

Music Machine Learning

IX – Hidden Markov Models

Master ATIAM - Informatique

Philippe Esling (esling@ircam.fr)

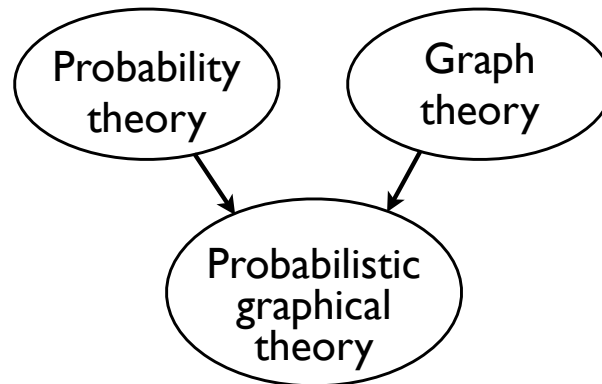
Maître de conférences – UPMC

Equipe représentations musicales (IRCAM, Paris)



Probabilistic Graphical Models

- Graphs endowed with a probability distribution
 - **Nodes** represent random variables and the **edges** encode conditional independence assumptions
- Graphical model express **sets of conditional independence** via graph structure (and conditional independence is useful)
- Graph structure plus associated parameters define joint probability distribution of the set of nodes/variables



Probrabilistic Graphical Models

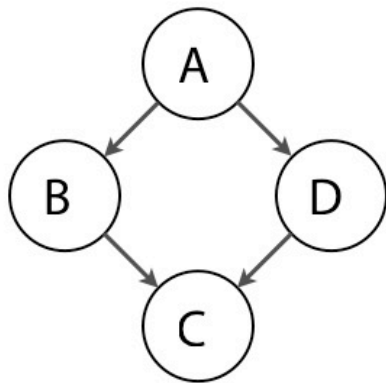
- Graphical models come in two main flavors:
 1. Directed graphical models (a.k.a Bayes Net, Belief Networks):
 - Consists of a set of nodes with arrows (directed edges) between some of the nodes
 - Arrows encode factorized conditional probability distributions
 2. Undirected graphical models (a.k.a Markov random fields):
 - Consists of a set of nodes with undirected edges between some of the nodes
 - Edges (or more accurately the lack of edges) encode conditional independence.

Probabilistic Graphical Models

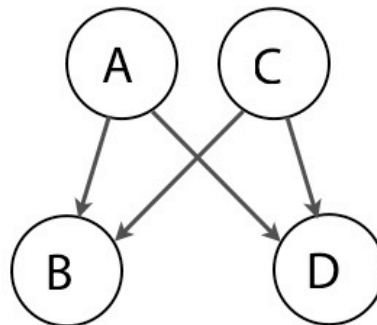
Some conditional independencies cannot be represented by directed graphical models:

- Consider 4 variables: A, B, C, D

- How do we represent the conditional independencies: $(A \perp C \mid B, D)$ $(B \perp D \mid A, C)$

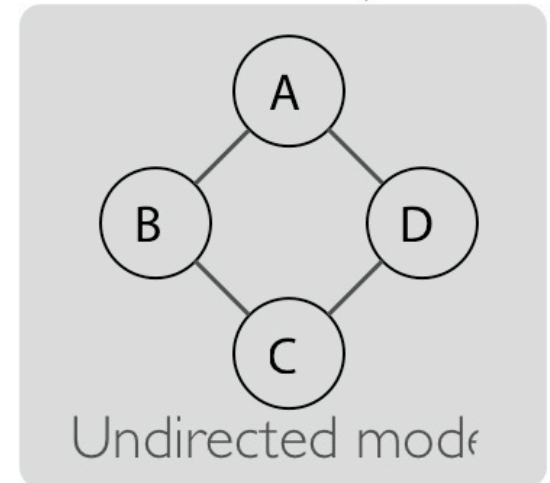


$(A \perp C \mid B, D)$
 $(B \perp D \mid A)$



$(A \perp C)$
 $(B \perp D \mid A, C)$

$(A \perp C \mid B, D)$
 $(B \perp D \mid A, C)$



Probabilistic Graphical Models

Sometime its awkward to model phenomena with directed models

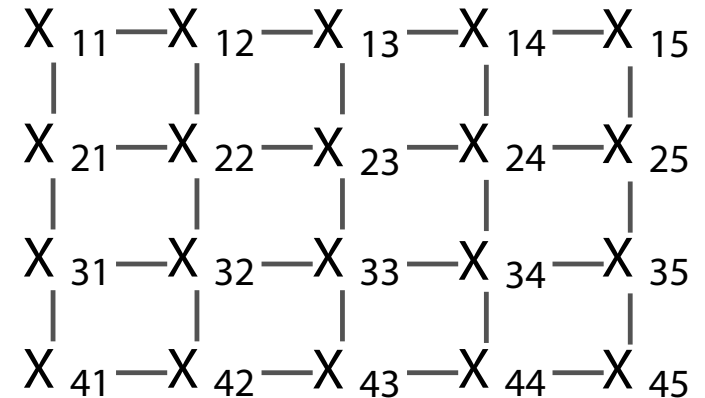
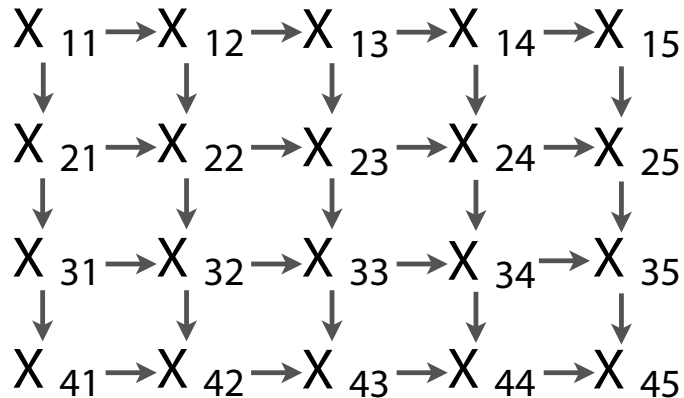
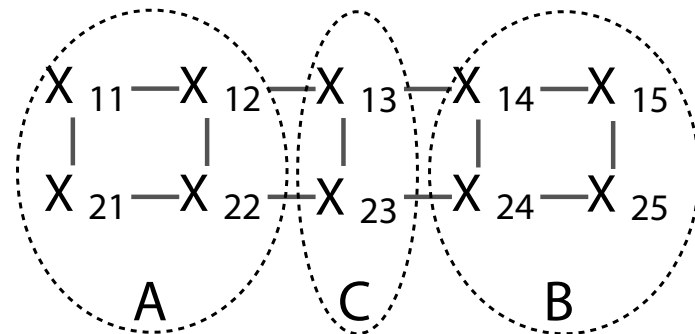


Image from "CRF as RNN Semantic Image Segmentation Live Demo" (http://www.robots.ox.ac.uk/~szheng/crfasrnn_demo/)

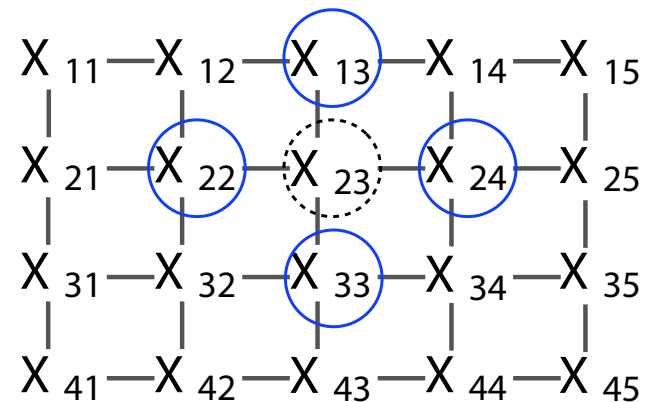
Conditional Independence

- Undirected graphical models:
 - Conditional independence encoded by simple graph separation.
 - Formally, consider 3 sets of nodes: **A**, **B** and **C**, we say $\mathbf{x}_A \perp \mathbf{x}_B \mid \mathbf{x}_C$ iff **C** separates **A** and **B** in the graph.
 - **C** separates **A** and **B** in the graph: If we remove all nodes in **C**, there is no path from **A** to **B** in the graph.



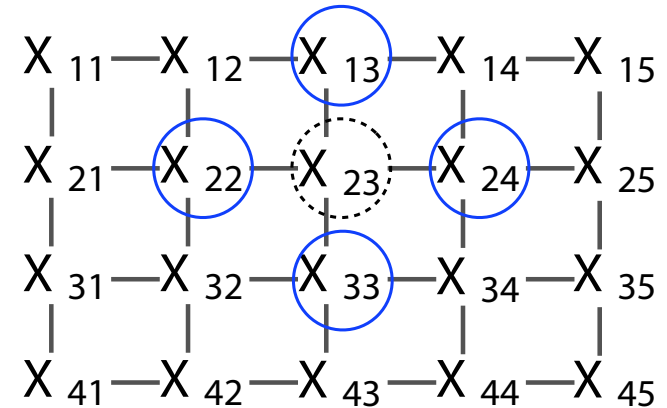
Markov blanket

- Markov Blanket: For a given node \mathbf{x} , Markov Blanket is smallest set of nodes which renders \mathbf{x} conditionally independent of all other nodes in graph.
- Markov blanket of the 2-d lattice MRF:

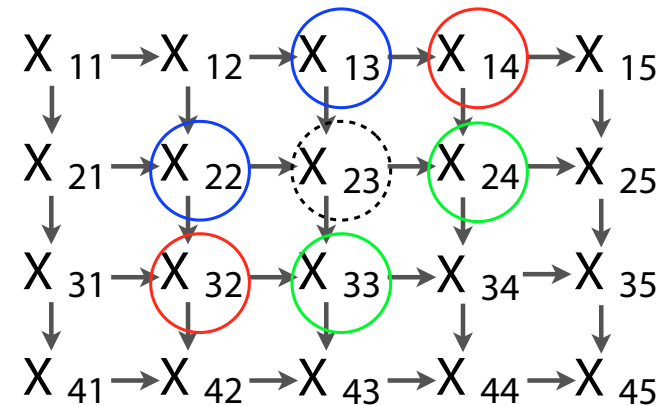
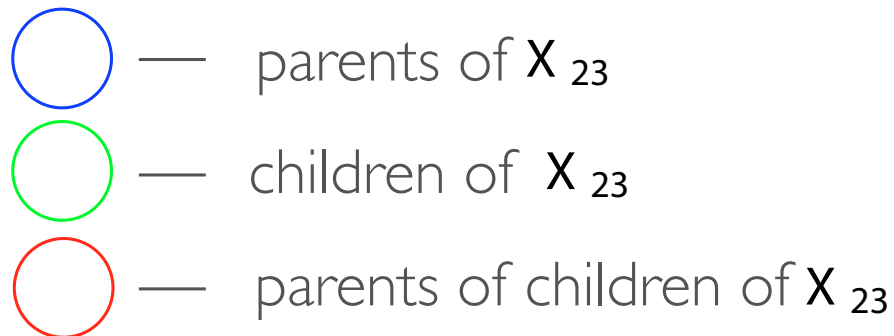


Directed vs. Undirected

- Markov blanket of the 2-d lattice MRF:



- Markov blanket of the 2-d causal MRF:



Parametrizing directed

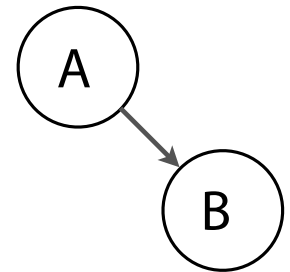
Directed graphical models:

- Parameterized by local conditional probability densities (CPDs)

$$P(A \mid B)$$

- Joint distributions are given as products of CPDs:

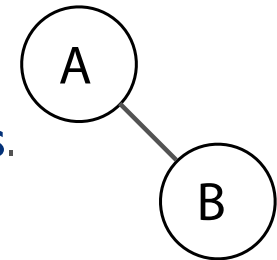
$$P(X_1, \dots, X_N) = \prod_{i=0}^N P(X_i \mid X_{\text{parents}(i)})$$



Parametrizing Markov nets

Undirected graphical models:

- Parameterized by symmetric **factors** or **potential functions**.



$$\varphi(A, B)$$

- Generalizes both the CPD and the joint distribution.
- Note: unlike CPDs, potential functions are not required to normalize.
- **Definition:** Let \mathbf{C} be a set of cliques. For each \mathbf{c} in \mathbf{C} , we define a factor (called potential function or clique potential) $\varphi_{\mathbf{c}}$ as a nonnegative function

$$\varphi_{\mathbf{c}}(\mathbf{x}_{\mathbf{c}}) \text{ in } \mathbb{R}$$

where $\mathbf{x}_{\mathbf{c}}$ is the set of variables in clique \mathbf{c} .

Parametrizing Markov nets

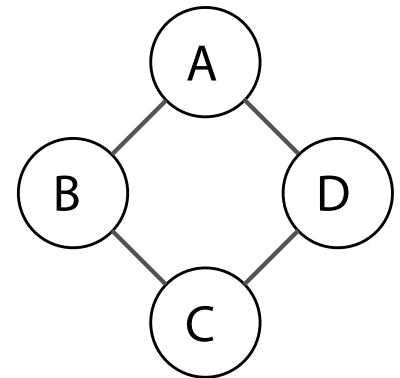
- Joint distribution given by a normalized product of factors:

$$P(x_1, \dots, x_n) = \frac{1}{Z} \prod_c \phi_c(x_c)$$

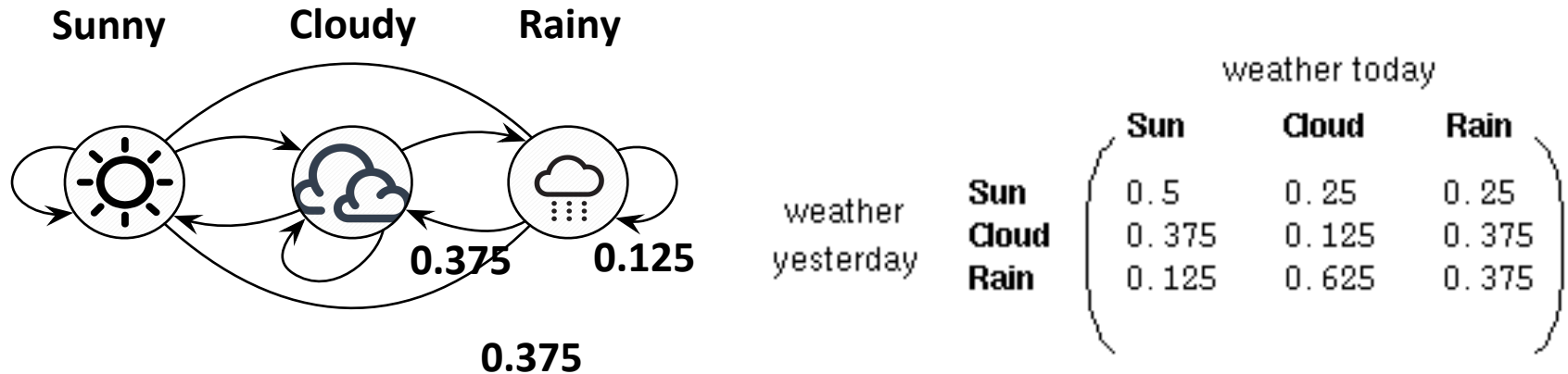
- Z is the **partition function**, it's the normalization constant: $\sum_{x_1, \dots, x_n} \prod_c \phi_c(x_c)$
- Our 4 variable example:

$$P(a, b, c, d) = \frac{1}{Z} \phi_1(a, b) \phi_2(b, c) \phi_3(c, d) \phi_4(d, a)$$

$$Z = \sum_{a,b,c,d} \phi_1(a, b) \phi_2(b, c) \phi_3(c, d) \phi_4(d, a)$$



Markov Chains



States : Three states sunny, cloudy, rainy.

State transition matrix : The probability of the weather given the previous day's weather.

Sun	Cloud	Rain
1.0	0.0	0.0

Initial Distribution : Defining the probability of the system being in each of the states at time 0.

$$\begin{aligned} \mathcal{P}(\text{Sunny}) &= 1.0 & \mathcal{P}(\text{Sunny}|\text{Rainy}) &= 0.125 \\ \mathcal{P}(\text{Rainy}) &= 0.0 & \mathcal{P}(\text{Cloudy}|\text{Rainy}) &= 0.625 \end{aligned}$$

Markov Models

- Set of states: $\{s_1, s_2, \dots, s_N\}$
- Process moves from one state to another generating a sequence of states : $s_{i1}, s_{i2}, \dots, s_{ik}, \dots$
- Markov chain property: probability of each subsequent state depends only on what was the previous state:

$$P(s_{ik} \mid s_{i1}, s_{i2}, \dots, s_{ik-1}) = P(s_{ik} \mid s_{ik-1})$$

- To define Markov model, the following probabilities have to be specified: transition probabilities $a_{ij} = P(s_i \mid s_j)$ and initial probabilities $\pi_i = P(s_i)$

Markov Models

- Can we find probability of an entire sequence of event ?

$$P(s_{i1}, s_{i2}, \dots, s_{ik})$$

- By Markov chain property (the probability of being in particular state depends only on the previous one), the probability of state sequence can be found by the formula:

$$\begin{aligned} P(s_{i1}, s_{i2}, \dots, s_{ik}) &= P(s_{ik} | s_{i1}, s_{i2}, \dots, s_{ik-1}) P(s_{i1}, s_{i2}, \dots, s_{ik-1}) \\ &= P(s_{ik} | s_{ik-1}) P(s_{i1}, s_{i2}, \dots, s_{ik-1}) = \dots \\ &= P(s_{ik} | s_{ik-1}) P(s_{ik-1} | s_{ik-2}) \dots P(s_{i2} | s_{i1}) P(s_{i1}) \end{aligned}$$

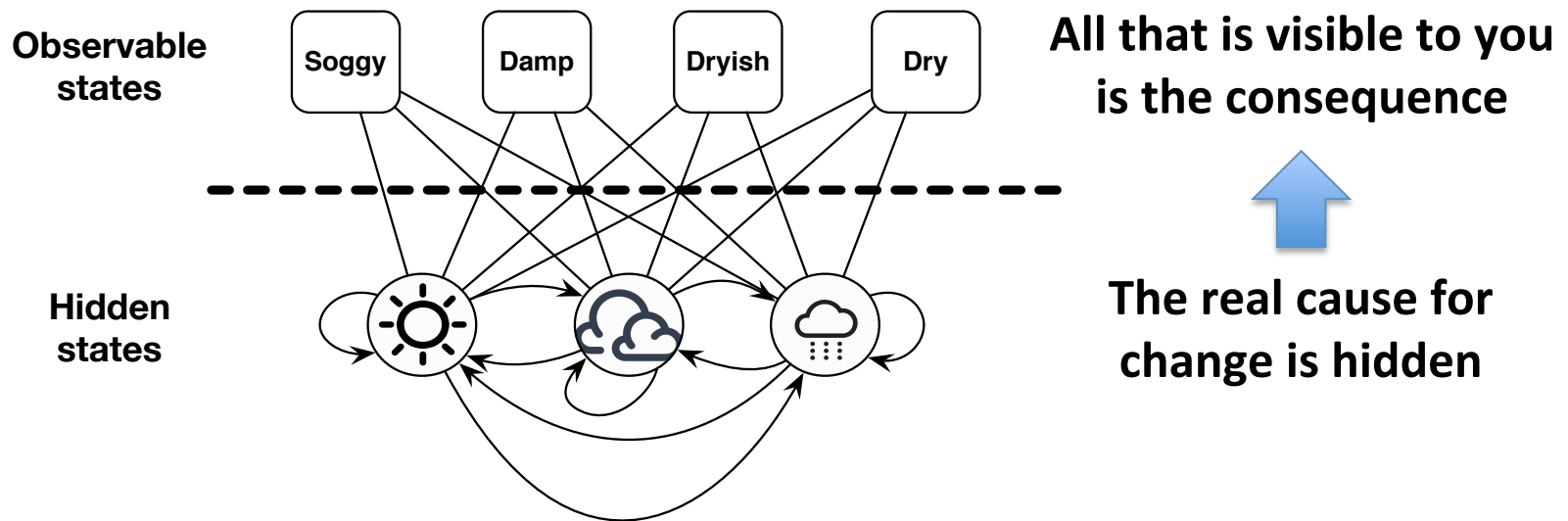
- Suppose we want to calculate a probability of a sequence of states in our example, { 'Sunny' , ' Sunny' , ' Cloudy' , Rainy' }

$$\begin{aligned} \mathcal{P}(\{Sunny, Sunny, Cloudy, Rainy\}) &= \\ \mathcal{P}(Rainy|Cloudy)\mathcal{P}(Cloudy|Sunny)\mathcal{P}(Sunny|Sunny)\mathcal{P}(Sunny) \end{aligned}$$

$$0.375 \times 0.275 \times 0.5 \times 1.0 = 0.052$$

Hidden Markov Models

- Now imagine that there has been a nuclear war
- Trapped in a cave, you still want to know the weather (habits die hard)
- All that you can see is the state of the mud in the cave



Hidden states : the (TRUE) states of a system that may be described by a Markov process (e.g., the weather).

Observable states : the states of the process that are 'visible' (e.g., mud in the cave).

Hidden Markov Models

- Set of states: $\{s_1, s_2, \dots, s_N\}$
- Process moves from one state to another generating a sequence of states : $s_{i1}, s_{i2}, \dots, s_{ik}, \dots$
- Markov chain property: probability of each subsequent state depends only on what was the previous state:

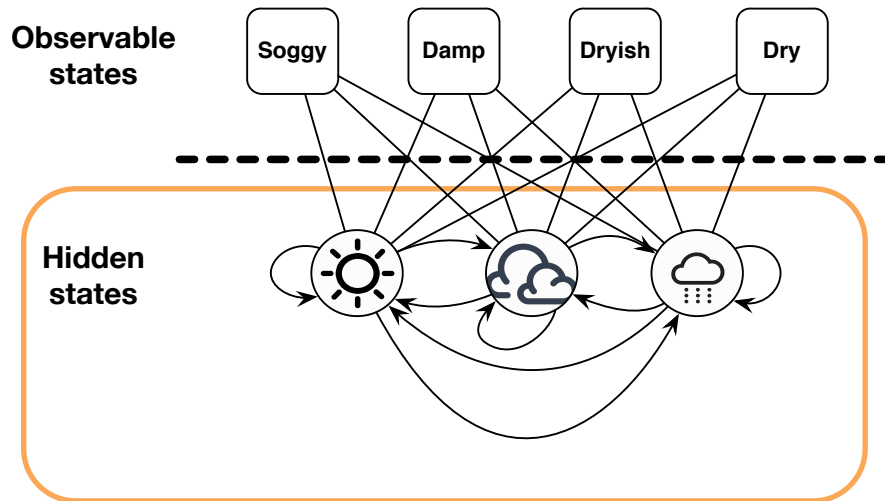
$$P(s_{ik} \mid s_{i1}, s_{i2}, \dots, s_{ik-1}) = P(s_{ik} \mid s_{ik-1})$$

We keep all properties of Markov chains

- States are not visible, but each state randomly generates one of M observations (or visible states)



Markov Models



Markov chain



State transition weather today

weather yesterday	Sun	Cloud	Rain
Sun	0.5	0.25	0.25
Cloud	0.375	0.125	0.375
Rain	0.125	0.625	0.375

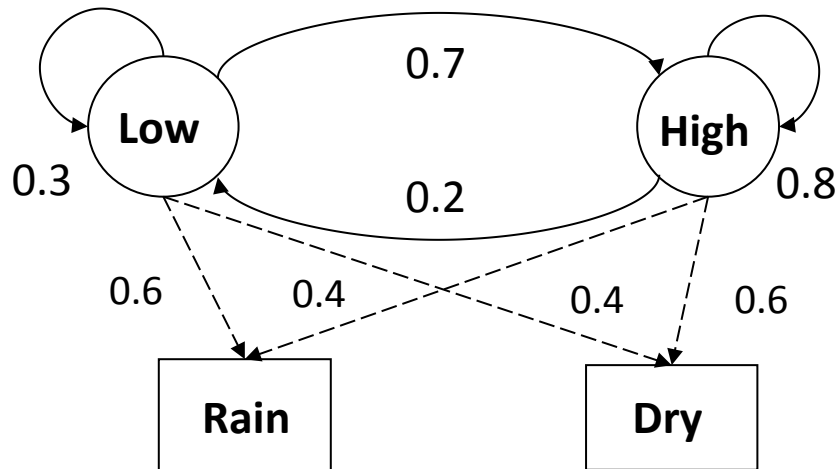
	Sun	Cloud	Rain
	1.0	0.0	0.0

Initial distribution

		Seaweed			
		Dry	Dryish	Damp	Soggy
weather	Sun	0.60	0.20	0.15	0.05
	Cloud	0.25	0.25	0.25	0.25
	Rain	0.05	0.10	0.35	0.50

Output matrix : containing the probability of observing a particular observable state given that the hidden model is in a particular hidden state.

Simpler example



But in fact this is the **consequence** of the atmospheric pressure

We can observe outside that it rains or not (**visible state**)

We want to calculate a probability of a **sequence of observations** : { 'Dry' , ' Rain' }.

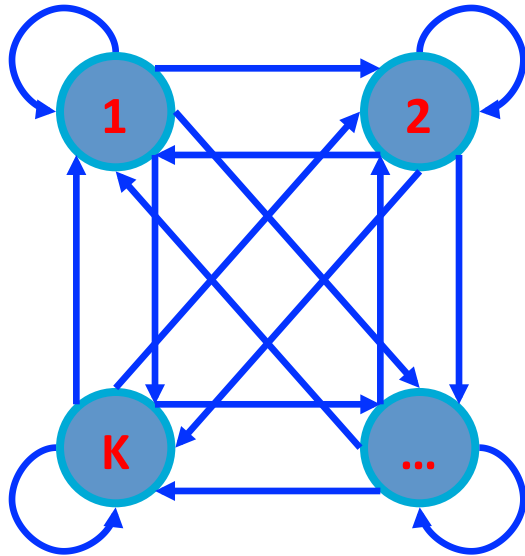
Consider all possible hidden state sequences:

$$P(\{ \text{'Dry' , ' Rain' } \}) = P(\{ \text{'Dry' , ' Rain' } \}, \{ \text{'Low' , ' Low' } \}) + P(\{ \text{'Dry' , ' Rain' } \}, \{ \text{'Low' , ' High' } \}) + P(\{ \text{'Dry' , ' Rain' } \}, \{ \text{'High' , ' Low' } \}) + P(\{ \text{'Dry' , ' Rain' } \}, \{ \text{'High' , ' High' } \})$$

For the first term we have

$$\begin{aligned} &P(\{ \text{'Dry' , ' Rain' } \}, \{ \text{'Low' , ' Low' } \}) = \\ &P(\{ \text{'Dry' , ' Rain' } \} \mid \{ \text{'Low' , ' Low' } \}) P(\{ \text{'Low' , ' Low' } \}) = \\ &P(\text{'Dry' } \mid \text{'Low' }) P(\text{'Rain' } \mid \text{'Low' }) P(\text{'Low' }) P(\text{'Low' } \mid \text{'Low' }) = 0.4 * 0.4 * 0.6 * 0.4 * 0.3 \end{aligned}$$

A HMM is memoryless



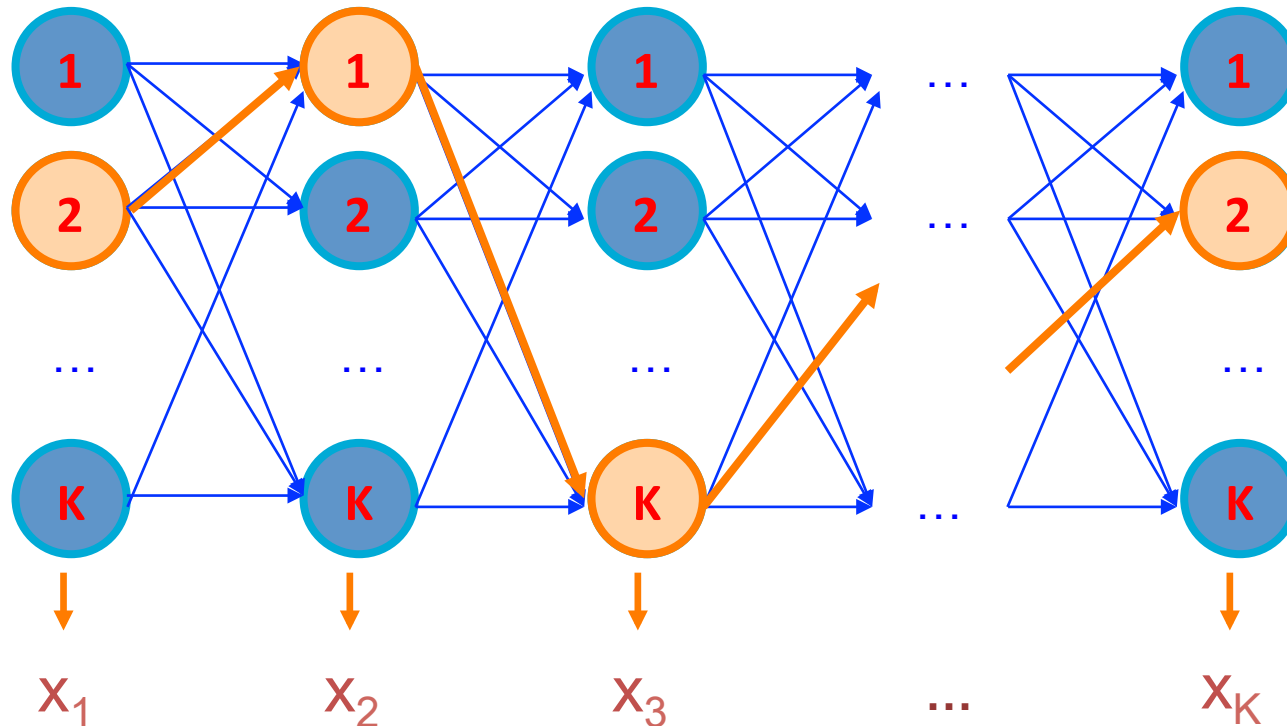
- At any time step t ,
- The only thing that affects future states $t+1$
- **Is the current state π_t**

$$\begin{aligned} P(\pi_{t+1} = k \mid \text{“whatever happened so far”}) &= \\ P(\pi_{t+1} = k \mid \pi_1, \pi_2, \dots, \pi_t, x_1, x_2, \dots, x_t) &= \\ P(\pi_{t+1} = k \mid \pi_t) \end{aligned}$$

Parse of a sequence

Given a sequence $\mathbf{x} = x_1, \dots, x_N$, (observations)

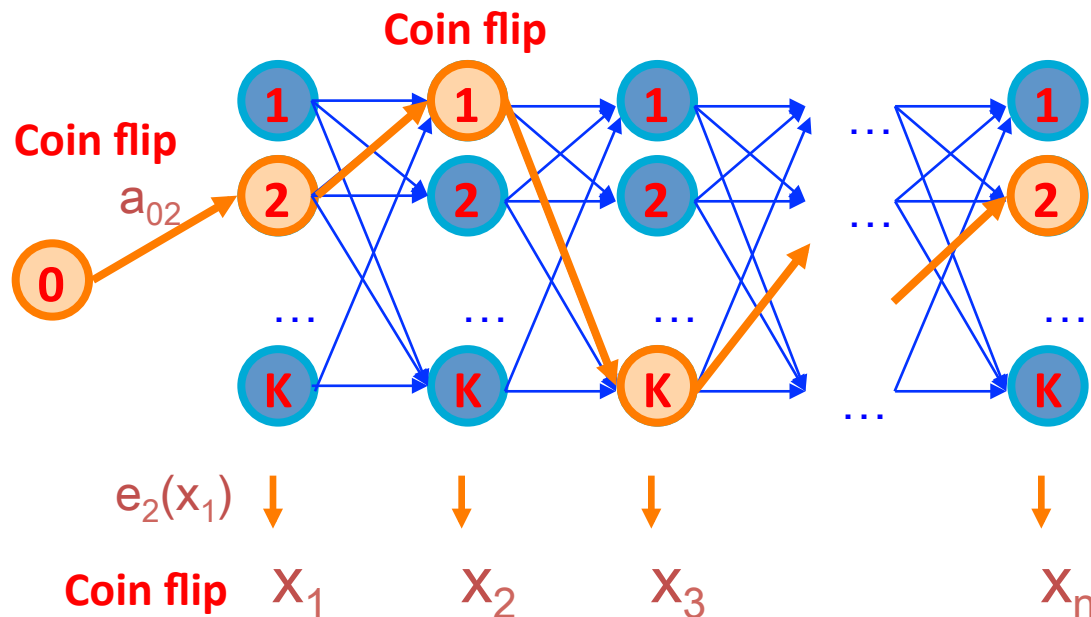
A parse of \mathbf{x} is a sequence of states $\pi = \pi_1, \dots, \pi_N$
(hidden)



Generate a sequence by the model

Given a HMM, we can generate a sequence of length n as follows

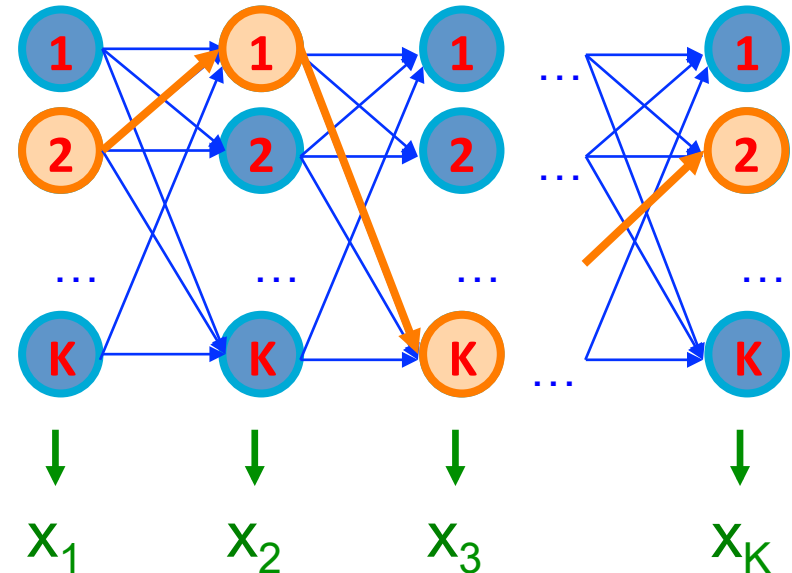
1. Start at state π_1 according to prob $a_{0\pi_1}$
2. Emit letter x_1 according to prob $e_{\pi_1}(x_1)$
3. Go to state π_2 according to prob $a_{\pi_1\pi_2}$
4. ... until emitting x_n



Likelihood of a parse

Given a sequence $\mathbf{x} = x_1 \dots x_N$
and a parse $\pi = \pi_1, \dots, \pi_N$,

To find how **likely this scenario** is
(given our HMM)
Compute the **likelihood of a parse**



$$\begin{aligned} P(\mathbf{x}, \pi) &= P(x_1, \dots, x_N, \pi_1, \dots, \pi_N) = \\ &P(x_N | \pi_N) P(\pi_N | \pi_{N-1}) \dots P(x_2 | \pi_2) P(\pi_2 | \pi_1) P(x_1 | \pi_1) P(\pi_1) = \\ &a_{0\pi_1} a_{\pi_1\pi_2} \dots a_{\pi_{N-1}\pi_N} e_{\pi_1}(x_1) \dots e_{\pi_N}(x_N) \end{aligned}$$

Transition probabilities

Output (emitting) probabilities

Three main questions in HMMs

1. Decoding

Given a HMM $M=(T, E, \pi)$, a sequence $\mathbf{x}=\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$,

Find the most likely sequence π of states that produced the sequence \mathbf{x} , ie. π that maximizes $P(\mathbf{x}, \pi|M)$

2. Evaluation

Given a HMM $M=(T, E, \pi)$, a sequence $\mathbf{x}=\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$,

Find probability that \mathbf{x} was generated by M , ie. $P(\mathbf{x}|M)$

3. Learning

Given a dataset of training sequences X ,

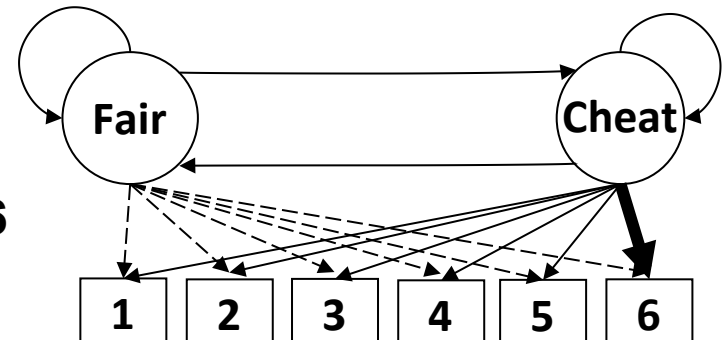
Find the HMM $M=(T, E, \pi)$ with its transition probabilities T and its emission probabilities E that maximize $P(X|T, E, \pi)$

Simple example:

A casino with two dices (fair, loaded)

The fair dice is our usual honest dice

The cheat dice produces awful amounts of 6



Question #1 - Decoding

GIVEN

A sequence of rolls by the casino player



QUESTION

What portion of the sequence was generated with the fair die, and what portion with the loaded die?


This is the **DECODING** question in HMMs

Question #2 - Evaluation

GIVEN

A sequence of rolls by the casino player

1245526462146146136136661664661636616366163616515615115146123562344



Prob = 1.3×10^{-35}

QUESTION

How likely is this sequence, given our model of how the casino works?

This is the **EVALUATION** problem in HMMs

Question #3 - Learning

GIVEN

Many sequences of rolls by several casino players

1245526462146146136136661664661636616366163616515615115146123562344

Prob(6) = 64%

QUESTION

How “loaded” is the loaded die? How “fair” is the fair die? How often does the casino player change from fair to loaded, and back?

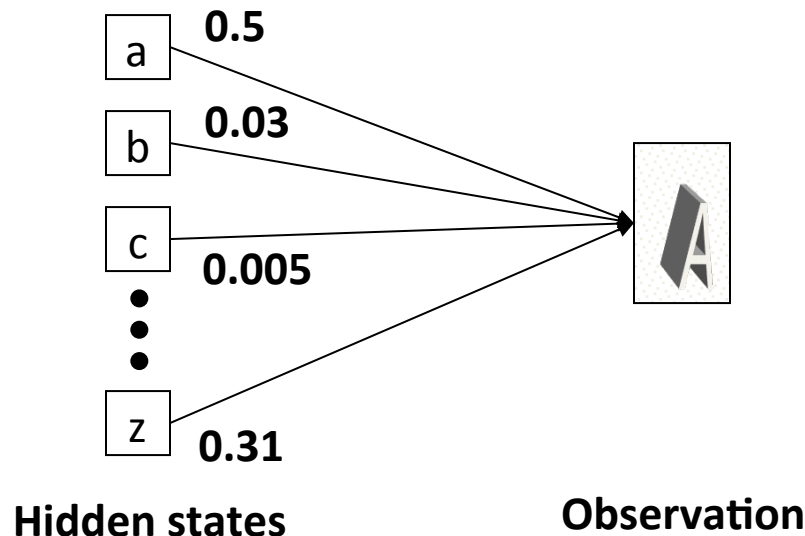
This is the **LEARNING** question in HMMs

Ugly wordart recognition (1)

- Ugly wordart recognition, all characters separated.



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



Ugly wordart recognition (2)

- Hidden states of HMM = **characters**.
- **Observations** = typed images of characters segmented from the image v_α . Note that there is an infinite number of observations

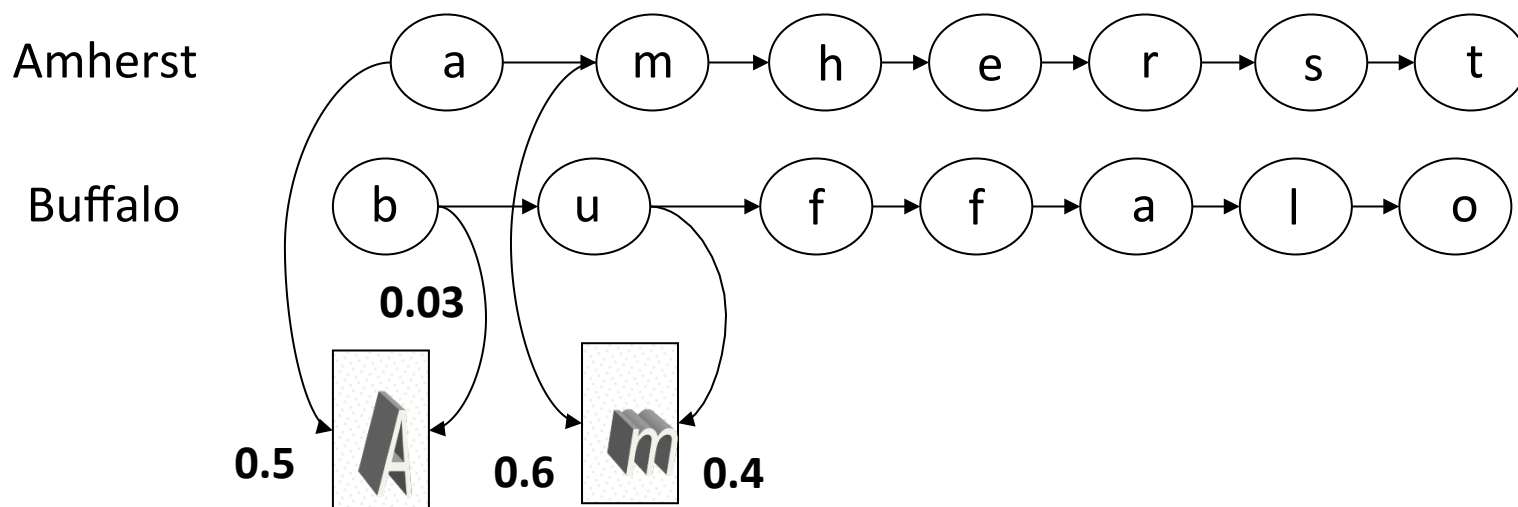
- Observation **probabilities** = character **recognizer scores**.

$$B = (b_i(v_\alpha)) = (P(v_\alpha | s_i))$$

- Transition probabilities will be defined differently in two subsequent models.

Ugly wordart recognition (3)

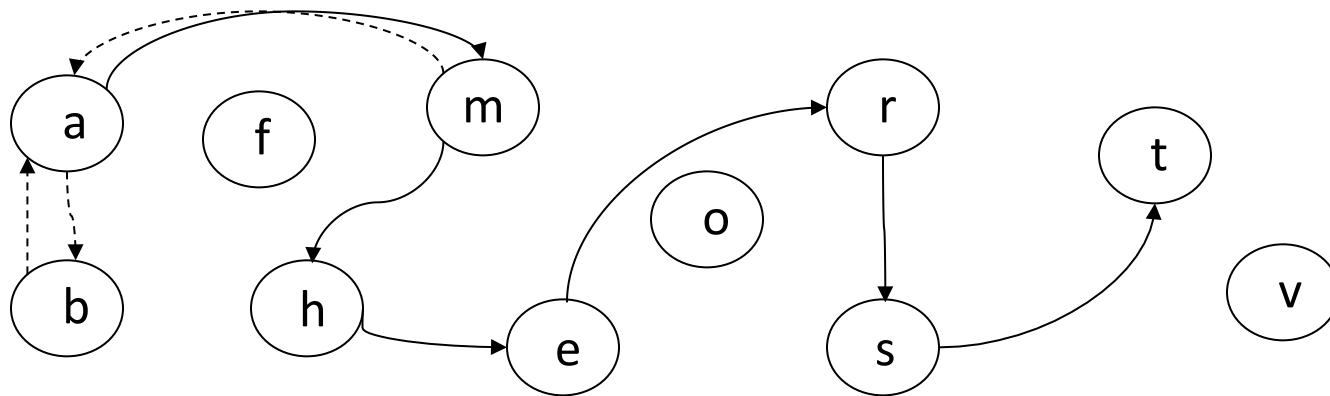
- If a dictionary is given, we can construct separate HMM models for each word in it.



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

Ugly wordart recognition (4)

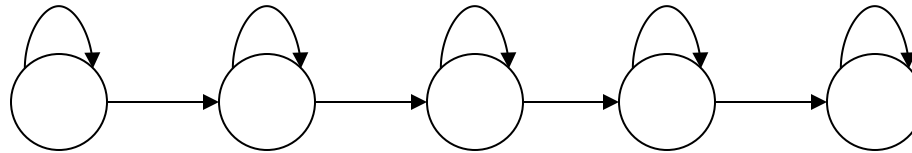
- We can construct a single HMM for all words.
- Hidden states = all characters in the alphabet.
- Transition probabilities and initial probabilities are calculated from language model.
- Observations and observation probabilities are as before.



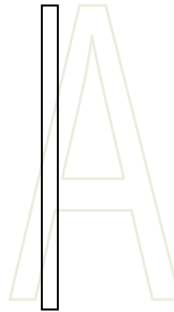
- Here we have to determine the best sequence of hidden states, the one that most likely produced word image.
- This is an application of **Decoding problem**.

Character recognition (5)

- The structure of hidden states is chosen.



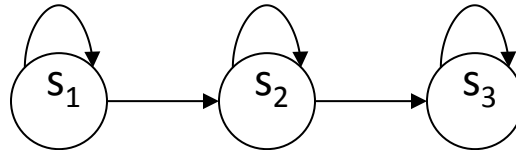
- Observations are feature vectors extracted from vertical slices.



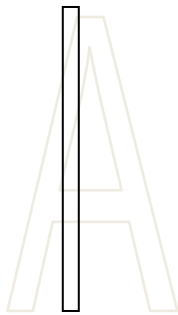
- Probabilistic mapping from hidden state to feature vectors
 1. Use mixture of Gaussian models
 2. Quantize feature vector space.

Markov Models

- The structure of hidden states:



- Observation = number of islands in the vertical slice.



• **HMM for character 'A' :**
 Transition probabilities: $\{a_{ij}\} = \begin{pmatrix} .8 & .2 & 0 \\ 0 & .8 & .2 \\ 0 & 0 & 1 \end{pmatrix}$

Observation probabilities: $\{b_{jk}\} = \begin{pmatrix} .9 & .1 & 0 \\ .1 & .8 & .1 \\ .9 & .1 & 0 \end{pmatrix}$



• **HMM for character 'B' :**
 Transition probabilities: $\{a_{ij}\} = \begin{pmatrix} .8 & .2 & 0 \\ 0 & .8 & .2 \\ 0 & 0 & 1 \end{pmatrix}$

Observation probabilities: $\{b_{jk}\} = \begin{pmatrix} .9 & .1 & 0 \\ 0 & .2 & .8 \\ .6 & .4 & 0 \end{pmatrix}$

Markov Models

- Suppose we observe the sequence {1, 3, 2, 1}
- What HMM is more likely to generate it ? HMM for 'A' or HMM for 'B'
- Consider likelihood of generating given observation for each possible sequence of hidden states:

• HMM for character 'A'

Hidden state sequence	Transition probabilities	Observation probabilities
$s_1 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3$	$.8 * .2 * .2$	$.9 * 0 * .8 * .9 = 0$
$s_1 \rightarrow s_2 \rightarrow s_2 \rightarrow s_3$	$.2 * .8 * .2$	$.9 * .1 * .8 * .9 = 0.0020736$
$s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_3$	$.2 * .2 * 1$	$.9 * .1 * .1 * .9 = 0.000324$
Total = 0.0023976		

• HMM for character 'B'

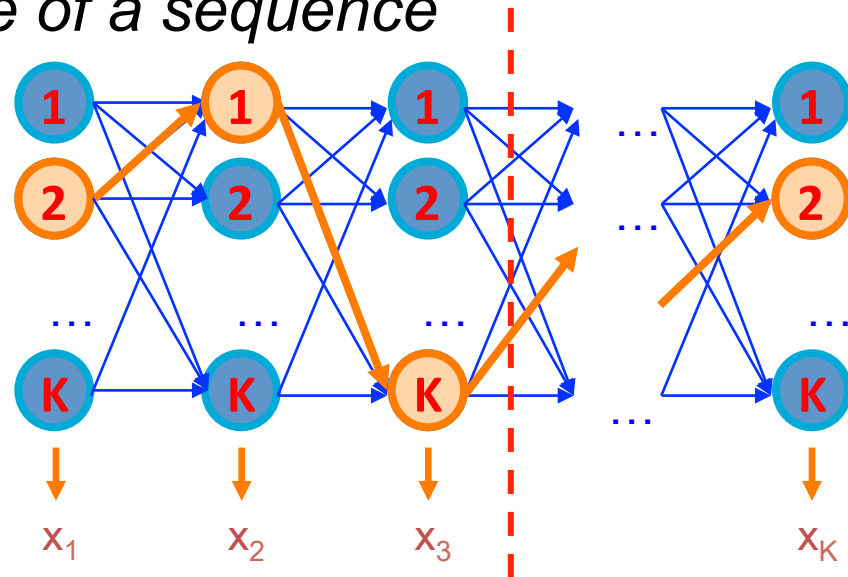
Hidden state sequence	Transition probabilities	Observation probabilities
$s_1 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3$	$.8 * .2 * .2$	$.9 * 0 * .2 * .6 = 0$
$s_1 \rightarrow s_2 \rightarrow s_2 \rightarrow s_3$	$.2 * .8 * .2$	$.9 * .8 * .2 * .6 = 0.0027648$
$s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_3$	$.2 * .2 * 1$	$.9 * .8 * .4 * .6 = 0.006912$
Total = 0.0096768		

Problem 1 - Decoding

Find the most likely parse of a sequence

Given $x = x_1 x_2 \dots x_N$
Find $\pi = \pi_1, \dots, \pi_N$,
 to maximize $P[x, \pi]$

$$\pi^* = \operatorname{argmax}_{\pi} P[x, \pi]$$



Maximizes

$$a_{0\pi_1} e_{\pi_1}(x_1) a_{\pi_1\pi_2} \dots a_{\pi_{N-1}\pi_N} e_{\pi_N}(x_N)$$

Given that we end up in state k at step i , maximize product to the left and right

Dynamic Programming!

$$V_k(i) = \max_{\{\pi_1 \dots \pi_{i-1}\}} P[x_1 \dots x_{i-1}, \pi_1, \dots, \pi_{i-1}, x_i, \pi_i = k]$$

= Prob. of most likely sequence of states ending at state $\pi_i = k$

Decoding – Main idea

Induction: Given that for **all states** k , and for a **fixed position** i ,

$$V_k(i) = \max_{\{\pi_1 \dots \pi_{i-1}\}} P[x_1 \dots x_{i-1}, \pi_1, \dots, \pi_{i-1}, x_i, \pi_i = k]$$

What is $V_l(i+1)$? (the next state l at position $i+1$)

From definition,

$$\begin{aligned} V_l(i+1) &= \max_{\{\pi_1 \dots \pi_i\}} P[x_1 \dots x_i, \pi_1, \dots, \pi_i, x_{i+1}, \pi_{i+1} = l] \\ &= \max_{\{\pi_1 \dots \pi_i\}} P(x_{i+1}, \pi_{i+1} = l \mid x_1 \dots x_i, \pi_1, \dots, \pi_i) P[x_1 \dots x_i, \pi_1, \dots, \pi_i] \\ &= \max_{\{\pi_1 \dots \pi_i\}} P(x_{i+1}, \pi_{i+1} = l \mid \pi_i) P[x_1 \dots x_{i-1}, \pi_1, \dots, \pi_{i-1}, x_i, \pi_i] \\ &= \max_k [P(x_{i+1}, \pi_{i+1} = l \mid \pi_i = k) \max_{\{\pi_1 \dots \pi_{i-1}\}} P[x_1 \dots x_{i-1}, \pi_1, \dots, \pi_{i-1}, x_i, \pi_i = k]] \\ &= \max_k [P(x_{i+1} \mid \pi_{i+1} = l) P(\pi_{i+1} = l \mid \pi_i = k) V_k(i)] \end{aligned}$$

Emission probability

Probability of $k \rightarrow l$

Previous max

Decoding – Main idea

Induction: Given that for **all states** k , and for a **fixed position** i ,

$$\mathbf{V}_k(i) = \max_{\{\pi_1 \dots \pi_{i-1}\}} \mathbf{P}[\mathbf{x}_1 \dots \mathbf{x}_{i-1}, \pi_1, \dots, \pi_{i-1}, \mathbf{x}_i, \pi_i = k]$$

What is $V_l(i+1)$? (the next state l at position $i+1$)

From definition

$$\begin{aligned} V_l(i+1) &= \max_{\{\pi_1 \dots \pi_i\}} P[\mathbf{x}_1 \dots \mathbf{x}_i, \pi_1, \dots, \pi_i, \mathbf{x}_{i+1}, \pi_{i+1} = l] \\ &= \max_{\{\pi_1 \dots \pi_i\}} P(\mathbf{x}_{i+1}, \pi_{i+1} = l \mid \mathbf{x}_1 \dots \mathbf{x}_i, \pi_1, \dots, \pi_i) P[\mathbf{x}_1 \dots \mathbf{x}_i, \pi_1, \dots, \pi_i] \\ &= \max_{\{\pi_1 \dots \pi_i\}} P(\mathbf{x}_{i+1}, \pi_{i+1} = l \mid \pi_i) P[\mathbf{x}_1 \dots \mathbf{x}_{i-1}, \pi_1, \dots, \pi_{i-1}, \mathbf{x}_i, \pi_i] \\ &= \max_k [P(\mathbf{x}_{i+1}, \pi_{i+1} = l \mid \pi_i = k) \max_{\{\pi_1 \dots \pi_{i-1}\}} \mathbf{P}[\mathbf{x}_1 \dots \mathbf{x}_{i-1}, \pi_1, \dots, \pi_{i-1}, \mathbf{x}_i, \pi_i = k]] \\ &= \max_k [P(\mathbf{x}_{i+1} \mid \pi_{i+1} = l) P(\pi_{i+1} = l \mid \pi_i = k) \mathbf{V}_k(i)] \\ &= e_l(\mathbf{x}_{i+1}) \max_k a_{kl} \mathbf{V}_k(i) \end{aligned}$$

Decoding – Viterbi algorithm

Input: $x = x_1 \dots x_N$

Initialization:

$$V_0(0) = 1 \quad (0 \text{ is the imaginary first position})$$

$$V_k(0) = 0, \text{ for all } k > 0$$

Iteration:

$$V_j(i) = e_j(x_i) \times \max_k a_{kj} V_k(i-1)$$

$$\text{Ptr}_j(i) = \operatorname{argmax}_k a_{kj} V_k(i-1)$$

Termination:

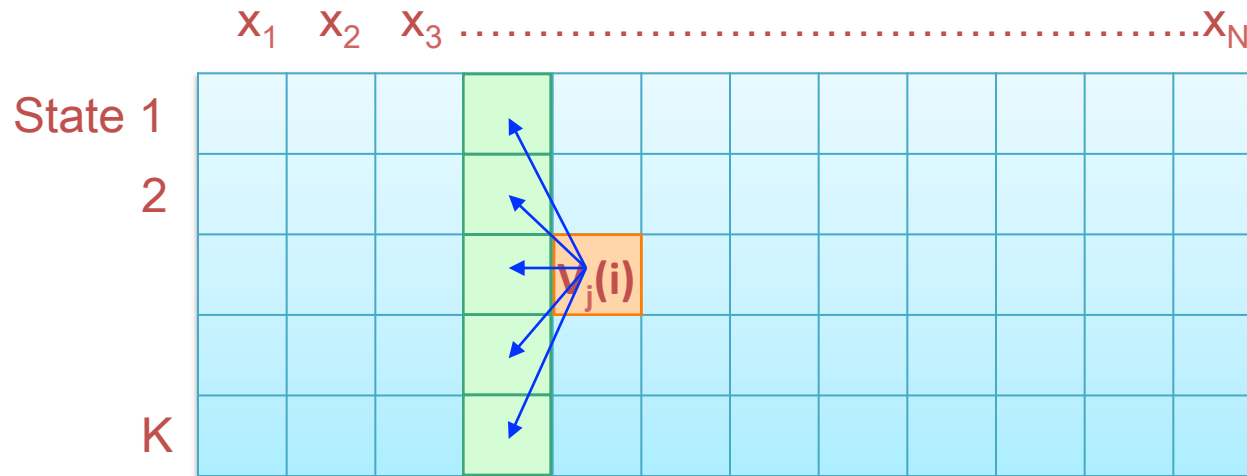
$$P(x, \pi^*) = \max_k V_k(N)$$

Traceback:

$$\pi_N^* = \operatorname{argmax}_k V_k(N)$$

$$\pi_{i-1}^* = \text{Ptr}_{\pi_i}(i)$$

Decoding – Viterbi algorithm



Complexity in time:

$$O(K^2N)$$

Complexity in space:

$$O(KN)$$

Decoding – Underflow problem

Underflows are a significant problem

$$P[\mathbf{x}_1, \dots, \mathbf{x}_i, \pi_1, \dots, \pi_i] = a_{0\pi_1} a_{\pi_1\pi_2} \dots a_{\pi_i} e_{\pi_1}(\mathbf{x}_1) \dots e_{\pi_i}(\mathbf{x}_i)$$

These numbers become extremely small – underflow

Solution: Take the logs of all values

$$V_l(i) = \log e_k(\mathbf{x}_i) + \max_k [V_k(i-1) + \log a_{kl}]$$

Decoding – Example

Let x be a long sequence with a portion of $\sim 1/6$ 6's,
followed by a portion of $\sim 1/2$ 6's...

$x = 123456123456\dots 123456 \text{ } 6626364656\dots 1626364656$

Then, it is not hard to show that optimal parse is (exercise):

FFF.....F LLL.....L

6 characters “123456” parsed as F, contribute $.95^6 \times (1/6)^6 = 1.6 \times 10^{-5}$
parsed as L, contribute $.95^6 \times (1/2)^1 \times (1/10)^5 = 0.4 \times 10^{-5}$

“162636” parsed as F, contribute $.95^6 \times (1/6)^6 = 1.6 \times 10^{-5}$
parsed as L, contribute $.95^6 \times (1/2)^3 \times (1/10)^3 = 9.0 \times 10^{-5}$

Problem 2 - Evaluation

Compute the likelihood that a sequence is generated by the model

We will develop algorithms that allow us to compute:

$P(x)$ Probability of x given the model

$P(x_i \dots x_j)$ Probability of a substring of x given the model

$P(\pi_i = k \mid x)$ “**Posterior**” probability that the i^{th} state is k , given x

A more refined measure of which states x may be in

The forward algorithm

We want to calculate

$P(x)$ = probability of x , given the HMM

Sum over all possible ways of generating x :

$$P(x) = \sum_{\pi} P(x, \pi) = \sum_{\pi} P(x \mid \pi) P(\pi)$$

To avoid summing over an exponential number of paths π , define

$$f_k(i) = P(x_1 \dots x_i, \pi_i = k) \quad (\text{the forward probability})$$

“generate i first observations and end up in state k ”

The forward algorithm - derivation

Define the forward probability:

$$\begin{aligned} f_k(i) &= P(x_1 \dots x_i, \pi_i = k) \\ &= \sum_{\pi_1 \dots \pi_{i-1}} P(x_1 \dots x_{i-1}, \pi_1, \dots, \pi_{i-1}, \pi_i = k) e_k(x_i) \\ &= \sum_l \sum_{\pi_1 \dots \pi_{i-2}} P(x_1 \dots x_{i-1}, \pi_1, \dots, \pi_{i-2}, \pi_{i-1} = l) a_{lk} e_k(x_i) \\ &= \sum_l P(x_1 \dots x_{i-1}, \pi_{i-1} = l) a_{lk} e_k(x_i) \\ &= e_k(x_i) \sum_l f_l(i-1) a_{lk} \end{aligned}$$

The forward algorithm

We can compute $f_k(i)$ for all k, i , using dynamic programming!

Initialization:

$$f_0(0) = 1$$

$$f_k(0) = 0, \text{ for all } k > 0$$

Iteration:

$$f_k(i) = e_k(x_i) \sum_l f_l(i-1) a_{lk}$$

Termination:

$$P(x) = \sum_k f_k(N)$$

Relation Forward / Viterbi

VITERBI

Initialization:

$$V_0(0) = 1$$

$$V_k(0) = 0, \text{ for all } k > 0$$

Iteration:

$$V_j(i) = e_j(x_i) \max_k V_k(i-1) a_{kj}$$

Termination:

$$P(x, \pi^*) = \max_k V_k(N)$$

FORWARD

Initialization:

$$f_0(0) = 1$$

$$f_k(0) = 0, \text{ for all } k > 0$$

Iteration:

$$f_l(i) = e_l(x_i) \sum_k f_k(i-1) a_{kl}$$

Termination:

$$P(x) = \sum_k f_k(N)$$

Motivation for Backward algorithm

We want to compute

$$P(\pi_i = k \mid x),$$

the probability distribution on the i^{th} position, given x

We start by computing

$$\begin{aligned} P(\pi_i = k, x) &= P(x_1 \dots x_i, \pi_i = k, x_{i+1} \dots x_N) \\ &= P(x_1 \dots x_i, \pi_i = k) P(x_{i+1} \dots x_N \mid x_1 \dots x_i, \pi_i = k) \\ &= \boxed{P(x_1 \dots x_i, \pi_i = k)} \boxed{P(x_{i+1} \dots x_N \mid \pi_i = k)} \\ &\quad \text{Forward, } f_k(i) \quad \text{Backward, } b_k(i) \end{aligned}$$

$$\text{Then, } P(\pi_i = k \mid x) = P(\pi_i = k, x) / P(x)$$

Backward algorithm - derivation

Define the backward probability:

$$b_k(i) = P(x_{i+1} \dots x_N \mid \pi_i = k) \text{ “starting from } i^{\text{th}} \text{ state } = k, \text{ generate rest of } x \text{”}$$

$$= \sum_{\pi_{i+1} \dots \pi_N} P(x_{i+1}, x_{i+2}, \dots, x_N, \pi_{i+1}, \dots, \pi_N \mid \pi_i = k)$$

$$= \sum_l \sum_{\pi_{i+1} \dots \pi_N} P(x_{i+1}, x_{i+2}, \dots, x_N, \pi_{i+1} = l, \pi_{i+2}, \dots, \pi_N \mid \pi_i = k)$$

$$= \sum_l e_l(x_{i+1}) a_{kl} \sum_{\pi_{i+1} \dots \pi_N} P(x_{i+2}, \dots, x_N, \pi_{i+2}, \dots, \pi_N \mid \pi_{i+1} = l)$$

$$= \sum_l e_l(x_{i+1}) a_{kl} \mathbf{b_l(i+1)}$$

Backward algorithm

We can compute $b_k(i)$ for all k, i , using dynamic programming

Initialization:

$$b_k(N) = 1, \text{ for all } k$$

Iteration:

$$b_k(i) = \sum_l e_l(x_{i+1}) a_{kl} b_l(i+1)$$

Termination:

$$P(x) = \sum_l a_{0l} e_l(x_1) b_l(1)$$

Computational complexity

What is the running time, and space required, for Forward and Backward?

Time: $O(K^2N)$

Space: $O(KN)$

Useful implementation technique to avoid underflows

Viterbi: sum of logs

Forward/Backward: rescaling at each few positions by multiplying by a constant

Posterior decoding

We can now calculate

$$P(\pi_i = k \mid x) = \frac{f_k(i) b_k(i)}{P(x)}$$

Then, we can ask

$$P(\pi_i = k \mid x) =$$

$$P(\pi_i = k, x) / P(x) =$$

$$P(x_1, \dots, x_i, \pi_i = k, x_{i+1}, \dots, x_n) / P(x) =$$

$$P(x_1, \dots, x_i, \pi_i = k) P(x_{i+1}, \dots, x_n \mid \pi_i = k) / P(x) =$$

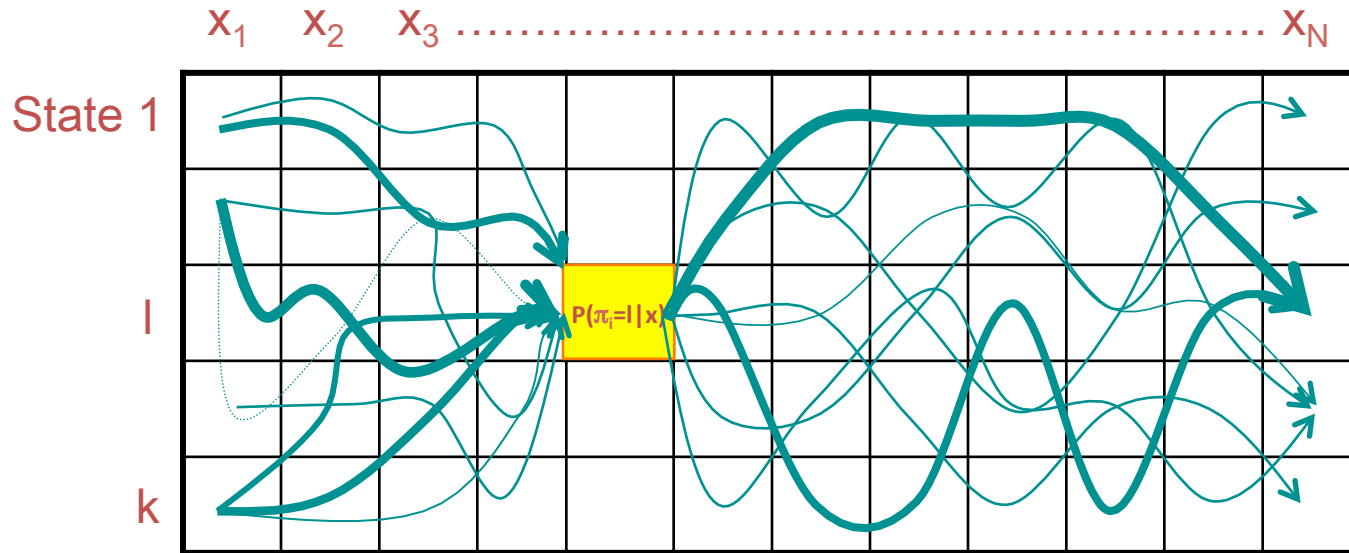
$$f_k(i) b_k(i) / P(x)$$

What is the most likely state at position i of sequence x :

Define π^\wedge by Posterior Decoding:

$$\pi^\wedge_i = \operatorname{argmax}_k P(\pi_i = k \mid x)$$

Posterior decoding



- $$P(\pi_i = k | x) = \sum_{\pi} P(\pi | x) \mathbf{1}(\pi_i = k)$$

$$= \sum_{\{\pi: \pi[i] = k\}} P(\pi | x)$$

$\mathbf{1}(\psi) = 1$, if ψ is true
0, otherwise

Viterbi / Forward / Backward

VITERBI

Initialization:

$$V_0(0) = 1$$

$$V_k(0) = 0, \text{ for all } k > 0$$

Iteration:

$$V_l(i) = e_l(x_i) \max_k V_k(i-1) a_{kl}$$

Termination:

$$P(x, \pi^*) = \max_k V_k(N)$$

FORWARD

Initialization:

$$f_0(0) = 1$$

$$f_k(0) = 0, \text{ for all } k > 0$$

Iteration:

$$f_l(i) = e_l(x_i) \sum_k f_k(i-1) a_{kl}$$

Termination:

$$P(x) = \sum_k f_k(N)$$

BACKWARD

Initialization:

$$b_k(N) = 1, \text{ for all } k$$

Iteration:

$$b_l(i) = \sum_k e_l(x_{i+1}) a_{kl} b_k(i+1)$$

Termination:

$$P(x) = \sum_k a_{0k} e_k(x_1) b_k(1)$$

Problem 3 - Learning

Find the parameters that maximize the likelihood of the observed (dataset of) sequence(s)

- “Easy” if we know the sequence of hidden states
 - Count # times each transition occurs
 - Count # times each observation occurs in each state
- Given an HMM and observed sequence, we can compute the distribution over paths, and therefore the expected counts
- “Chicken and egg” problem

Solution ?

- **Expectation-Maximization !**
- Guess initial HMM parameters
- **E step:** Compute distribution over paths
- **M step:** Compute max likelihood parameters

Baum-Welch algorithm

General idea

Transition probabilities

$$T_{ij} = P(s_i | s_j) = \frac{\text{Expected number of transitions from states } S_j \text{ to } S_i}{\text{Expected number of transitions out of state } S_j}$$

Emission probabilities

$$E_i(v_m) = P(v_m | s_i) = \frac{\text{Expected number of observing } V_m \text{ when in state } S_i}{\text{Expected number of times in state } S_i}$$

Initial probabilities

$$\pi_i = P(s_i) = \text{Expected frequency in state } S_i \text{ at time } k=1.$$