semantic similarity, even with query expansion, etc.
- “LSA assumes that there exists a LATENT structure in word usage – obscured by variability in word choice”  
(<http://ir.dcs.gla.ac.uk/oldseminars/Girolami.ppt>)

## Latent Semantic Analysis (Landauer et al.)

- From training data (large sample of documents), create word-by-document matrix.

### Technical Memo Example

**Titles:**

- c1: *Human machine interface* for Lab ABC *computer applications*
- c2: A *survey* of *user opinion* of *computer system response time*
- c3: The *EPS user interface* management *system*
- c4: *System* and *human system* engineering testing of *EPS*
- c5: Relation of *user-perceived response time* to error measurement

- m1: The generation of random, binary, unordered *trees*
- m2: The intersection *graph* of paths in *trees*
- m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
- m4: *Graph minors*: A *survey*

A sample dataset consisting of the titles of 9 technical memoranda. Terms occurring in more than one title are italicized. There are two classes of documents - five about human-computer interaction (c1-c5) and four about graphs (m1-m4). This dataset can be described by means of a term by document matrix where each cell entry indicates the frequency with which a term occurs in a document.

From Deerwester et al., *Indexing by latent semantic analysis*

Terms	Documents									
	c1	c2	c3	c4	c5	m1	m2	m3	m4	
<i>human</i>	1	0	0	1	0	0	0	0	0	
<i>interface</i>	1	0	1	0	0	0	0	0	0	
<i>computer</i>	1	1	0	0	0	0	0	0	0	
<i>user</i>	0	1	1	0	1	0	0	0	0	
<i>system</i>	0	1	1	2	0	0	0	0	0	
<i>response</i>	0	1	0	0	1	0	0	0	0	
<i>time</i>	0	1	0	0	1	0	0	0	0	
<i>EPS</i>	0	0	1	1	0	0	0	0	0	
<i>survey</i>	0	1	0	0	0	0	0	0	1	
<i>trees</i>	0	0	0	0	0	1	1	1	0	
<i>graph</i>	0	0	0	0	0	0	1	1	1	
<i>minors</i>	0	0	0	0	0	0	0	1	1	

- Now apply “singular value decomposition” to this matrix
- SVD is similar to principal components analysis (if you know what that is)
- Basically, reduce dimensionality of the matrix by re-representing matrix in terms of “features” (derived from eigenvalues and eigenvectors), and using only the ones with highest value.
- Result: Each document is represented by a vector of *features* obtained by SVD.
- Given a new document (or query), compute its representation vector in this *feature space*, compute its similarity with other documents using cosine between vector angles. Retrieve documents with highest similarities.

## Result of Applying LSA

From <http://lair.indiana.edu/courses/i502/lectures/lect6.ppt>

	DOC1	DOC2	DOC3	DOC4	DOC5	DOC6	DOC7	DOC8	DOC9
HUMAN	0.16	0.4	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
INTERFACE	0.14	0.37	0.33	0.4	0.16	-0.03	-0.07	-0.1	-0.04
COMPUTER	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
USER	0.26	0.84	0.61	0.7	0.39	0.03	0.08	0.12	0.19
SYSTEM	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
RESPONSE	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
TIME	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.2	-0.11
SURVEY	0.1	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
TREES	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
GRAPH	-0.06	0.34	-0.15	-0.3	0.2	0.31	0.69	0.98	0.85
MINORS	-0.04	0.25	-0.1	-0.21	0.15	0.22	0.5	0.71	0.62

In the above matrix we can now observe:

$$r(\text{human.user}) = 0.94$$

$$r(\text{human.minors}) = -0.83$$

## How does it find the latent associations?

From <http://lair.indiana.edu/courses/i502/lectures/lect6.ppt>

- By analyzing the contexts in which the words appear
- The word *user* has co-occurred with words that *human* has co-occurred with (e.g., system and interface)
- It downgrades associations when such contextual similarities are not found

## Some General LSA Based Applications

From <http://lsa.colorado.edu/~quesadaj/pdf/LSATutorial.pdf>

### **Information Retrieval**

#### **Text Assessment**

Compare document to documents of known quality / content

#### **Automatic summarization of text**

Determine best subset of text to portray same meaning

#### **Categorization / Classification**

Place text into appropriate categories or taxonomies

## Application: Automatic Essay Scoring (in collaboration with Educational Testing Service)

Create domain semantic space

Compute vectors for essays, add to vector database

To predict grade on a new essay, compare it to ones previously scored by humans

From <http://lsa.colorado.edu/~quesadaj/pdf/LSATutorial.pdf>

**Mutual information between two sets of grades:**

human – human .90

LSA – human .81

From <http://lsa.colorado.edu/~quesadaj/pdf/LSATutorial.pdf>

[Demo](#)

(<http://www.pearsonkt.com/>)