

In The Name of Allah



Digital Media Laboratory  
Sharif University of Technology

# Statistical Pattern Recognition

## Graphical Methods

**Hamid R. Rabiee**  
**Jafar Muhammadi**

Spring 2012

<http://ce.sharif.edu/courses/90-91/2/ce725-1/>

# Agenda

## ✧ Graphical Methods

## ✧ Bayesian Networks

- ✧ Constructing Bayesian Network
- ✧ Inference (General Querying)
- ✧ Stochastic Simulation
- ✧ Learning of Bayesian Networks

## ✧ Hidden Markov Models (HMM)

- ✧ Markov Random Processes
- ✧ Motivation and Background
- ✧ Specification of an HMM
- ✧ Central problems in HMM modelling
- ✧ Word Recognition Example
- ✧ Character recognition with HMM example



# Graphical Methods

- ✧ **The use of probabilistic (specifically Bayesian) methods in pattern recognition amounts to a consistent application of the sum and product rules of probability**
- ✧ **These lead to complicated algebraic formulae. Hence, it is highly advantageous to augment the analysis using diagrammatic representations of probability distributions, called *probabilistic graphical models*.**



# Graphical Methods

## ✧ **Key Idea:**

- ✧ **Conditional independence assumptions useful.**
- ✧ **Graphical models express sets of conditional independence assumptions via graph structure**
- ✧ **Graph structure plus associated parameters define joint probability distribution over set of variables/nodes**



# Graphical Methods

## ✧ Advantages

- ✧ They provide a simple way to visualize the structure of a probabilistic model and can be used to design and motivate new models.
- ✧ Insights into the properties of the model can be obtained by inspection of the graph.
- ✧ Complex computations, required to perform inference and learning in sophisticated models, can be expressed in terms of graphical manipulations, in which underlying mathematical expressions are carried along implicitly.

## ✧ Two types of graphical models:

- ✧ Undirected graphs (e.g. Markov Random Fields)
- ✧ Directed graphs (e.g. Bayesian Networks and HMM)

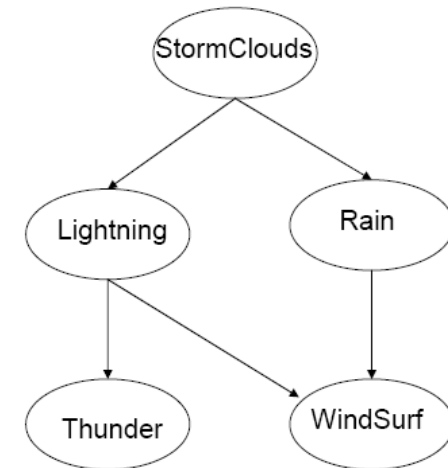


# Bayesian Networks

## ✧ Bayesian network

- ✧ Also called a belief network
- ✧ a directed acyclic graph defining a joint probability distribution over a set of variables
- ✧ Node denote random variables and edges denote dependencies
- ✧ Each node is conditionally independent of its non-descendants, given its immediate parents.
- ✧ A conditional probability distribution (CPD) is associated with each node N, defining  $P(N \mid \text{Parents}(N))$
- ✧ In Bayesian net we have:

$$P(X_1, \dots, X_n) = \prod_i P(X_i \mid \text{Parent}(X_i))$$



Parents	$P(W Pa)$	$P(\neg W Pa)$
L, R	0	1.0
L, $\neg$ R	0	1.0
$\neg$ L, R	0.2	0.8
$\neg$ L, $\neg$ R	0.9	0.1



# Constructing Bayesian Network

## ✧ Algorithm

- ✧ Choose an ordering over variables, e.g.,  $X_1, X_2, \dots, X_n$
- ✧ For  $i=1$  to  $n$ 
  - ✧ Add  $X_i$  to the network
  - ✧ Select parents  $\text{Parent}(X_i)$  as minimal subset of  $X_1, \dots, X_{i-1}$  such that

$$P(X_i | \text{Parent}(X_i)) = P(X_i | X_1, \dots, X_{i-1})$$

- ✧ Notice this choice of parents assures

$$P(X_1, \dots, X_n) = \prod_i P(X_i | X_1, X_2, \dots, X_{i-1}) = \prod_i P(X_i | \text{Parent}(X_i))$$



# Example

## ✧ Assume five variables

- ✧ **T: The lecture started by 10:35**
- ✧ **L: The lecturer arrives late**
- ✧ **R: The lecture concerns robots**
- ✧ **M: The lecturer is Manuela**
- ✧ **S: It is sunny**

## ✧ Assumptions

- ✧ **T only directly influenced by L (i.e. T is conditionally independent of R,M,S given L)**
- ✧ **L only directly influenced by M and S (i.e. L is conditionally independent of R given M & S)**
- ✧ **R only directly influenced by M (i.e. R is conditionally independent of L,S, given M)**
- ✧ **M and S are independent**





# Example

## ✧ Step 1: Add variables

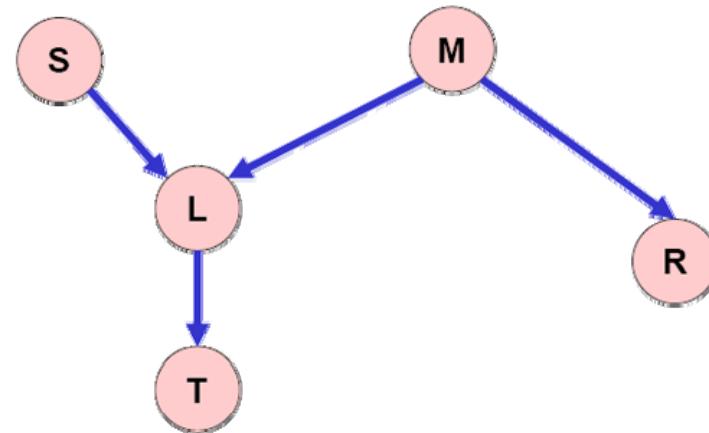
- ✧ Just choose the variables you'd like to be included in the net

## ✧ Step 2: add links

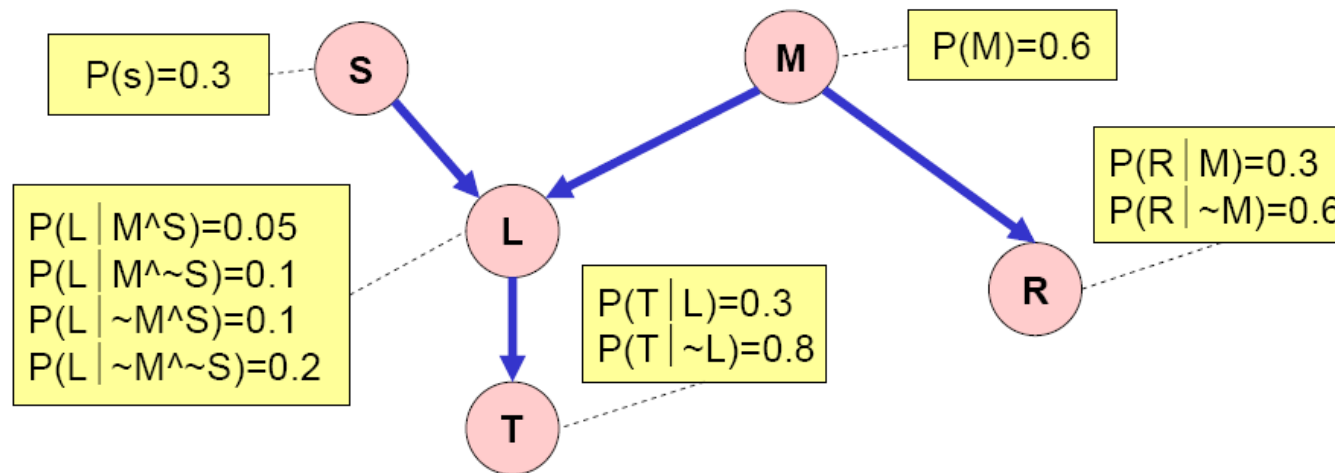
- ✧ The link structure must be acyclic.
- ✧ If node  $X$  is given parents  $Q_1, Q_2, \dots, Q_n$  you are promising that any variable that's a non-descendent of  $X$  is conditionally independent of  $X$  given  $\{Q_1, Q_2, \dots, Q_n\}$

## ✧ Step 3: add a probability table for each node.

- ✧ The table for node  $X$  must list  $P(X|\text{Parent}(x))$  for each possible combination of parent values



# Example

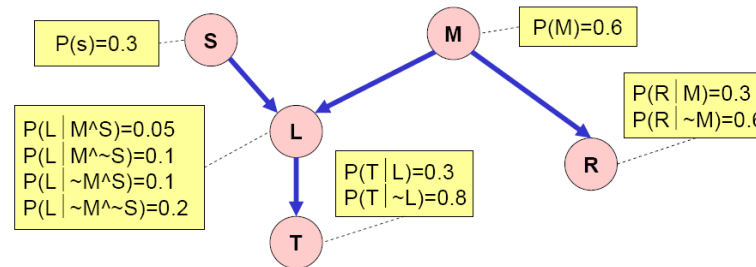


✧ **Note that:**

- ✧ Two unconnected variables may still be correlated (why?)
- ✧ Each node is conditionally independent of all non-descendants in the tree, given its parents.
- ✧ You can deduce many other conditional independence relations from a Bayesian net.



# Example



## ✧ Computing with Bayesian Network (inference)

✧  $P(T \wedge \sim R \wedge L \wedge \sim M \wedge S) = ?$

$$P(T \wedge \sim R \wedge L \wedge \sim M \wedge S) = P(T | \sim R \wedge L \wedge \sim M \wedge S) * P(\sim R \wedge L \wedge \sim M \wedge S) =$$

$$P(T | L) * P(\sim R \wedge L \wedge \sim M \wedge S) =$$

$$P(T | L) * P(\sim R | L \wedge \sim M \wedge S) * P(L \wedge \sim M \wedge S) =$$

$$P(T | L) * P(\sim R | \sim M) * P(L \wedge \sim M \wedge S) =$$

$$P(T | L) * P(\sim R | \sim M) * P(L | \sim M \wedge S) * P(\sim M \wedge S) =$$

$$P(T | L) * P(\sim R | \sim M) * P(L | \sim M \wedge S) * P(\sim M | S) * P(S) =$$

$$P(T | L) * P(\sim R | \sim M) * P(L | \sim M \wedge S) * P(\sim M) * P(S).$$



# Inference (General Querying)

✧ **We can compute answers to any questions**

✧ **E.G. What could we do to compute  $P(R | T, \sim S)$ ?**

$$P(R | T, \sim S) = \frac{P(R \wedge T \wedge \sim S)}{P(R \wedge T \wedge \sim S) + P(\sim R \wedge T \wedge \sim S)}$$

✧ **Answering queries need some tricks**

✧ **There is a general querying of Bayesian nets, but it is NP-complete, unfortunately.**

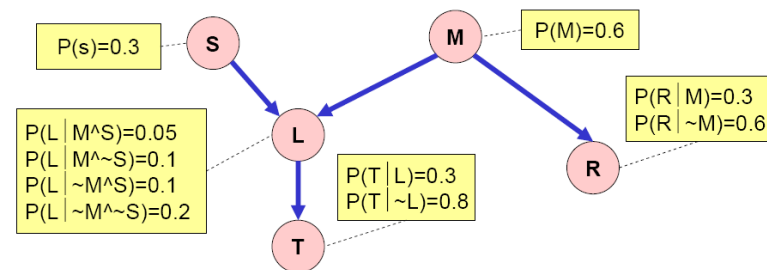
✧ **A popular, practical approach is Stochastic Simulation.**



# Stochastic Simulation

✧ It's pretty easy to generate a set of variable-assignments at random with the same probability as the underlying joint distribution.

✧ How?



1. Randomly choose S. S = True with prob 0.3
2. Randomly choose M. M = True with prob 0.6
3. Randomly choose L. The probability that L is true depends on the assignments of S and M. E.G. if steps 1 and 2 had produced S=True, M=False, then probability that L is true is 0.1
4. Randomly choose R. Probability depends on M.
5. Randomly choose T. Probability depends on L



# Stochastic Simulation Algorithm

✧ Let's generalize the example on the previous slide to a general Bayesian net

✧ Call the variables  $X_1, \dots, X_n$ , where  $\text{Parent}(X_i)$  must be a subset of  $\{X_1, \dots, X_{i-1}\}$ .

✧ For  $i = 1$  to  $n$ :

1. Find parents, if any, of  $X_i$ . Assume  $n(i)$  parents. Call them  $X_{p(i,1)}, X_{p(i,2)}, \dots, X_{p(i,n(i))}$ .

2. Recall the values that those parents were randomly given:  $x_{p(i,1)}, x_{p(i,2)}, \dots, x_{p(i,n(i))}$ .

3. Look up in the lookup-table for

$$P(X_i=\text{True} \mid X_{p(i,1)}=x_{p(i,1)}, X_{p(i,2)}=x_{p(i,2)}, \dots, X_{p(i,n(i))}=x_{p(i,n(i))})$$

4. Randomly set  $x_i=\text{True}$  according to this probability

✧  $x_1, x_2, \dots, x_n$  are now a sample from the joint distribution of  $X_1, X_2, \dots, X_n$ .



# Stochastic Simulation Example

- ✧ Someone wants to know  $P(R = \text{True} \mid T = \text{True} \wedge S = \text{False})$
- ✧ We'll do lots of random samplings and count the number of occurrences of the following:
  - ✧  $N_c$ : Number of samples in which  $T=\text{True}$  and  $S=\text{False}$ .
  - ✧  $N_s$ : Number of samples in which  $R=\text{True}$ ,  $T=\text{True}$  and  $S=\text{False}$ .
  - ✧  $N$ : Number of random samplings
- ✧ Now if  $N$  is big enough:
  - ✧  $N_c/N$  is a good estimate of  $P(T=\text{True} \text{ and } S=\text{False})$ .
  - ✧  $N_s/N$  is a good estimate of  $P(R=\text{True} , T=\text{True} , S=\text{False})$ .
  - ✧  $P(R|T \wedge \sim S) = P(R \wedge T \wedge \sim S) / P(T \wedge \sim S)$ , so  $N_s/N_c$  can be a good estimate of  $P(R|T \wedge \sim S)$ .



# Learning of Bayesian Networks

## ✧ Variants of learning task:

### ✧ Network structure

- ✧ Structure might be unknown

### ✧ Networks variables

- ✧ Training examples might provide values of only some variables
- ✧ If structure of network be known we can use for example gradient ascent algorithm for training the variables

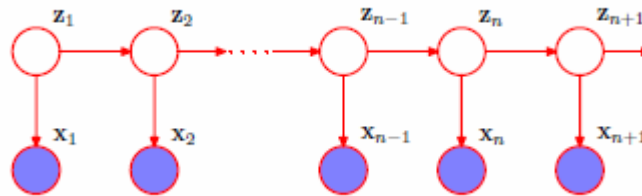
## ✧ We don't cover the learning tasks in this course!





# Hidden Markov Models (HMMs)

- ✧ A specific, and very familiar, example of a directed graph
  - ✧ It is widely used in speech recognition, handwriting recognition, DNA analysis, and other sequential data applications



# Markov Random Processes

## ✧ Markov random process

- ✧ A random sequence has the Markov property if its distribution is determined solely by its current state. Any random process having this property is called a *Markov random process*.

## ✧ Markov chain

- ✧ For observable state sequences (state is known from data), this leads to a *Markov chain* model.

## ✧ Hidden Markov Model

- ✧ For non-observable states, this leads to a *Hidden Markov Model (HMM)*.



# HMM Motivation

- ✧ **Real-world has structures and processes which have (or produce) observable outputs**
  - ✧ **Usually sequential (process unfolds over time)**
  - ✧ **Cannot see the event producing the output**
    - ✧ Example: speech signals
    - ✧ Example: Weather forecasting
- ✧ **Problem: how to construct a model of the structure or process given only observations?**



# HMM Background

- ✧ **Basic theory developed and published in 1960s and 70s**
- ✧ **No widespread understanding and application until late 80s**
- ✧ **Why?**
  - ✧ **Theory published in mathematic journals which were not widely read by practicing engineers**
  - ✧ **Insufficient tutorial material for readers to understand and apply concepts**



# HMM Uses

## ✧ Some uses of HMM are:

### ✧ **Speech recognition and processing**

- ✧ Recognizing spoken words and phrases
- ✧ Speech synthesis
- ✧ Many applications in this field

### ✧ **Text processing**

- ✧ Parsing raw records into structured records
- ✧ Part-of-Speech Tagging

### ✧ **Bioinformatics**

- ✧ Protein sequence prediction

### ✧ **Financial**

- ✧ Stock market forecasts (price pattern prediction)
- ✧ Comparison shopping services



# Specification of an HMM

## ✧ Components

### ✧ **N** - number of states

✧  $Q = \{q_1; q_2; \dots; q_T\}$  - set of states

### ✧ **M** - number of symbols (observables)

✧  $O = \{o_1; o_2; \dots; o_T\}$  - set of symbols

### ✧ **A** - the state transition probability matrix

✧  $a_{ij} = P(q_{t+1} = j | q_t = i)$

### ✧ **B**- observation probability distribution

✧  $b_j(k) = P(o_t = k | q_t = j) \quad i \leq k \leq M$

### ✧ $\pi$ - the initial state distribution

## ✧ Full HMM is thus specified as a triplet:

✧  $\lambda = (A, B, \pi)$



# Central problems in HMM modelling

## ✧ **Problem 1: Evaluation:**

- ✧ **Probability of occurrence of a particular observation sequence,  $O = \{o_1, \dots, o_k\}$ , given the model -  $P(O|\lambda)$**
- ✧ **Complicated – hidden states**
- ✧ **Useful in sequence classification**

## ✧ **Problem 2: Decoding:**

- ✧ **Optimal state sequence to produce given observations,  $O = \{o_1, \dots, o_k\}$ , given model**
- ✧ **Optimality criterion**
- ✧ **Useful in recognition problems**

## ✧ **Problem 3: Learning:**

- ✧ **Determine optimum model, given a training set of observations**
- ✧ **Find  $\lambda$ , such that  $P(O|\lambda)$  is maximal**



# Problem 1: Naive solution

✧ **State sequence**  $Q = (q_1, \dots, q_T)$

✧ **Assume independent observations:**

$$P(O | q, \lambda) = \prod_{i=1}^T P(o_i | q_i, \lambda) = b_{q_1}(o_1) b_{q_2}(o_2) \dots b_{q_T}(o_T)$$

✧ **Observations are mutually independent, given the hidden states. (Joint distribution of independent variables factorises into marginal distributions of the independent variables.)**

✧ **Observe that:**  $P(q | \lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \dots a_{q_{T-1} q_T}$

✧ **And that:**  $P(O | \lambda) = \sum_q P(O | q, \lambda) P(q | \lambda)$





# Problem 1: Efficient solution

## ✧ Forward algorithm:

### ✧ A Dynamic Programming approach

### ✧ Define auxiliary forward variable $\alpha$ : $\alpha_t(\mathbf{i}) = P(o_1, \dots, o_t \mid q_t = \mathbf{i}, \lambda)$

✧  $\alpha_t(\mathbf{i})$  is the probability of observing a partial sequence of observables  $o_1, \dots, o_t$  such that at time  $t$ , state  $q_t = \mathbf{i}$

### ✧ Recursive algorithm:

✧ Initialise:  $\alpha_1(\mathbf{i}) = \pi_{\mathbf{i}} b_{\mathbf{i}}(o_1)$

✧ Calculate:  $\alpha_{t+1}(\mathbf{j}) = \left[ \sum_{\mathbf{i}=1}^N \alpha_t(\mathbf{i}) a_{\mathbf{ij}} \right] b_{\mathbf{j}}(o_{t+1})$

✧ Obtain:  $P(O \mid \lambda) = \sum_{\mathbf{i}=1}^N \alpha_T(\mathbf{i})$



# Problem 1: Alternative solution

## ✧ Backward algorithm:

### ✧ Again Dynamic Programming

### ✧ Define auxiliary forward variable $\beta$ : $\beta_t(\mathbf{i}) = \mathbf{P}(\mathbf{o}_{t+1}, \mathbf{o}_{t+2}, \dots, \mathbf{o}_T \mid \mathbf{q}_t = \mathbf{i}, \lambda)$

✧  $\beta_t(\mathbf{i})$  : the probability of observing a sequence of observables  $\mathbf{o}_{t+1}, \dots, \mathbf{o}_T$  given state  $\mathbf{q}_t = \mathbf{i}$  at time  $t$ , and  $\lambda$

### ✧ Recursive algorithm:

✧ Initialise:  $\beta_T(\mathbf{j}) = \mathbf{1}$

✧ Calculate:  $\beta_t(\mathbf{i}) = \sum_{j=1}^N \beta_{t+1}(\mathbf{j}) a_{ij} b_j(o_{t+1})$

✧ Terminate:  $p(\mathbf{O} \mid \lambda) = \sum_{\mathbf{i}=1}^N \beta_1(\mathbf{i})$



## Problem 2: Decoding

- ✧ **Choose state sequence to maximise probability of observation sequence**
- ✧ **Viterbi algorithm - inductive algorithm that keeps the best state sequence at each instance**
  - ✧ **Utilizes dynamic programming**
  - ✧ **State sequence to maximise:  $P(q_1, q_2, \dots, q_T | O, \lambda)$**
  - ✧ **Define auxiliary variable  $\delta$ :  $\delta_t(i) = \max_q P(q_1, q_2, \dots, q_t = i, o_1, o_2, \dots, o_t | \lambda)$** 
    - ✧  $\delta_t(i)$  – the probability of the most probable path ending in state  $q_t=i$



## Problem 2: Decoding

✧ **Recurrent property:**  $\delta_{t+1}(j) = \max_i (\delta_t(i) a_{ij}) b_j(o_{t+1})$

✧ **Algorithm:**

✧ To get state seq, need to keep track of argument to maximise this, for each  $t$  and  $j$ . Done via the array  $\psi_t(j)$ .

✧ **1. Initialise:**  $\delta_1(i) = \pi_i b_i(o_1)$ ,  $1 \leq i \leq N$   
 $\psi_1(i) = 0$

✧ **2. Recursion:**  $\delta_t(j) = \max_{1 \leq i \leq N} (\delta_{t-1}(i) a_{ij}) b_j(o_t)$   
 $\psi_t(j) = \arg \max_{1 \leq i \leq N} (\delta_{t-1}(i) a_{ij})$   $2 \leq t \leq T, 1 \leq j \leq N$

✧ **3. Terminate:**  $P^* = \max_{1 \leq i \leq N} \delta_T(i)$   
 $q_T^* = \arg \max_{1 \leq i \leq N} \delta_T(i)$

✧ **P\*** gives the state-optimised probability

✧ **Q\*** is the optimal state sequence ( $Q^* = \{q_1^*, q_2^*, \dots, q_T^*\}$ )

✧ **4. Backtrack state sequence:**  $q_t^* = \psi_{t+1}(q_{t+1}^*)$ ,  $t = T-1, T-2, \dots, 1$



## Problem 3: Learning

- ✧ **Training HMM to encode observation sequence such that HMM should identify a similar observation sequence in future**
- ✧ **Find  $\lambda=(A,B,\pi)$ , maximising  $P(O|\lambda)$**
- ✧ **General algorithm:**
  - ✧ **Initialise:  $\lambda_0$**
  - ✧ **Compute new model  $\lambda$ , using  $\lambda_0$  and observed sequence  $O$**
  - ✧ **Then  $\lambda_0 \leftarrow \lambda$**
  - ✧ **Repeat steps 2 and 3 until:  $\log P(O|\lambda) - \log P(O|\lambda_0) < d$**
- ✧ **We don't cover the learning algorithms in this course.**

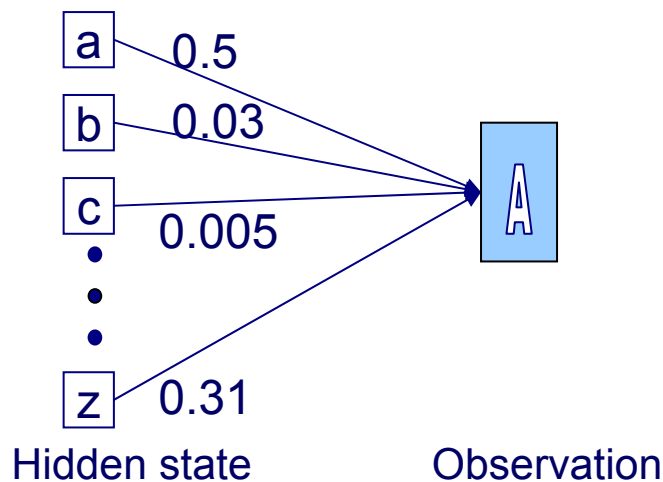


# Word Recognition Example

- ✧ **Typed word recognition, assume all characters are separated.**

Amherst

- ✧ **Character recognizer outputs probability of the image being particular character,  $P(\text{image} | \text{character})$ .**



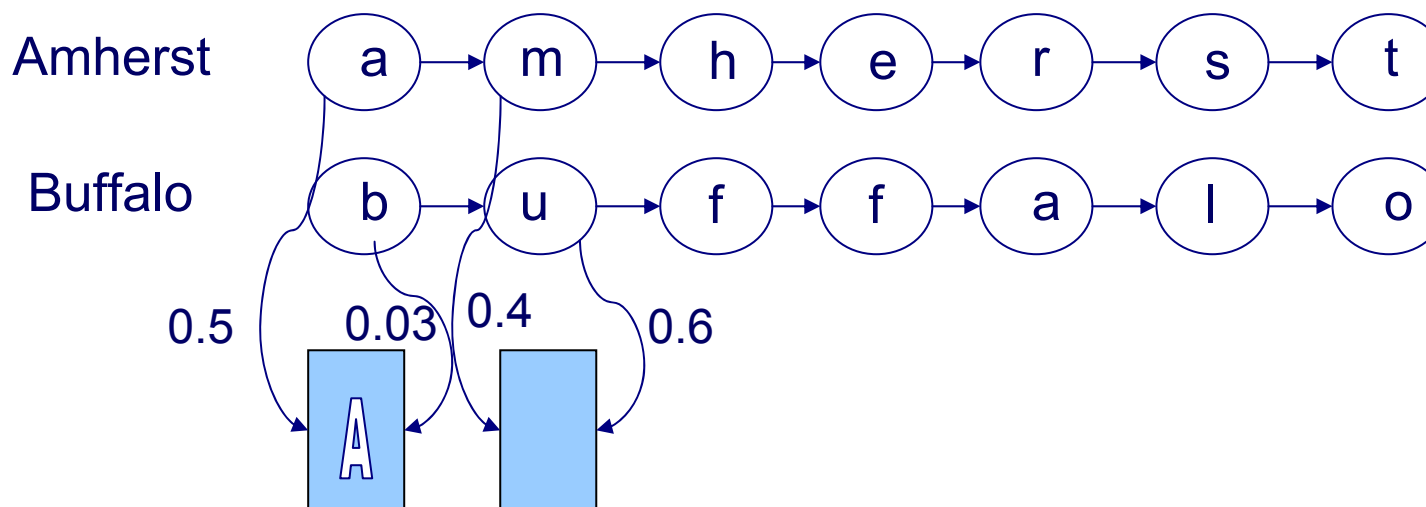
# Word Recognition Example

- ✧ **Hidden states of HMM = characters.**
- ✧ **Observations = typed images of characters segmented from the image  $v_\alpha$ . Note that there is an infinite number of observations.**
- ✧ **Observation probabilities = character recognizer scores.**
- ✧ **Transition probabilities will be defined differently in two subsequent models.**



# Word Recognition Example

- ✧ If lexicon is given, we can construct separate HMM models for each lexicon word.



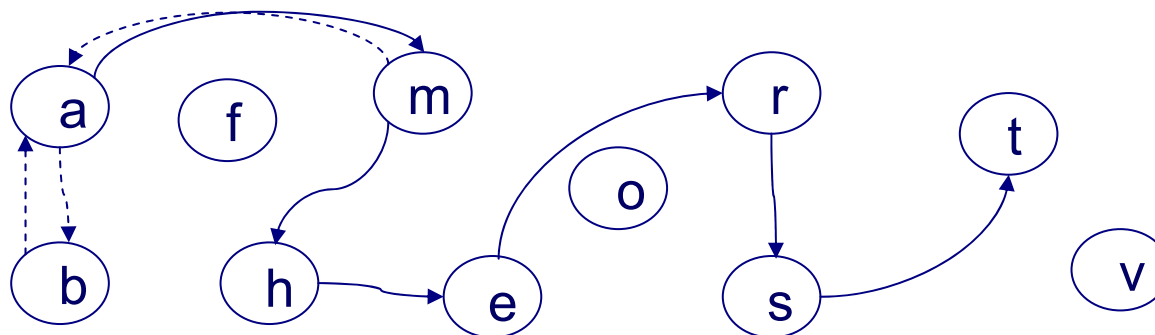
- ✧ Here recognition of word image is equivalent to the problem of evaluating few HMM models.
- ✧ This is an application of Evaluation problem.





# Word Recognition Example

- ✧ We can construct a single HMM for all words.
- ✧ Hidden states = all characters in the alphabet.
- ✧ Transition probabilities and initial probabilities are calculated from language model.
- ✧ Observations and observation probabilities are as before.



- ✧ Here we have to determine the best sequence of hidden states, the one that most likely produced word image.
- ✧ This is an application of Decoding problem.



## Further Reading

- ✧ **L. R. Rabiner, "A tutorial on Hidden Markov Models and selected applications in speech recognition," Proceedings of the IEEE, vol. 77, pp. 257-286, 1989.**
- ✧ **R. Dugad and U. B. Desai, "A tutorial on Hidden Markov models," Signal Processing and Artificial Neural Networks Laboratory, Dept of Electrical Engineering, Indian Institute of Technology, Bombay Technical Report No.: SPANN-96.1, 1996.**
- ✧ **Andrew W. Moore, HMM tutorial, [www.autonlab.org/tutorials](http://www.autonlab.org/tutorials).**



Any Question?

## End of Lecture 12

# Thank you!

Spring 2012

<http://ce.sharif.edu/courses/90-91/2/ce725-1/>

- ✧ Thanks to Andrew W. Moore for his good slides
  - ✧ [www.autonlab.org/tutorials](http://www.autonlab.org/tutorials)
- ✧ Thanks to Tom Mitchell for his good slides
  - ✧ [www.cs.cmu.edu/~tom/](http://www.cs.cmu.edu/~tom/)

