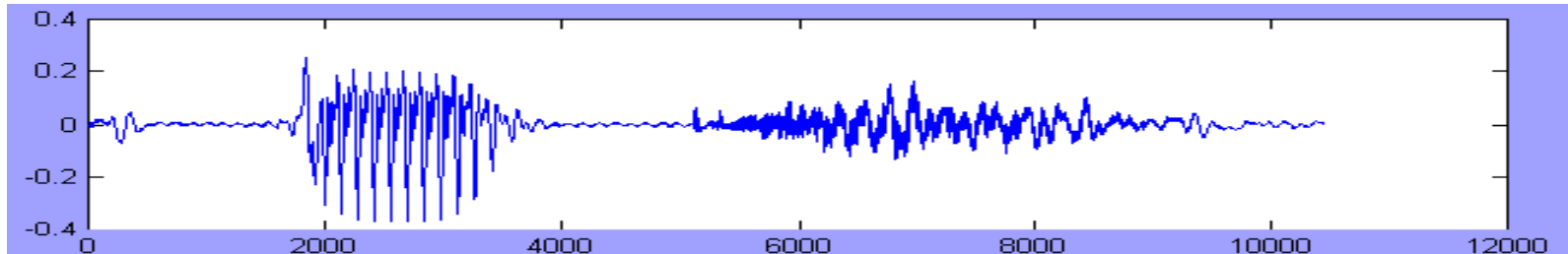


Hidden Markov Model

This lecture note was made based on the notes of
Prof. B.K.Shin(Pukyung Nat'l Univ) and Prof. Wilensky (UCB)

Sequential Data



서울시 중구 남구 망우동 신원 아파트 내의
부동산 가격

- ✓ Often highly variable, but has an embedded structure
- ✓ Information is contained in the structure

More examples

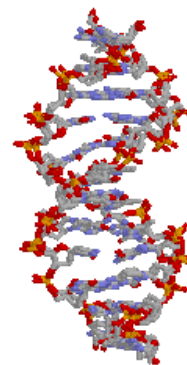
- Text, on-line handwriting, music notes, DNA sequence, program codes



Abstract Model checking is a formal verification technique that takes as input a requirement specification and a behavior model. We apply model checking to the verification of Actor-based models using an existing model checker SPIN. First, we propose an Actor-based modeling language Promela which is a modeling language for Actor-based systems.

```
main() { char q=34, n=10, *a="main() {
char q=34, n=10, *a=%c%s%c; printf(
a,q,a,q,n);}%c"; printf(a,q,a,n); }
```

DNA



```
AAAAGAAAAGCTTACAAAGATGAGACATGATAAAGGCTCCATTTG
AGCTTAGGTAATATGTTTTGCTATCCCTGTAGTTAAAAGTTTTTG
TCTTATTTTGAATACTGTGACTATTTCTTTACTATTAAATTTTC
CTTCTGTTTTCTCATCTAGGGAACCCCAAGAGCATCCAATAGAA
GCTGTGCAATTATGTAATAATTTTCAACTGTCTTCTCATAAATAAA
GAAGTATGTAATCTTTACCTGTATACAGTCCAGAGCCTTCTCAG
AAGCACAGAATATTTTATATTTCTTTATGTGAATTTTAAAGCT
GCAAACTCTGATGGCCTTAATTTCTTTTGCACACTGAAAGTTTTG
TAAAGAAATCATGTCCATACACTTTTGTGCAAGATGTGAATTAT
TGACACTGAACTTAATAACTGTGTACTGTTCGGAAGGGGTTCTCTC
AAATTTTTTGACTTTTTTTGTATGTGTGTTTTTCTTTTTTTTTTA
AGTTCTTATGAGGAGGGAGGCTAAATAAACCACTGTGCGTCTTGG
TGTAATTTGAAGATTGCCCATCTAGACTAGCAATCTCTTCATTA
TTCTCTGCTATATATAAAACGGTCTGTGAGGGAGGGGAAAAGCA
TTTTTCAATATATTGAACTTTTGTACTGAATTTTTTTGTAATAAG
CAATCAAGGTTATAATTTTTTTTAAATAGAAATTTTGAAGAAG
GCAATATTAACTTAATCACCATGTAAGCACTCTGGATGATGCATT
CCACAAAACCTTGGTTTTATGTTACTTCTTCTCTTAGATTCTTAA
TTCATGAGGAGGGTGGGGAGGGAGGCTGAGGGAGGGAAGGGTTT
CTCTATTAATAATGCAATTCGTTCTTTTTTTTAAAGATAGCTAACTT
GCTAAATTTCTTATGTGACATTAACAAATAAAAAAGCTCTTTTAA
TATTAGATAA
```

Example: Speech Recognition

- Given a sequence of inputs-features of some kind extracted by some hardware, guess the words to which the features correspond.
- Hard because features dependent on
 - Speaker, speed, noise, nearby features(“co-articulation” constraints), word boundaries
- “How to wreak a nice beach.”
- “How to recognize speech.”

Defining the problem

- Find $\operatorname{argmax}_{w \in L} P(w|y)$
 - y , a string of acoustic features of some form,
 - w , a string of words, from some fixed vocabulary
 - L , a language (defined as all possible strings in that language),
- Given some features, what is the most probable string the speaker uttered?

Analysis

- By Bayes' rule:

$$P(\mathbf{w}|\mathbf{y}) = P(\mathbf{w})P(\mathbf{y}|\mathbf{w})/P(\mathbf{y})$$

- Since \mathbf{y} is the same for different \mathbf{w} 's we might choose, the problem reduces to

$$\operatorname{argmax}_{\mathbf{w} \in L} P(\mathbf{w})P(\mathbf{y}|\mathbf{w})$$

- we need to be able to predict
 - each possible string in our language
 - pronunciation, given an utterance.

$P(w)$ where w is an utterance

- Problem: There are a very large number of possible utterances!
- Indeed, we create new utterances all the time, so we cannot hope to have their probabilities.
- So, we will need to make some independent assumptions.
- First attempt: Assume that words are uttered independently of one another.
 - Then $P(w)$ becomes $P(w_1) \dots P(w_n)$, where w_i are the individual words in the string.
 - Easy to estimate these numbers-count the relative frequency of words in the language.

Assumptions

- However, assumption of independence is pretty bad.
 - Words don't just follow each other randomly
- Second attempt: Assume each word depends only on the previous word.
 - E.g., "the" is more likely to be followed by "ball" than by "a",
 - despite the fact that "a" would otherwise be a very common, and hence, highly probably word.
- Of course, this is still not a great assumption, but it may be a decent approximation

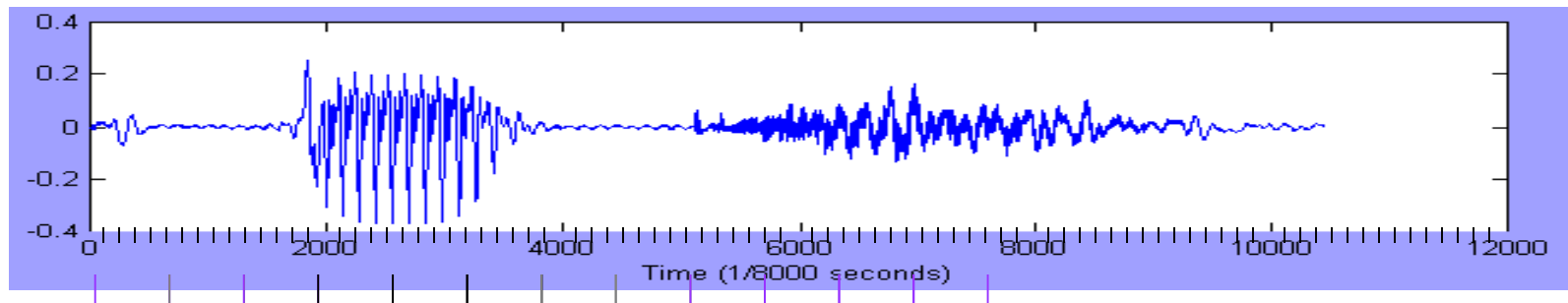
In General

- This is typical of lots of problems, in which
 - we view the probability of some event as dependent on potentially many past events,
 - of which there too many actual dependencies to deal with.
- So we simplify by making assumption that
 - Each event depends only on previous event, and
 - it doesn't make any difference when these events happen
 - in the sequence.

Speech Example

- Representation

$$\begin{aligned} - \mathbf{X} &= \mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_3 \mathbf{x}_4 \mathbf{x}_5 \dots \mathbf{x}_{T-1} \mathbf{x}_T \\ &= s \phi p \text{ iy iy iy } \phi \phi \text{ ch ch ch ch} \end{aligned}$$



Analysis Methods

- Probability-based analysis?

$$P(s \phi p iy iy iy \phi \phi ch ch ch ch) = ?$$

- Method I

$$P(s)P(\phi)^3 P(p)P(iy)^3 P(ch)^4$$

- Observations are independent; no time/order
- A poor model for temporal structure
 - Model size = $|V| = N$

Analysis methods

- Method II

$$P(s)P(s | s)P(\phi | s)P(p | \phi)P(iy | p)P(iy | iy)^2 \\ \times P(\phi | iy)P(\phi | \phi)P(ch | \phi)P(ch | ch)^2$$

- A simple model of ordered sequence

- A symbol is dependent only on the immediately preceding:

$$P(x_t | x_1 x_2 x_3 \cdots x_{t-1}) = P(x_t | x_{t-1})$$

- $|V| \times |V|$ matrix model

- 50×50 – not very bad ...
- $10^5 \times 10^5$ – doubly outrageous!!

Another analysis method

- Method III

- What you see is a clue to what lies behind and is not known a priori
 - The source that generated the observation
 - The source evolves and generates characteristic observation sequences

$$q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow \cdots \rightarrow q_T$$

$$P(s, q_1)P(s, q_2 | q_1)P(\phi, q_3 | q_2) \cdots P(\text{ch}, q_T | q_{T-1}) = \prod_t P(x_t, q_t | q_{t-1})$$
$$\sum_Q P(s, q_1)P(s, q_2 | q_1)P(\phi, q_3 | q_2) \cdots P(\text{ch}, q_T | q_{T-1}) = \sum_Q \prod_t P(x_t, q_t | q_{t-1})$$

More Formally

- We want to know $P(w_1, \dots, w_{n-1}, w_n)$.

- To clarify, let's write the sequence this way:

$$P(q_1=S_i, q_2=S_j, \dots, q_{n-1}=S_k, q_n=S_i)$$

Here the q_l indicate the l -th position of the sequence,
and the S_i the possible different words from our
vocabulary.

- E.g., if the string were “The girl saw the boy”, we might have

$$S_1 = \text{the} \quad q_1 = S_1$$

$$S_2 = \text{girl} \quad q_2 = S_2$$

$$S_3 = \text{saw} \quad q_3 = S_3$$

$$S_4 = \text{boy} \quad q_4 = S_1$$

$$q_5 = S_4$$

Formalization (continue)

- We want $P(q_1=S_i, q_2=S_j, \dots, q_{n-1}=S_k, q_n=S_i)$
- Let's break this down as we usually break down a joint :
- $= P(q_n=S_i \mid q_1=S_j, \dots, q_{n-1}=S_k) \times P(q_1=S_j, \dots, q_{n-1}=S_k)$
- ...
- $= P(q_n=S_i \mid q_1=S_j, \dots, q_{n-1}=S_k) \times P(q_{n-1}=S_k \mid q_1=S_j, \dots, q_{n-1}=S_m) \times P(q_2=S_j \mid q_1=S_j) \times P(q_1=S_i)$
- Our simplifying assumption is that each event is only
- dependent on the previous event, and that we don't care when
- the events happen, I.e.,
- $P(q_i=S_i \mid q_1=S_j, \dots, q_{i-1}=S_k) \times P(q_i=S_i \mid q_{i-1}=S_k)$ and
- $P(q_i=S_i \mid q_{i-1}=S_k) = P(q_j=S_i \mid q_{j-1}=S_k)$
- This is called the ***Markov assumption***.

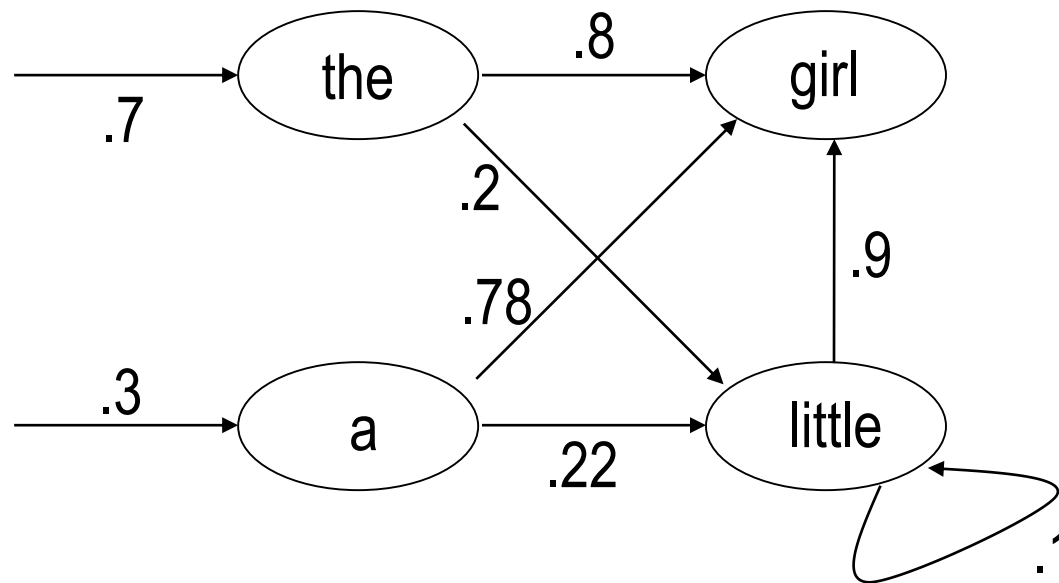
Markov Assumption

- “The future does not depend on the past, given the present.”
- Sometimes this is called the first-order Markov assumption.
- second-order assumption would mean that each event depends on the previous two events.
 - This isn’t really a crucial distinction.
 - What’s crucial is that there is some limit on how far we are willing to look back.

Markov Models

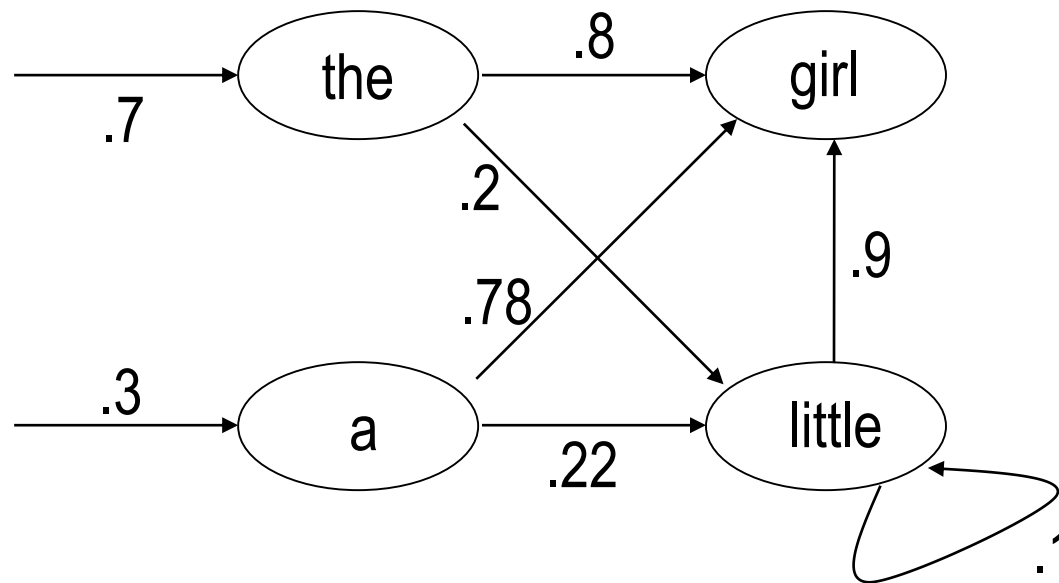
- The Markov assumption means that there is only one probability to remember for each event type (e.g., word) to another event type.
- Plus the probabilities of starting with a particular event.
- This lets us define a Markov model as:
 - finite state automaton in which
 - the states represent possible event types(e.g., the different words in our example)
 - the transitions represent the probability of one event type following another.
- It is easy to depict a Markov model as a graph.

Example: A Markov Model for a Tiny Fragment of English



- Numbers on arrows between nodes are “transition” probabilities, e.g., $P(q_i = \text{girl} | q_{i-1} = \text{the}) = .8$
- The numbers on the initial arrows show the probability of starting in the given state.
- Missing probabilities are assumed to be 0.

Example: A Markov Model for a Tiny Fragment of English



- Generates/recognizes a tiny(but infinite!) language, along with probabilities :

$$P(\text{"The little girl"}) = .7 \times .2 \times .9 = .126$$

$$P(\text{"A little little girl"}) = .3 \times .22 \times .1 \times .9 = .00594$$

Example(con't)

- $P(\text{"The little girl"})$ is really shorthand for

$$P(q_1=\text{the}, q_2=\text{little}, q_3=\text{girl})$$

where q_1 , q_2 , and q_3 are states.

- We can easily answer other questions, e.g.:

“Given that sentence begins with “a”, what is the probability that the next words were “little girl”?”

$$P(q_3=\text{the}, q_2=\text{little}, q_1=\text{a})$$

$$= P(q_3=\text{girl} \mid q_2=\text{little}, q_1=\text{a})P(q_2=\text{little} \mid q_1=\text{a})$$

$$= P(q_3=\text{girl} \mid q_2=\text{little})P(q_2=\text{little} \mid q_1=\text{a})$$

$$= .9 \times .22 = .198$$

Markov Models and Graphical Models

- Markov models and Belief Networks can both be represented by nice graphs.
- Do the graphs mean the same thing?
 - No! In the graphs for Markov models; nodes do *not* represent random variables, CPTs.
- Suppose we wanted to encode the same information via a belief network.
 - We would have to “unroll” it into a sequence of nodes-as many as there are elements in the sequence-each dependent on the previous, each with the same CPT.
 - This redrawing is valid, and sometimes useful, but doesn’t explicitly represent useful facts, such as that the CPTs are the same everywhere.

Back to the Speech Recognition Problem

- A Markov model for all of English would have one node for each word, which would be connected to the node for each word that can follow it.
- Without Loss Of Generality, we could have connections from every node to every node, some of which have transition probability 0.
- Such a model is sometimes called a bigram model.
 - This is equivalent to knowing the probability distribution of pair of words in sequences (and the probability distribution for individual words).
 - A bigram model is an example of a language model, i.e., some (in this case, extremely simply) view of what sentences or sequences are likely to be seen.

Bigram Model

- Bigram models are rather inaccurate language models.
 - E.g., the word after “a” is much more likely to be “missile” if the word preceding “a” is “launch”.
- the Markov assumption is pretty bad.
- If we could condition on a few previous words, life gets a bit better:
 - E.g., we could predict “missile” is more likely to follow “launch a” than “saw a”.
- This would require a “second order” Markov model.

Higher-Order Models

- In the case of words, this is equivalent to going to trigrams.
- Fundamentally, this isn't a big difference:
 - We can convert a second order model into a first order model, but with a lot more states.
 - And we would need much more data!
- Note, though, that a second-order model still couldn't accurately predict what follows "launch a large"
 - i.e., we are predicting the next word based on only the two previous words, so the useful information before "a large" is lost.
- Nevertheless, such language models are very useful approximations.

Back to Our Spoken Sentence recognition Problem

- We are trying to find

$$\operatorname{argmax}_{\mathbf{w} \in L} P(\mathbf{w})P(\mathbf{y}|\mathbf{w})$$

- We just discussed estimating $P(\mathbf{w})$.
- Now let's look at $P(\mathbf{y}|\mathbf{w})$.
 - That is, how do we pronounce a sequence of words?
 - Can make the simplification that how we pronounce words is independent of one another.

$$P(\mathbf{y}|\mathbf{w}) = \sum P(o_1=v_i, o_2=v_j, \dots, o_k=v_l | w_1) \times \\ \dots \times P(o_{x-m}=v_p, o_{x-m+1}=v_q, \dots, o_x=v_r | w_n)$$

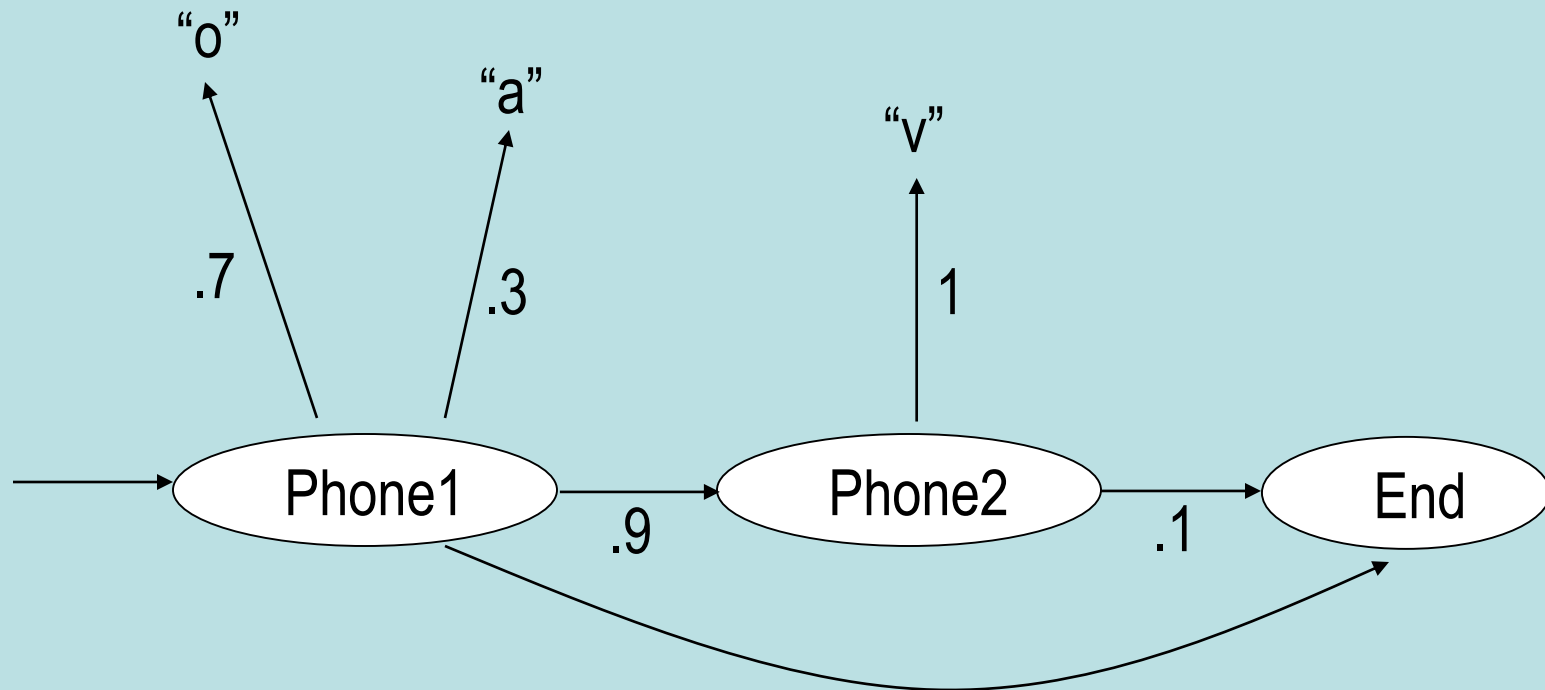
i.e., each word produces some of the sounds with some probability; we have to sum over possible different word boundaries.

- So, what we need is model of how we pronounce individual words.

A Model

- Assume there are some underlying states, called “phones”, say, that get pronounced in slightly different ways.
- We can represent this idea by complicating the Markov model:
 - Let's add probabilistic *emissions* of outputs from each state.

Example: A (Simplistic) Model for Pronouncing “of”



- Each state can emit a different sound, with some probability.
- Variant: Have the emissions on the transitions, rather than the states.

How the Model Works

- We see outputs, e.g., “o v”.
- We can’t “see” the actual state transitions.
- But we can infer possible underlying transitions from the observations, and then assign a probability to them
- E.g., from “o v”, we infer the transition “phone1 phone2”
 - with probability $.7 \times .9 = .63$.
- I.e., the probability that the word “of” would be pronounced as “o v” is 63%.

Hidden Markov Models

- This is a “hidden Markov model”, or HMM.
- Like (fully observable) Markov models, transitions from one state to another are independent of everything else.
- Also, the emission of an output from a state depends only on that state, i.e.:

$$\begin{aligned} P(\mathbf{O}|\mathbf{Q}) &= P(o_1, o_2, \dots, o_n | q_1, \dots, q_n) \\ &= P(o_1 | q_1) \times P(o_2 | q_2) \times \dots \times P(o_n | q_n) \end{aligned}$$

HMMs Assign Probabilities to Sequences

- We want to know how probable a sequence of observations is given an HMM.
- This is slightly complicated because there might be multiple ways to produce the observed output.
- So, we have to consider all possible ways an output might be produced, i.e., for a given HMM:

$$P(\mathbf{O}) = \sum_{\mathbf{Q}} P(\mathbf{O}|\mathbf{Q})P(\mathbf{Q})$$

where \mathbf{O} is a given output sequence, and \mathbf{Q} ranges over all possible sequence of states in the model.

- $P(\mathbf{Q})$ is computed as for (visible) Markov models.

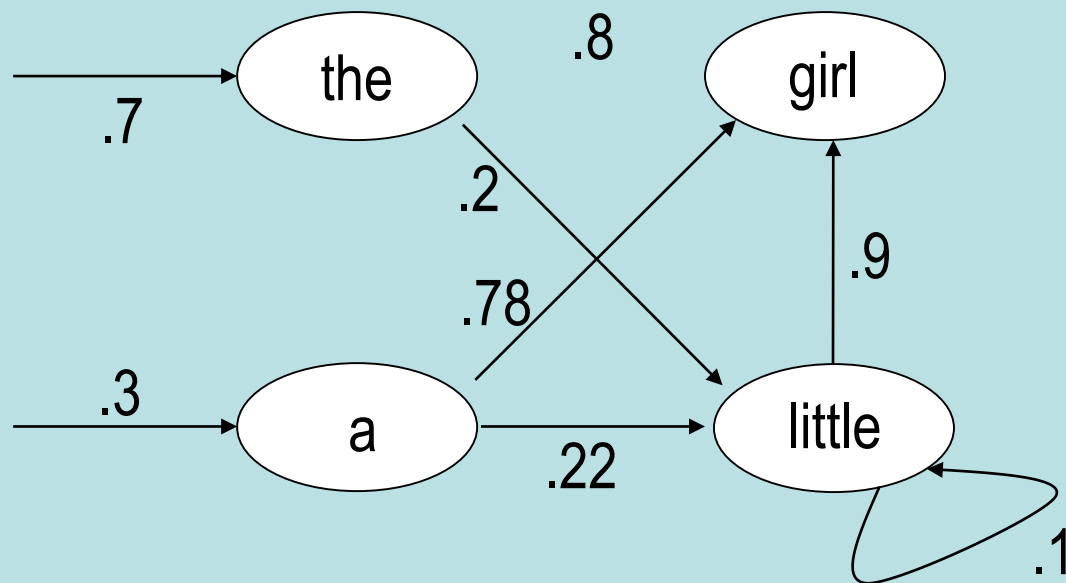
$$\begin{aligned} P(\mathbf{O}|\mathbf{Q}) &= P(o_1, o_2, \dots, o_n | q_1, \dots, q_n) \\ &= P(o_1 | q_1) \times P(o_2 | q_2) \times \dots \times P(o_n | q_n) \end{aligned}$$

- We'll look at computing this efficiently in a while...

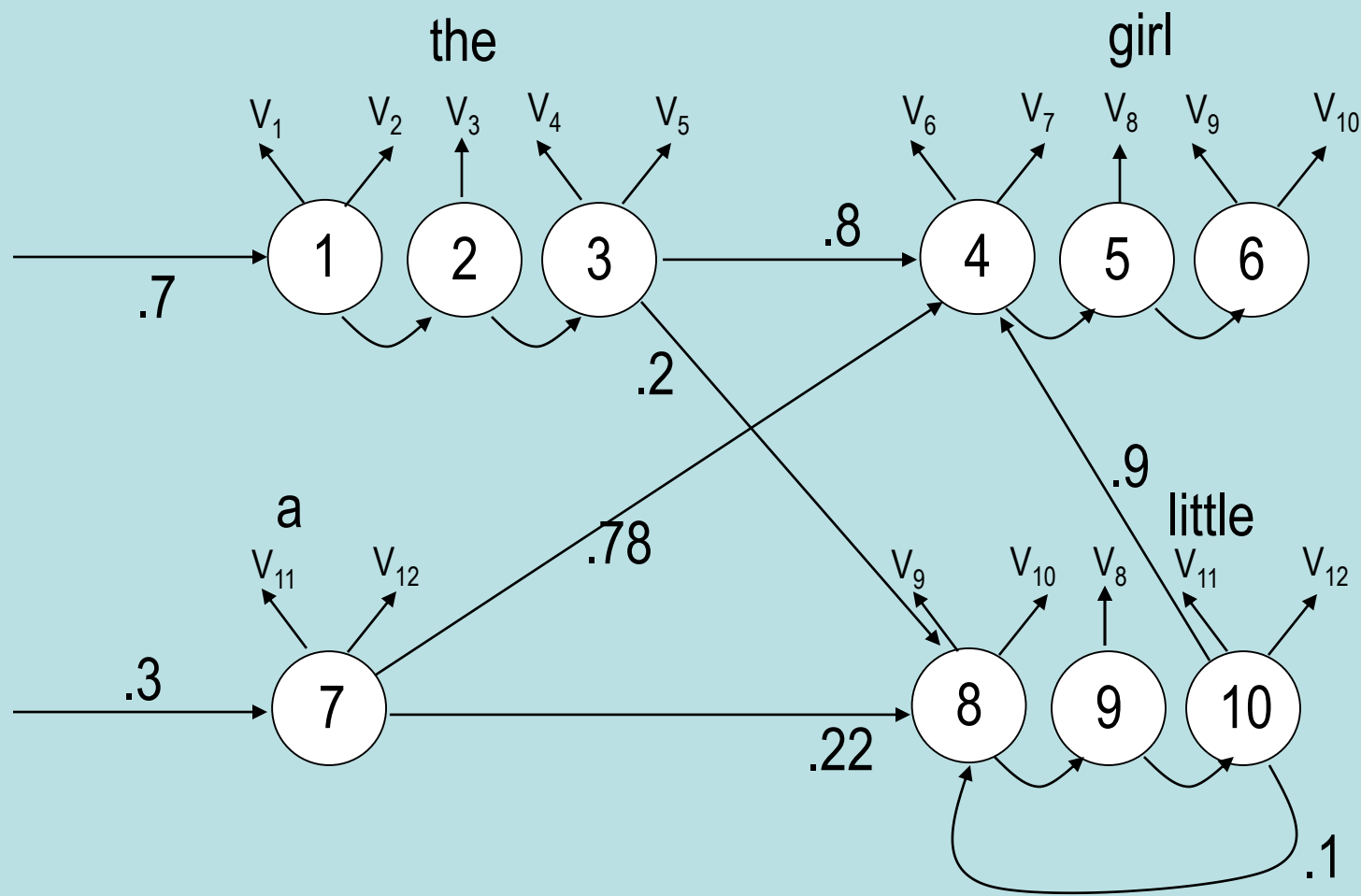
Finishing Solving the Speech Problem

- To find $\text{argmax}_{\mathbf{w} \in L} P(\mathbf{w})P(\mathbf{y}|\mathbf{w})$, just consider these probabilities over all possible strings of words.
- Could “splice in” each word in language model with its HMM pronunciation model to get one big HMM.
 - Lets us incorporate more dependencies.
 - E.g., could have two models of “of”, one of which has a much higher probability of transitioning to words beginning with consonants.
- Real speech systems have another level in which phonemes are broken up into acoustic vectors.
 - but these are also HMMs.
 - So we can make one gigantic HMM out of the whole thing.
- So, given \mathbf{y} , all we need is to find most probable path through the model that generates it.

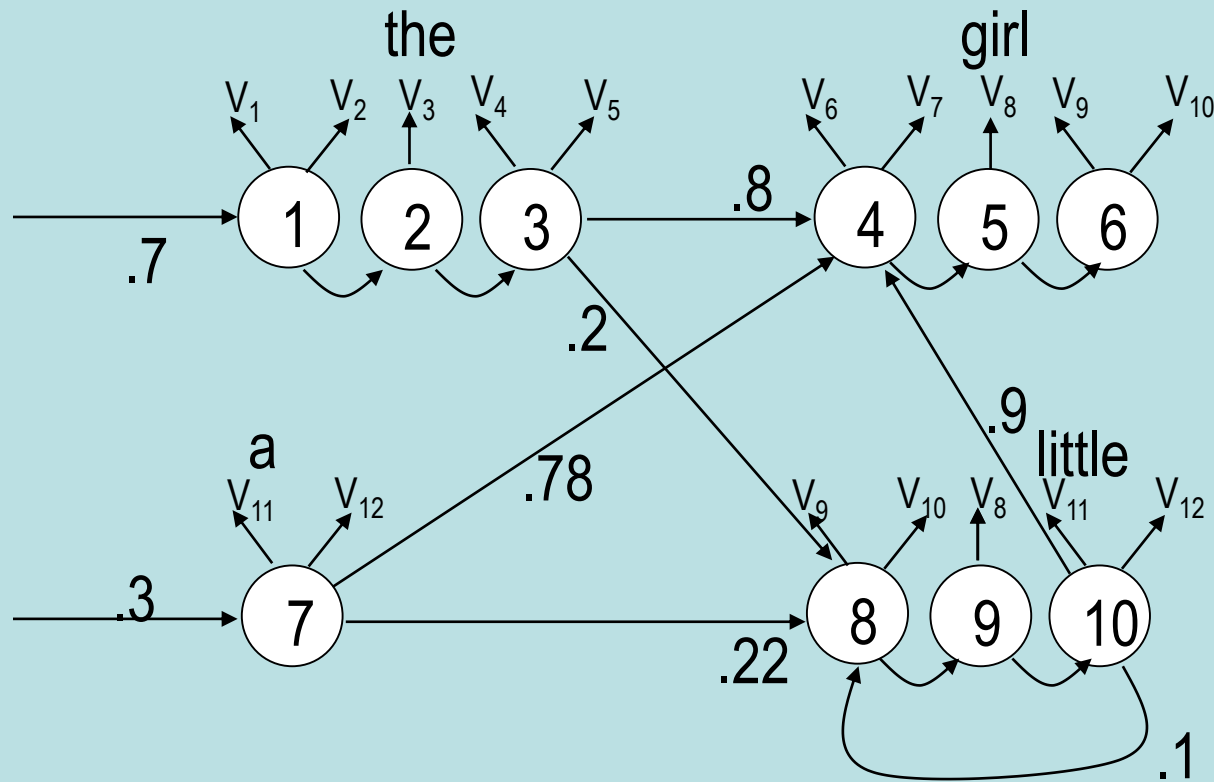
Example: Our Word Markov Model



Example: Splicing in Pronunciation HMMs



Example: Best Sequence



- Suppose observation is " $v_1 v_3 v_4 v_9 v_8 v_{11} v_7 v_8 v_{10}$ "
- Suppose most probable sequence is determined to be "1,2,3,8,9,10,4,5,6" (happens to be only way in example)
- Then interpretation is "the little girl".

Hidden Markov Models

- Modeling sequences of events
- Might want to
 - Determine the probability of a give sequence
 - Determine the probability of a model producing a sequence in a particular way
 - equivalent to recognizing or interpreting that sequence
 - Learning a model form some observations.

Why HMM?

- Because the HMM is a very good model for such patterns!
 - highly variable spatiotemporal data sequence
 - often unclear, uncertain, and incomplete
- Because it is very successful in many applications!
- Because it is quite easy to use!
 - Tools already exist...

The problem

- “What you see is the truth”
 - Not quite a valid assumption
 - There are often errors or noise
 - Noisy sound, sloppy handwriting, ungrammatical or Kornglish sentence
 - There may be some truth process
 - Underlying hidden sequence
 - Obscured by the incomplete observation

The Auxiliary Variable

$$q_t \in S = \{1, \dots, N\}$$

- N is also conjectured
- $\{q_t: t \geq 0\}$ is conjectured, not visible
 - nor is $Q = q_1 q_2 \cdots q_T$
 - is Markovian

$$P(q_1 q_2 \cdots q_T) = P(q_1) P(q_2 | q_1) \cdots P(q_T | q_{T-1})$$

- “Markov chain”

Summary of the Concept


$$\begin{aligned} P(X) &= \sum_Q P(X, Q) \\ &= \sum_Q P(Q) P(X | Q) \\ &= \sum_Q P(q_1 q_2 \cdots q_T) P(x_1 x_2 \cdots x_T | q_1 q_2 \cdots q_T) \\ &= \sum_Q \underbrace{\prod_{t=1}^T P(q_t | q_{t-1})}_{\text{Markov chain process}} \underbrace{\prod_{t=1}^T p(x_t | q_t)}_{\text{Output process}} \end{aligned}$$

Hidden Markov Model

- is a doubly stochastic process
 - stochastic chain process : $\{ q(t) \}$
 - output process : $\{ f(x|q) \}$
- is also called as
 - *Hidden Markov chain*
 - *Probabilistic function of Markov chain*

HMM Characterization

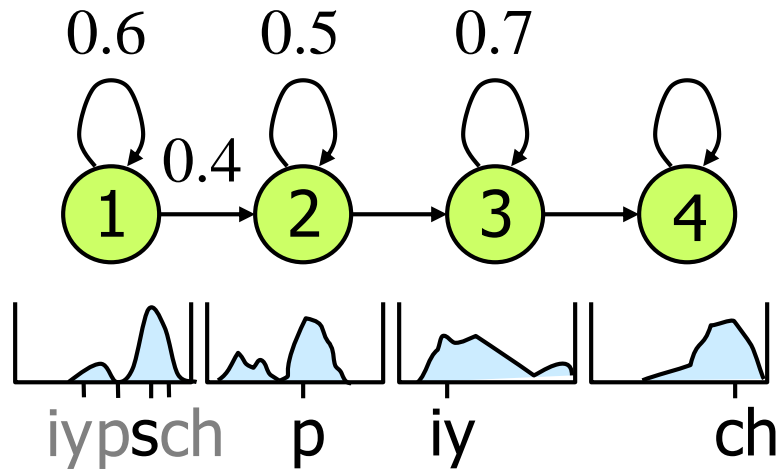
- $\lambda = (A, B, \pi)$
 - A : state transition probability
 $\{ a_{ij} \mid a_{ij} = p(q_{t+1}=j|q_t=i) \}$
 - B : symbol output/observation probability
 $\{ b_j(v) \mid b_j(v) = p(x=v|q_t=j) \}$
 - π : initial state distribution probability
 $\{ \pi_i \mid \pi_i = p(q_1=i) \}$


$$\sum_Q P(Q \mid \lambda) P(\mathbf{X} \mid Q, \lambda)$$
$$= \sum_Q \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \dots a_{q_{T-1} q_T} b_{q_1}(x_1) b_{q_2}(x_2) \dots b_{q_T}(x_T) \Big|_{\lambda}$$

HMM, Formally

- A set of states $\{S_1, \dots, S_N\}$
 - q_t denotes the state at time t .
- A *transition probability matrix* A , such that
$$A[i,j] = a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$$
 - This is an $N \times N$ matrix.
- A set of symbols, $\{v_1, \dots, v_M\}$
 - For all purposes, these might as well just be $\{1, \dots, M\}$
 - o_t denotes the observation at time t .
- A observation symbol probability distribution matrix B , such that
$$B[i,j] = b_{i,j} = P(o_t = v_j | q_t = S_i)$$
 - This is a $N \times M$ matrix.
- An *initial state distribution*, π , such that $\pi_i = P(q_1 = S_i)$
- For convenience, call the entire model $\lambda = (A, B, \pi)$
 - Note that N and M are important implicit parameters.

Graphical Example



$$\pi = [1.0 \quad 0 \quad 0 \quad 0]$$

$$A = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0.6 & 0.4 & 0.0 & 0.0 \\ 0.0 & 0.5 & 0.5 & 0.0 \\ 0.0 & 0.0 & 0.7 & 0.3 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \end{matrix}$$

$$B = \begin{matrix} & \begin{matrix} \text{ch} & \text{iy} & \text{p} & \text{s} \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0.2 & 0.2 & 0.0 & 0.6 & \dots \\ 0.0 & 0.2 & 0.5 & 0.3 & \dots \\ 0.0 & 0.8 & 0.1 & 0.1 & \dots \\ 0.6 & 0.0 & 0.2 & 0.2 & \dots \end{bmatrix} \end{matrix}$$

Data interpretation

$$\begin{aligned}
 &P(\text{s s p p i y i y i y c h c h c h}|\lambda) \\
 &= \sum_Q P(\text{ssppiiyiychchch}, Q|\lambda) \\
 &= \sum_Q \underbrace{P(Q|\lambda) p(\text{ssppiiyiychchch}|Q, \lambda)}
 \end{aligned}$$

0.6	0.4	0.0	0.0
0.0	0.5	0.5	0.0
0.0	0.0	0.7	0.3
0.0	0.0	0.0	1.0

Let $Q = 1\ 1\ 2\ 2\ 3\ 3\ 3\ 4\ 4\ 4$

0.2	0.2	0.0	0.6	...
0.0	0.2	0.5	0.3	...
0.0	0.8	0.1	0.1	...
0.6	0.0	0.2	0.2	...

$$\begin{aligned}
 &P(Q|\lambda) p(\text{ssppiiyiychchch}|Q, \lambda) \\
 &= P(1122333444|\lambda) p(\text{ssppiiyiychchch}|1122333444, \lambda) \\
 &= P(1|\lambda)P(s|1,\lambda) P(1|1, \lambda)P(s|1,\lambda) P(2|1, \lambda)P(p|2,\lambda) \\
 &\quad P(2|2, \lambda)P(p|2,\lambda) \dots \\
 &= (1 \times .6) \times (.6 \times .6) \times (.4 \times .5) \times (.5 \times .5) \times (.5 \times .8) \times (.7 \times .8)^2 \\
 &\quad \times (.3 \times .6) \times (1. \times .6)^2 \\
 &\cong 0.0000878
 \end{aligned}$$

#multiplications $\sim 2TN^T$

Issues in HMM

- Intuitive decisions
 1. number of states (N)
 2. topology (state inter-connection)
 3. number of observation symbols (V)
- Difficult problems
 4. efficient computation methods
 5. probability parameters (λ)

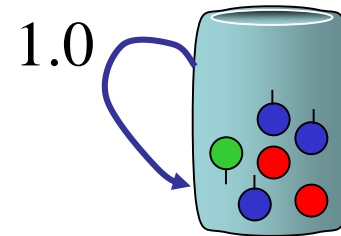
The Number of States

- How many states?
 - Model size
 - Model topology/structure
- Factors
 - Pattern complexity/length and variability
 - The number of samples
- Ex: r r g b b g b b b r
 ● ● ● ● ● ● ● ● ● ●
 | | | | | | | | | |
 | | | | | | | | | |

(1) The simplest model

- Model I

- $N = 1$
- $a_{11} = 1.0$
- $B \cong [1/3, 1/6, 1/2]$



$$P(r r g b b g b b b r | \lambda_1) = 1 \times \frac{1}{3} \times 1 \times \frac{1}{3} \times 1 \times \frac{1}{6} \times 1 \times \frac{1}{2} \times 1 \times \frac{1}{2} \times 1 \times \frac{1}{6} \\ \times 1 \times \frac{1}{2} \times 1 \times \frac{1}{2} \times 1 \times \frac{1}{2} \times 1 \times \frac{1}{3}$$

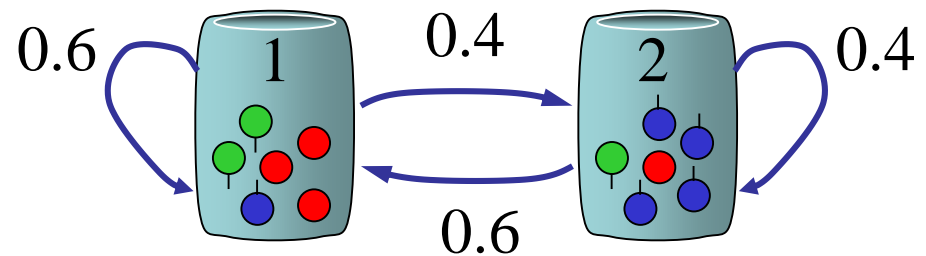
(2) Two state model

- Model II:

- $N = 2$

$$A = \begin{bmatrix} 0.6 & 0.4 \\ 0.6 & 0.4 \end{bmatrix}$$

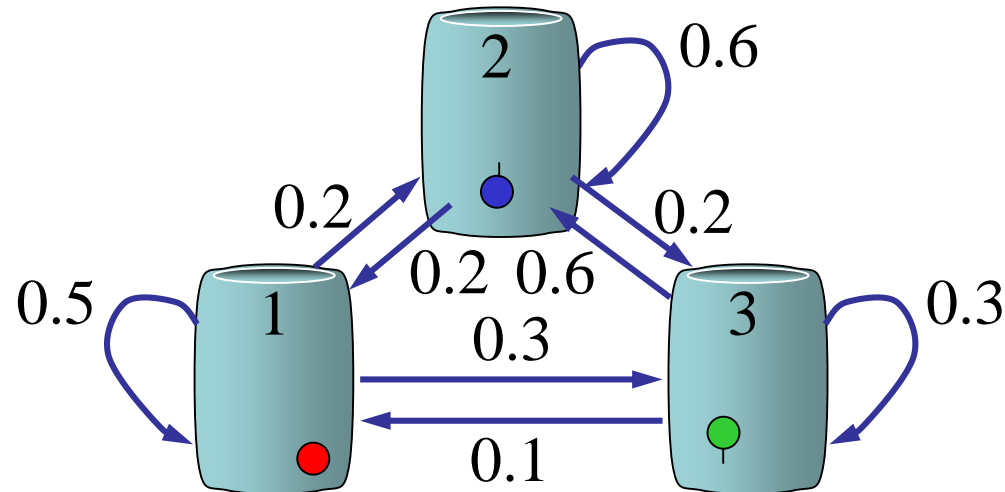
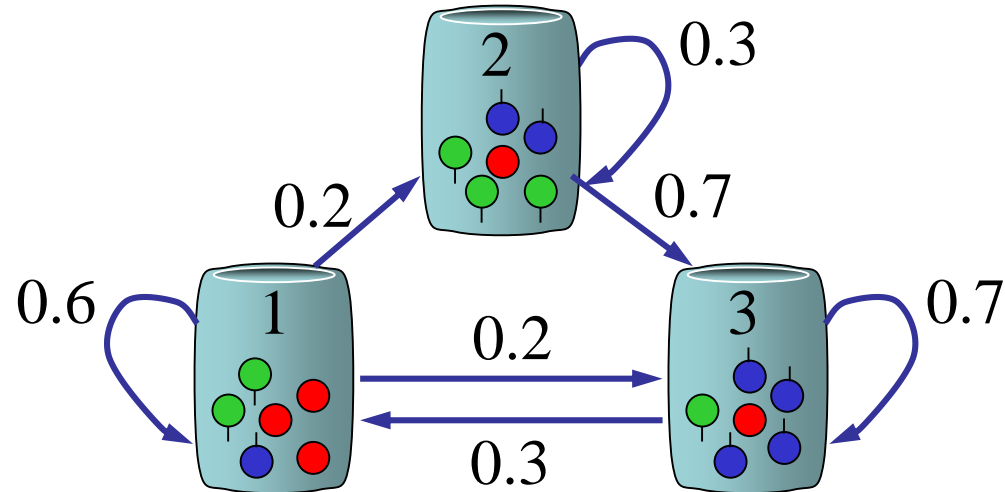
$$B = \begin{bmatrix} 1/2 & 1/3 & 1/6 \\ 1/6 & 1/6 & 2/3 \end{bmatrix}$$



$$\begin{aligned}
 P(\text{r r g b b g b b b r} | \lambda_1) &= .5 \times \frac{1}{2} \times .6 \times \frac{1}{2} \times .6 \times \frac{1}{3} \times .4 \times \frac{4}{6} \times .4 \times \frac{4}{6} \times .6 \times \frac{1}{3} \\
 &\quad \times .4 \times \frac{4}{6} \times .4 \times \frac{4}{6} \times .4 \times \frac{4}{6} \times .6 \times \frac{1}{2} + \dots \\
 &= ?
 \end{aligned}$$

(3) Three state models

- $N=3$:



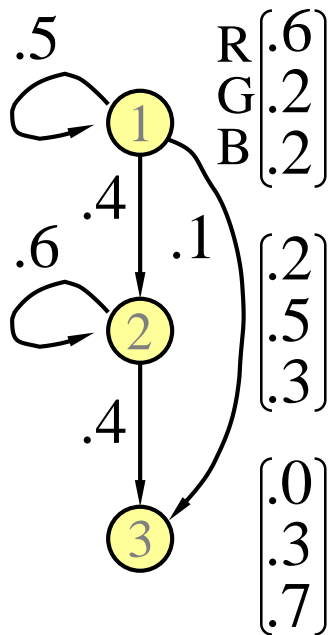
The Criterion is

- Obtaining the best model(λ) that maximizes

$$P(X \mid \hat{\lambda})$$

- The best topology comes from insight and experience
 - ← the # classes/symbols/samples

A trained HMM



$$\pi = \begin{bmatrix} 1. & 0. & 0. \end{bmatrix}$$

$$A = \begin{matrix} & \begin{matrix} 1 & 2 & 3 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & \begin{bmatrix} .5 & .4 & .1 \\ .0 & .6 & .4 \\ .0 & .0 & .0 \end{bmatrix} \end{matrix}$$

$$B = \begin{matrix} & \begin{matrix} R & G & B \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & \begin{bmatrix} .6 & .2 & .2 \\ .2 & .5 & .3 \\ .0 & .3 & .7 \end{bmatrix} \end{matrix}$$

Three Problems

1. Model evaluation problem
 - What is the probability of the observation?
 - Given an observed sequence and an HMM, how *probable* is that sequence?
 - Forward algorithm
2. Path decoding problem
 - What is the best state sequence for the observation?
 - Given an observed sequence and an HMM, what is *the most likely* state *sequence* that generated it?
 - Viterbi algorithm
3. Model training problem
 - How to estimate the model parameters?
 - Given an observation, can we *learn* an HMM for it?
 - Baum-Welch reestimation algorithm

Problem: How to Compute $P(\mathbf{O}|\lambda)$

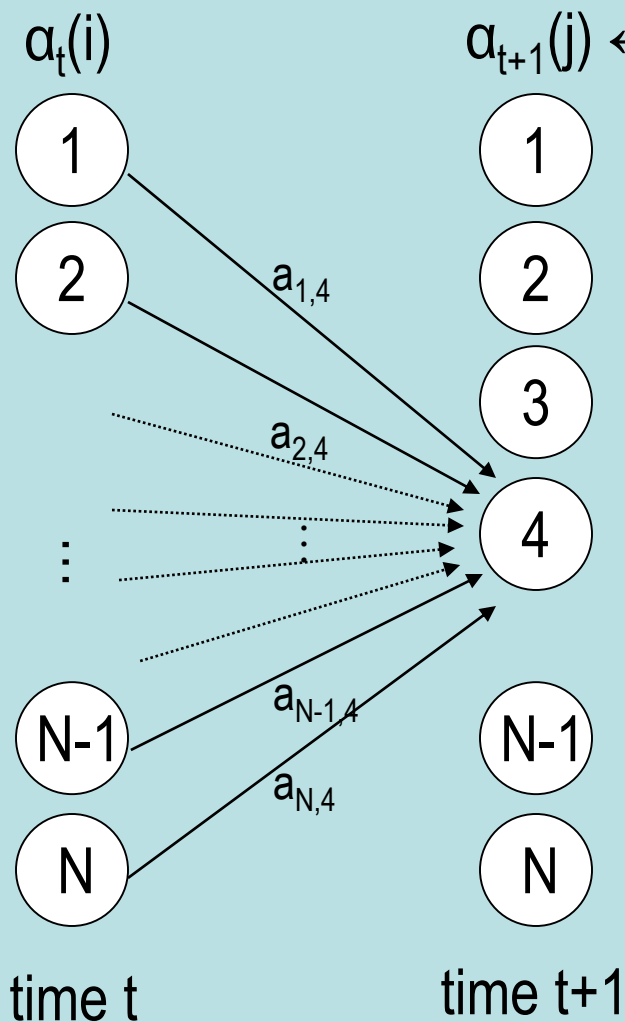
$$P(\mathbf{O}|\lambda) = \sum_{\text{all } \mathbf{Q}} P(\mathbf{O}|\mathbf{Q}, \lambda) P(\mathbf{Q}|\lambda)$$

where \mathbf{O} is an observation sequence and \mathbf{Q} is a sequence of states.

$$= \sum_{q_1, q_2, \dots, q_T} \mathbf{b}_{q_1, o_1} \mathbf{b}_{q_2, o_2} \dots \mathbf{b}_{q_T, o_T} \boldsymbol{\pi}_{q_1} \mathbf{a}_{q_1, q_2} \dots \mathbf{a}_{q_{T-1}, q_T}$$

- So, we could just enumerate all possible sequences through the model, and sum up their probabilities.
- Naively, this would involve $O(TN^T)$ operations:
 - At every step, there are N possible transitions, and there are T steps.
- However, we can take advantage of the fact that, at any given time, we can only be in one of N states, so we only have to keep track of paths to each state.

Basic Idea for $P(\mathbf{O}|\lambda)$ Algorithm



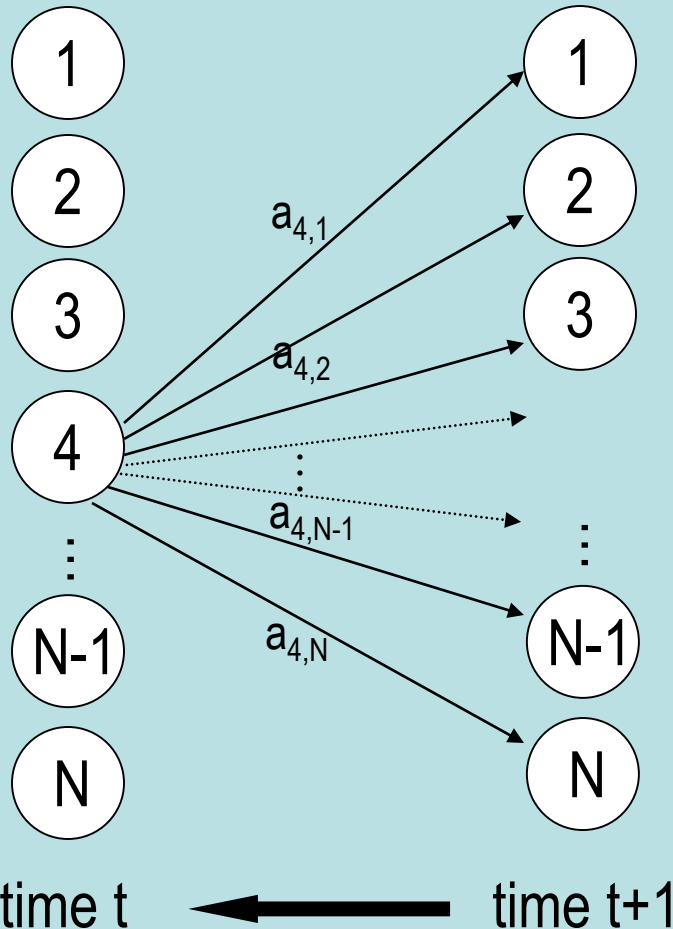
If we know the probability of being in each state at time t , and producing the observation so far ($\alpha_t(i)$), then the probability of being in a given state at the next clock tick, and then emitting the next output, is easy to compute.

A Better Algorithm For Computing $P(\mathbf{O}|\lambda)$

- Let $\alpha_t(i)$ be defined as $P(o_1 o_2 \dots o_t, q_t = S_i | \lambda)$.
 - I.e., the probability of seeing a prefix of the observation, and ending in a particular state.
- Algorithm:
 - Initialization: $\alpha_1(i) \leftarrow \pi_i b_{i,o} \quad 1 \leq i \leq N$
 - Induction: $\alpha_{t+1}(j) \leftarrow (\sum_{1 \leq i \leq N} \alpha_t(i) a_{ij}) b_{j,o_{t+1}}$
 $1 \leq t \leq T-1, 1 \leq j \leq N$
 - Finish: Return $\sum_{1 \leq i \leq N} \alpha_T(i)$
- This is called the forward procedure.
- How efficient is it?
 - At each time step, do $O(N)$ multiplications and additions for each of N nodes, or $O(N^2 T)$.

Can We Do it Backwards?

$$\beta_t(i) = \sum_{1 \leq j \leq N} a_{ij} b_{j, o_{t+1}} \beta_{t+1}(j)$$



$\beta_t(i)$ is probability that, given we are in state i at time t we produce $o_{t+1} \dots o_T$. If we know probability of outputting the tail of the observation, given that we are in some state, we can compute probability that, given we are in some state at the previous clock tick, we emit the (one symbol bigger) tail.

Going Backwards Instead

- Define $\beta_t(i)$ as $P(o_{t+1}o_{t+2}\dots o_T | q_t = S_i, \lambda)$.
 - I.e., the probability of seeing a *tail* of the observation, given that we were in a particular state.
- Algorithm:
 - Initialization: $\beta_T(i) \leftarrow 1, 1 \leq i \leq N$
 - Induction: $\beta_t(i) \leftarrow \sum_{1 \leq j \leq N} a_{ij} b_{j, o_{t+1}} \beta_{t+1}(j)$
 $T-1 \geq t \geq 1, 1 \leq i \leq N$
- This is called the backward procedure.
- We could use this to compute $P(\mathbf{O} | \lambda)$ too ($\beta_1(i)$ is $P(o_2 o_3 \dots o_T | q_1 = S_i, \lambda)$, so $P(\mathbf{O} | \lambda) = \sum_{1 \leq j \leq N} \pi_j b_{j, o_1} \beta_1(j)$).
 - But nobody does this.
 - Instead, we will have another use for it soon.
- How efficient is it?
 - Same as forward procedure.

1. Model Evaluation

- Solution: forward/backward procedure
 - Define: **forward probability** -> FW procedure
$$\alpha_t(j) = P(x_1 \cdots x_t, q_t = j \mid \lambda)$$
 - Define: **backward probability** -> BW procedure
$$\beta_t(i) = P(q_t = i, x_{t+1} \cdots x_T, q_t = j \mid \lambda)$$
- These are probabilities of the partial events leading to/from a point in space-time

Forward procedure

- Initialization:

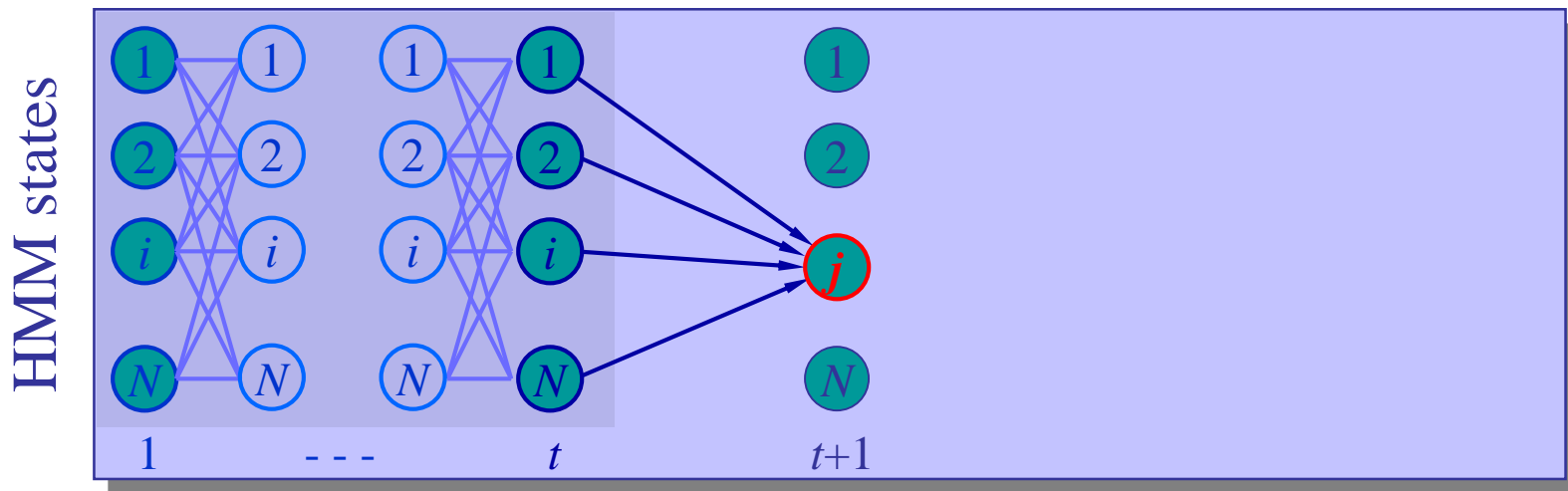
$$\alpha_1(i) = \pi_i b_i(\mathbf{x}_1) \quad 1 \leq i \leq N$$

- Recursion:

$$\alpha_{t+1}(j) = \sum_{i=1}^N \alpha_t(i) a_{ij} b_j(\mathbf{x}_{t+1}) \quad 1 \leq j \leq N, \quad t = 1, 2, \dots, T-1$$

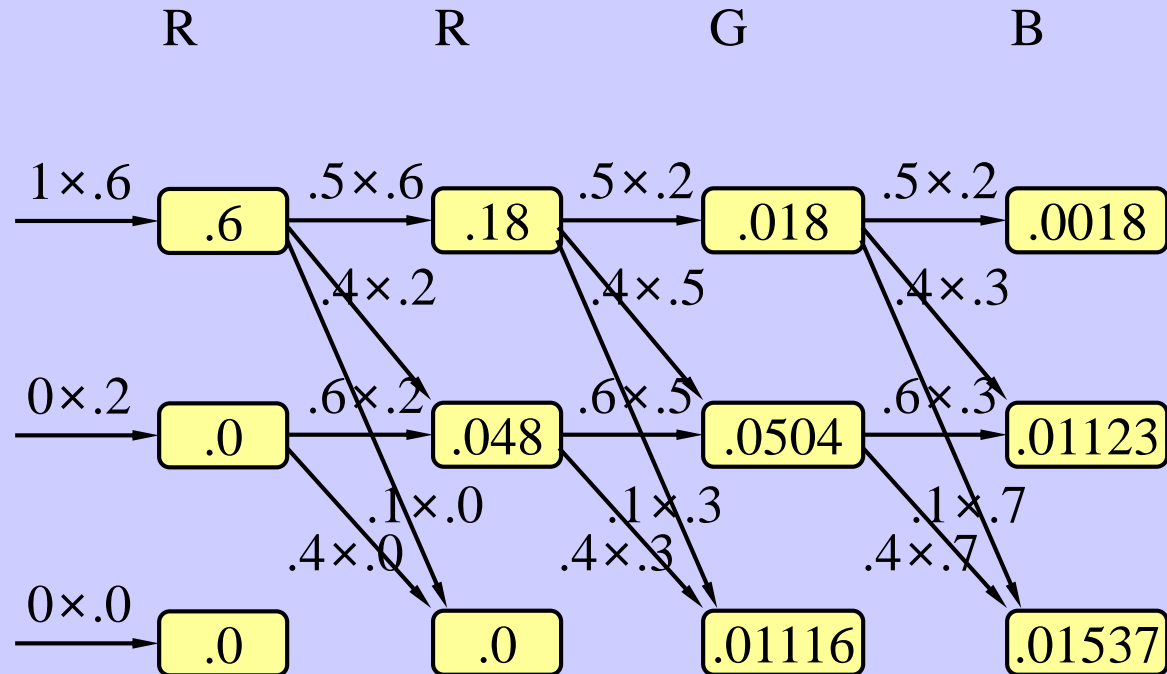
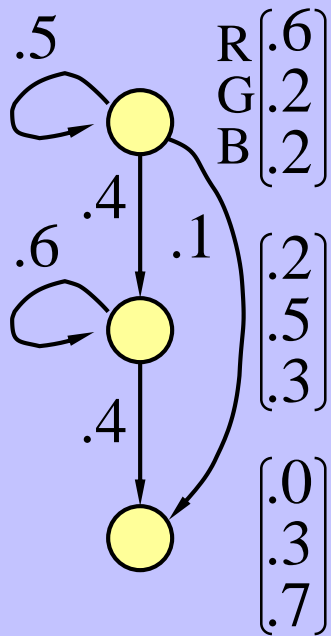
- Termination:

$$P(\mathbf{X} | \lambda) = \sum_{i=1}^N \alpha_T(i)$$



Numerical example: $P(\text{RRGB}|\lambda)$

$$\pi = [1 \ 0 \ 0]^T$$



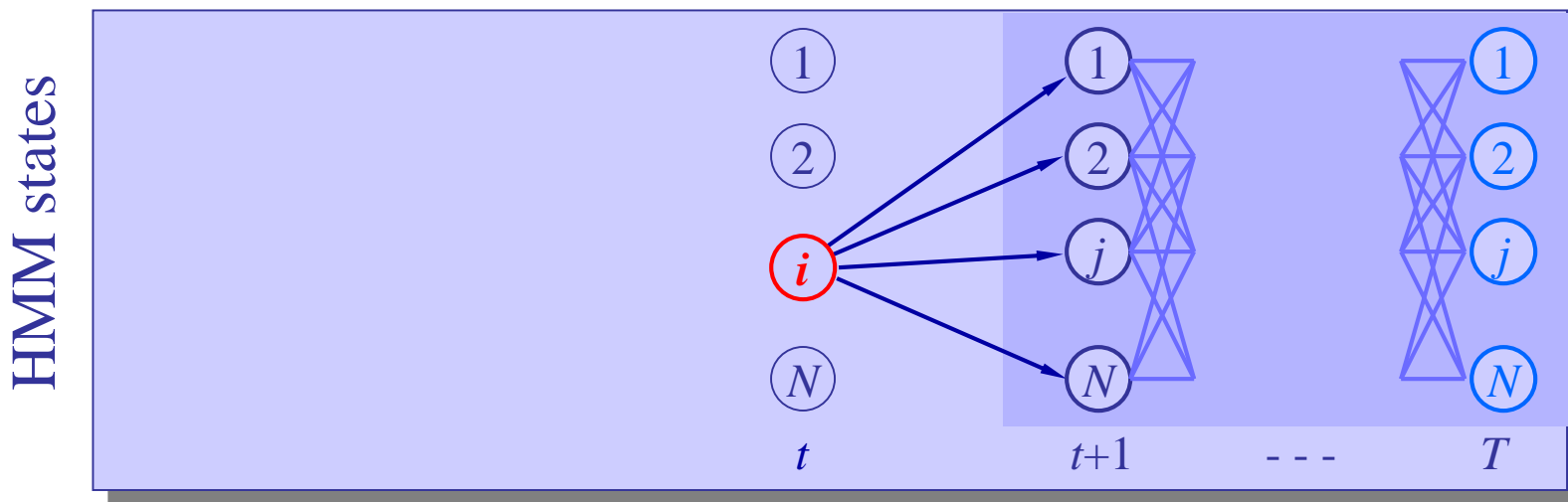
Backward procedure

- Initialization:

$$\beta_T(i) = 1 \quad 1 \leq i \leq N$$

- Recursion:

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(\mathbf{x}_{t+1}) \beta_{t+1}(j) \quad 1 \leq i \leq N, \quad t = T-1, T-2, \dots, 1$$



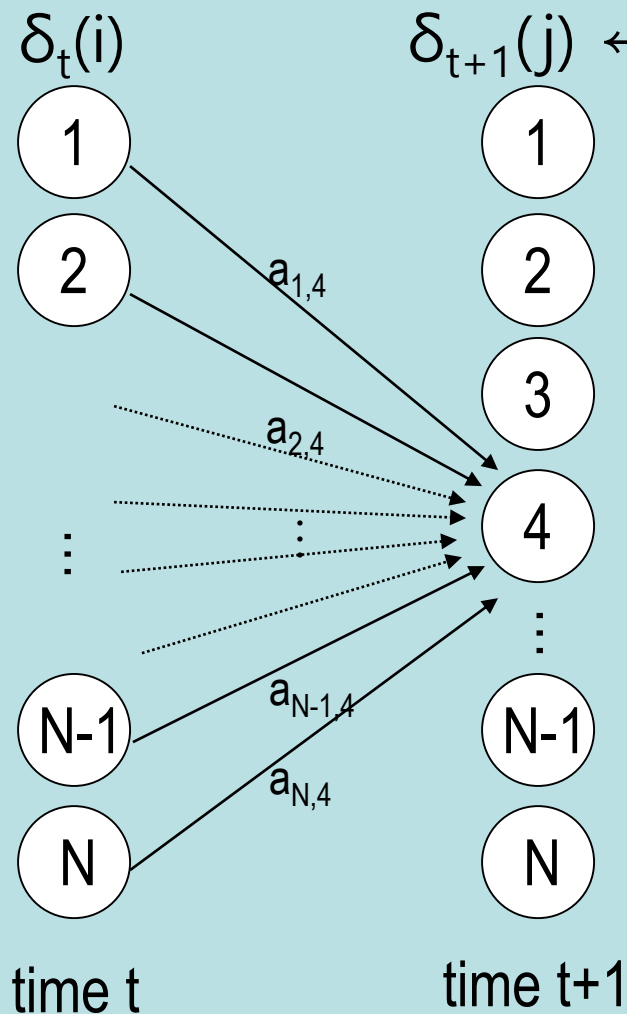
2nd Problem: What's the Most Likely State Sequence?

- Actually, there is more than one possible interpretation of “most likely state sequence”.
- One is, which states are *individually* most likely.
 - i.e., what is the most likely first state? The most likely second? And so on.
 - Note, though, that we can end up with a “sequence” that isn’t even a possible sequence through the HMM, much less a likely one.
- Another criteria is the single best state sequence.
 - i.e., **find $\text{argmax}_Q P(Q|O,\lambda)$**

Algorithm Idea

- Suppose we knew the highest probability path *ending* in each state at time step t .
- We can compute the highest probability path ending in a state at $t+1$ by considering each transition from each state at time t , and remembering only the best one.
- This is another application of the *dynamic programming principle*.

Basic Idea for $\text{argmax}_{\mathbf{Q}} P(\mathbf{Q}|\mathbf{O}, \lambda)$ Algorithm



$$\delta_{t+1}(j) \leftarrow (\max_i \delta_t(i) a_{ij}) b_{j, o_{t+1}}$$

If we know the probability of the best path to each state at time t producing the observation so far ($\delta_t(i)$), then the probability of the best path to a given state producing the next observation at the next clock tick is the *max* of this probability times the transition probability, times the probability of the right emission.

Definitions for Computing $\text{argmax}_{\mathbf{Q}} P(\mathbf{Q}|\mathbf{O},\lambda)$

- Note that $P(\mathbf{Q}|\mathbf{O},\lambda) = P(\mathbf{Q},\mathbf{O}|\lambda)/P(\mathbf{O}|\lambda)$, so maximizing $P(\mathbf{Q}|\mathbf{O},\lambda)$ is equivalent to maximizing $P(\mathbf{Q},\mathbf{O}|\lambda)$.
 - Turns out latter is a bit easier to work with.
- Define $\delta_t(i)$ as
$$\max_{q_1, q_2, \dots, q_{t-1}} P(q_1, q_2, \dots, q_{t-1}, i, o_1, o_2, \dots, o_t | \lambda).$$
- We'll use these to inductively find $P(\mathbf{Q},\mathbf{O}|\lambda)$. But how do we find the actual path?
- We'll maintain another array, $\psi_t(i)$, which keeps track of the argument that maximizes $\delta_t(i)$.

Algorithm for Computing $\text{argmax}_{\mathbf{Q}} P(\mathbf{Q}|\mathbf{O},\lambda)$

- Algorithm:

- Initialization: $\delta_1(i) \leftarrow \pi_i b_{i,o_1}, 1 \leq i \leq N$

$$\psi_1(i) \leftarrow 0$$

- Induction: $\delta_t(j) \leftarrow \max_i (\delta_{t-1}(i) a_{ij}) b_{j,o_t}$

$$\psi_t(i) \leftarrow \text{argmax}_i (\delta_{t-1}(i) a_{ij})$$
$$2 \leq t \leq T, 1 \leq j \leq N$$

- Finish: $P^* = \max_i (\delta_T(i))$ is probability of best path

$q_T^* = \text{argmax}_i (\delta_T(i))$ is best final state

- Extract path by backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), t=T-1, T-2, \dots, 1$$

- This is called the *Viterbi algorithm*.

- Note that it is very similar to the forward procedure (and hence as efficient).

2. Decoding Problem

- The best path Q^* given an input X ?
 - It can be obtained by maximizing the joint probability over state sequences
 - Path likelihood score:

$$\begin{aligned}\delta_t(j) &= \max_{q_1 q_2 \cdots q_{t-1}} P[q_1 q_2 \cdots q_{t-1}, q_t = j, x_1 x_2 \cdots x_t \mid \lambda] \\ &= \max_i \delta_{t-1}(i) a_{ij} b_j(x_t)\end{aligned}$$

$$\psi_t(j) = \hat{i} = \arg \max_i \delta_{t-1}(i) a_{ij} b_j(x_t)$$

$Q_{1,t} = q_1 q_2 \cdots q_t$: a partial (best) state sequence

- Viterbi algorithm

Viterbi algorithm

- Introduction:

$$\delta_1(i) = \pi_i b_i(\mathbf{x}_1)$$

$$\psi_1(i) = 0$$

- Recursion:

$$\delta_{t+1}(j) = \max_{1 \leq i \leq N} \delta_t(i) a_{ij} b_j(\mathbf{x}_{t+1})$$

$$\psi_{t+1}(j) = \arg \max_{1 \leq i \leq N} \delta_t(i) a_{ij}$$

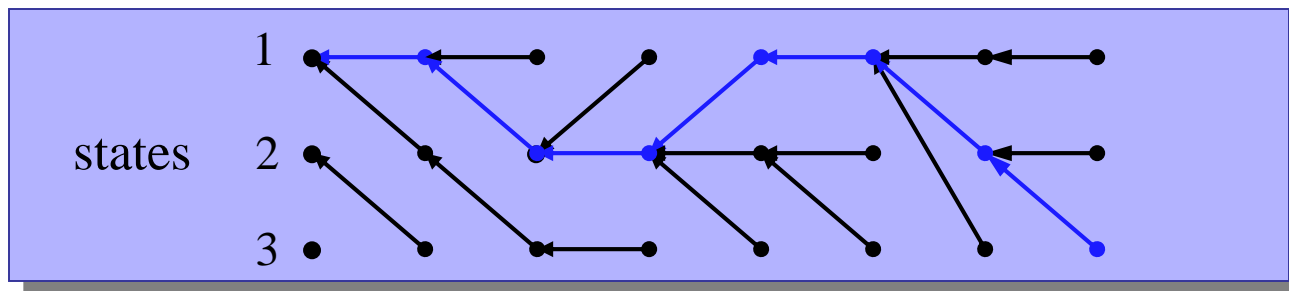
- Termination:

$$P^* = \max_{1 \leq i \leq N} \delta_T(i)$$

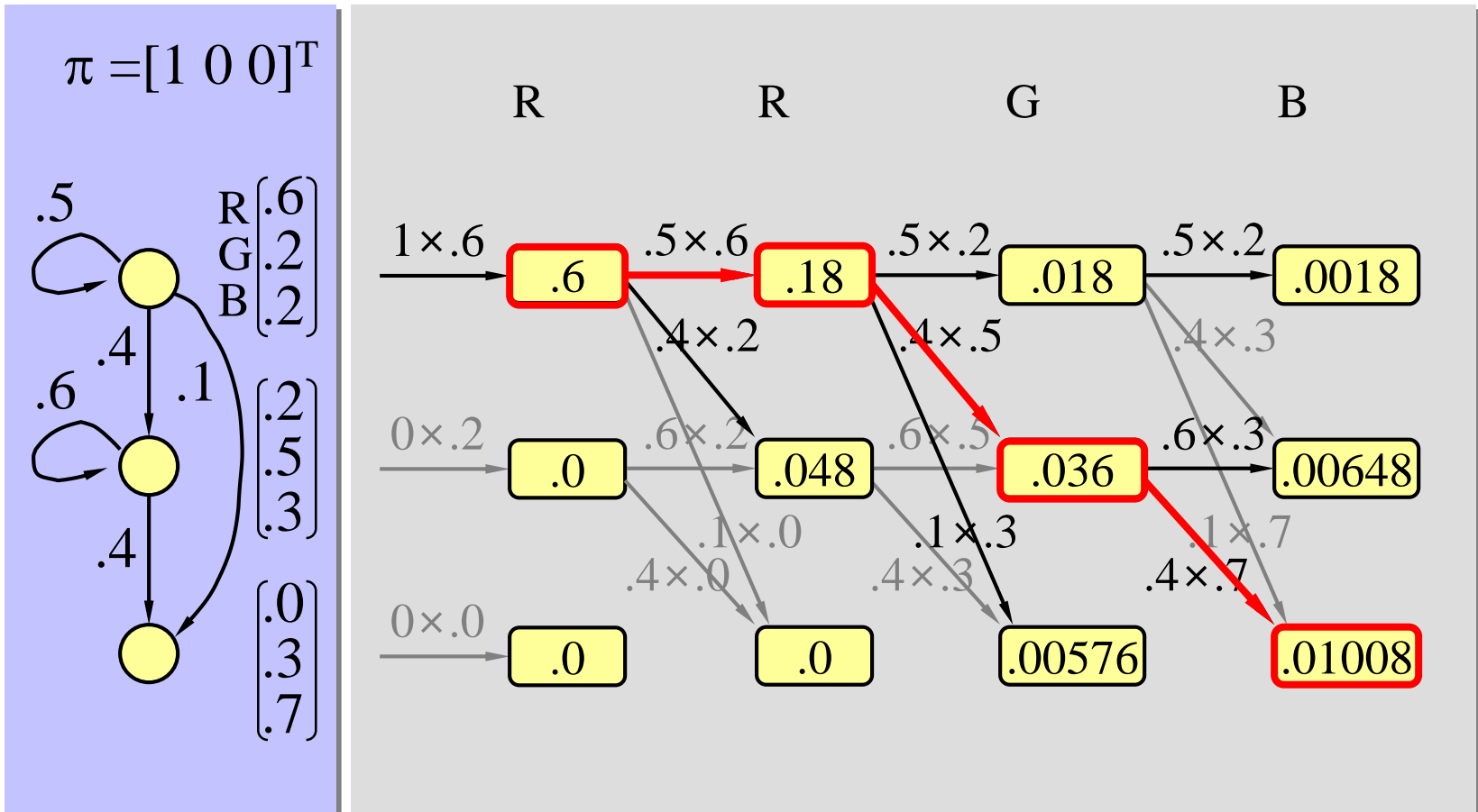
$$q_T^* = \arg \max_{1 \leq i \leq N} \delta_T(i)$$

- Path backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, \dots, 1$$



Numerical Example: $P(\text{RRGB}, Q^* | \lambda)$



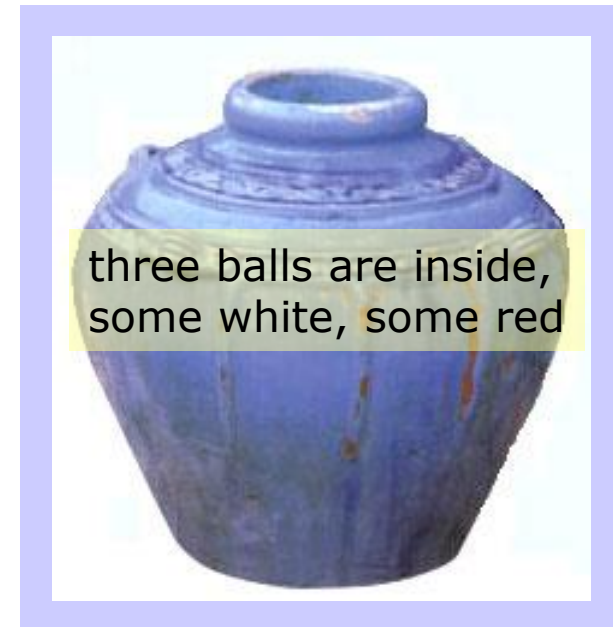
3. Model Training Problem

- Estimate $\lambda=(A,B,\pi)$ that maximizes $P(X|\lambda)$
- No analytical solution exists
- MLE + EM algorithm developed
 - Baum-Welch reestimation [Baum+68,70]
 - a local maximization using iterative procedure
 - maximizes the probability estimate of observed events
 - guarantees finite improvement
 - is based on forward-backward procedure

MLE Example

- Experiment

- Known: 3 balls inside (some white, some red; exact numbers unknown)
- Unknown: $R = \#$ red balls
- Observation: one random samples : ●● (two reds)



- Two models

- $p(\text{●●} | R=2) = {}_2C_2 \times {}_1C_0 / {}_3C_2 = 1/3$
- $p(\text{●●} | R=3) = {}_3C_2 / {}_3C_2 = 1$

- Which model?

Learning an HMM

- We will assume we have an observation \mathbf{O} , and want to know the “best” HMM for it.
- I.e., we would want to find $\operatorname{argmax}_{\lambda} P(\lambda|\mathbf{O})$, where $\lambda = (A, B, \pi)$ is some HMM.
 - I.e., what model is most likely, given the observation?
- When we have a fixed observation that we use to pick a model, we call the observation *training data*.
- Functions/values like $P(\lambda|\mathbf{O})$ are called *likelihoods*.
- What we want is the *maximum likelihood estimate*(MLE) of the parameters of our model, given some training data.
- This is an *estimate* of the parameters, because it depends for its accuracy on how representative the observation is.

Maximizing Likelihoods

- We want to know $\operatorname{argmax}_{\lambda} P(\lambda|\mathbf{O})$.
- By Bayes' rule, $P(\lambda|\mathbf{O}) = P(\lambda)P(\mathbf{O}|\lambda)/P(\mathbf{O})$.
 - The observation is constant, so it is enough to maximize $P(\lambda)P(\mathbf{O}|\lambda)$.
- $P(\lambda)$ is the prior probability of a model; $P(\mathbf{O}|\lambda)$ is the probability of the observation, given a model.
- Typically, we don't know much about $P(\lambda)$.
 - E.g., we might assume all models are equally likely.
 - Or, we might stipulate that some subclass are equally likely, and the rest not worth considering.
- We will ignore $P(\lambda)$, and simply optimize $P(\mathbf{O}|\lambda)$.
- I.e., we can find the most probable model by asking which model makes the data most probable.

Maximum Likelihood Example

- Simple example: We have a coin that may be biased. We would like to know the probability that it will come up heads.
- Let's flip it a number of times; use percentage of times it comes up heads to estimate the desired probability.
- Given m out of n trials come up heads, what probability should be assigned?
- In terms of likelihoods, we want to know $P(\lambda|\mathbf{O})$, where our model is just the simple parameter, the probability of a coin coming up heads.
- As per our previous discussion, can maximize this by maximizing $P(\lambda)P(\mathbf{O}|\lambda)$.

Maximizing $P(\lambda)P(\mathbf{O}|\lambda)$ For Coin Flips

- Note that knowing something about $P(\lambda)$ is knowing whether coins tended to be biased.
- If we don't, we just optimize $P(\mathbf{O}|\lambda)$.
- I.e., let's pick the model that makes the observation most likely.
- We can solve this analytically:
 - $P(m \text{ heads over } n \text{ coin tosses} | P(\text{Heads})=p) = {}_nC_m p^m (1-p)^{n-m}$
 - Take derivative, set equal to 0, solve for p .
- Turns out p is m/n .
 - This is probably what you guessed.
- So we have made a simple maximum likelihood estimate.

Comments

- Making a MLE seems sensible, but has its problems.
- Clearly, our result will just be an estimate.
 - but one we hope will become increasingly accurate with more data.
- Note, though, via MLE, the probability of everything we haven't seen so far is 0.
- For modeling rare events, there will never be enough data for this problem to go away.
- There are ways to smooth over this problem (in fact, one way is called “smoothing”!), but we won't worry about this now.

Back to HMMs

- MLE tells us to optimize $P(\mathbf{O}|\lambda)$.
- We know how to *compute* $P(\mathbf{O}|\lambda)$ for a given model.
 - E.g., use the “forward” procedure.
- How do we find the *best* one?
- Unfortunately, we don’t know how to solve this problem analytically.
- However, there is a procedure to find a better model, given an existing one.
 - So, if we have a good guess, we can make it better.
 - Or, start out with fully connected HMM (of given N , M) in which each state can emit every possible value; set all probabilities to random non-zero values.
- Will this guarantee us a best solution?
- No! So this is a form of ...
 - Hillclimbing!

Maximizing the Probability of the Observation

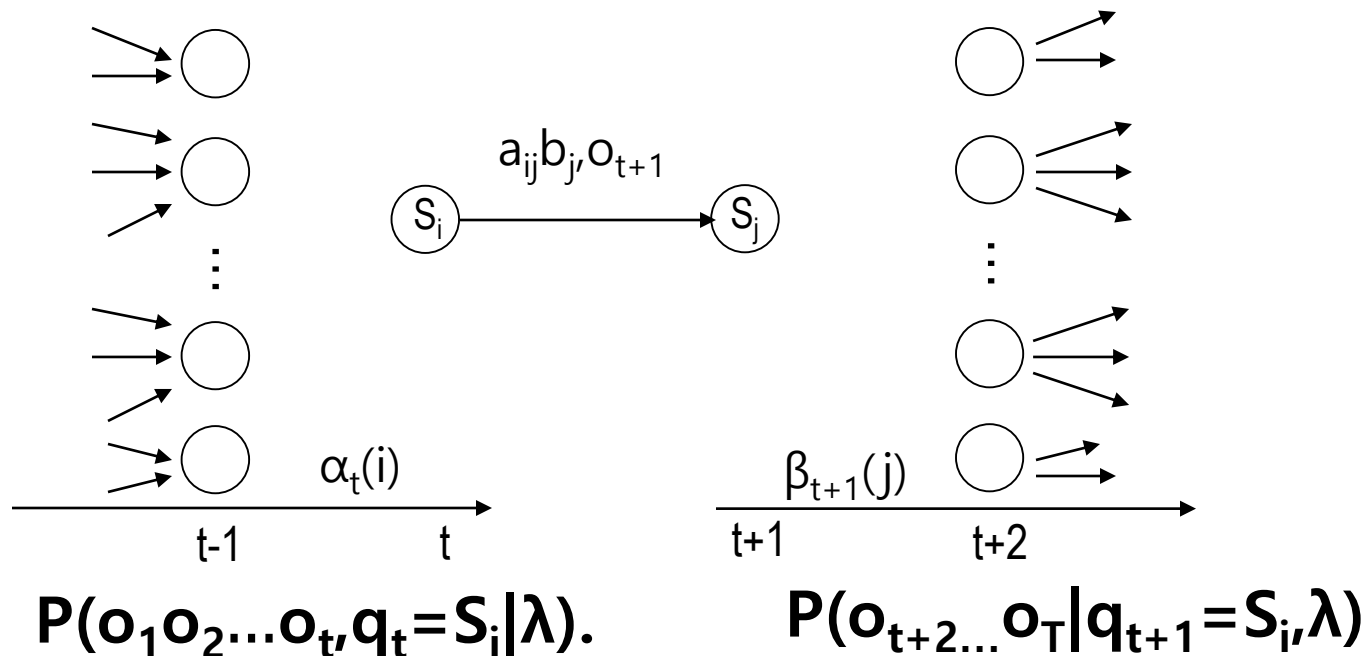
- Basic idea is:
 1. Start with an initial model, λ_{old} .
 2. Compute new λ_{new} based on λ_{old} and observation \mathbf{O} .
 3. If $P(\mathbf{O}|\lambda_{\text{new}}) - P(\mathbf{O}|\lambda_{\text{old}}) < \text{threshold}$
(or we've iterated enough), stop.
 4. Otherwise, $\lambda_{\text{old}} \leftarrow \lambda_{\text{new}}$ and go to step 2.

Increasing the Probability of the Observation

- Let's “count” the number of times, from each state, we
 - start in a given state
 - make a transition to each other state
 - emit each different symbol.given the model and the observation.
- If we knew these numbers, we can compute new probability estimates.
- We can't really “count” these.
 - We don't know for sure which path through the model was taken.
 - But we know the *probability* of each path, given the model.
- So we can compute the *expected* value of each figure.

Set Up

- Define $\xi_t(i,j) = P(q_t=S_i, q_{t+1}=S_j | \mathbf{O}, \lambda)$
 - i.e., the probability that, given observation and model, we are in state S_i at time t and state S_j at time $t+1$.
- Here is how such a transition can happen:



Computing $\xi_t(i,j)$

- $\alpha_t(i)a_{ij}b_{j,o_{t+1}}\beta_{t+1}(j) = P(q_t=S_i, q_{t+1}=S_j, \mathbf{O}|\lambda)$
- But we want $\xi_t(i,j) = P(q_t=S_i, q_{t+1}=S_j|\mathbf{O},\lambda)$
- By definition of conditional probability,
$$\xi_t(i,j) = \alpha_t(i)a_{ij}b_{j,o_{t+1}}\beta_{t+1}(j)/P(\mathbf{O}|\lambda)$$
- i.e., given
 - α , which we can compute by forward procedure,
 - β , which we can compute by backward procedure,
 - $P(\mathbf{O})$, which we can compute by forward, but also, by
$$\sum_i \sum_j \alpha_t(i)a_{ij}b_{j,o_{t+1}}\beta_{t+1}(j)$$
 - and A,B, which are given in model, – we can compute ξ .

One more preliminary...

- Define $\gamma_t(i)$ as probability of being in state S_i at time t , given the observation and the model.

- Given our definition of ξ :

$$\gamma_t(i) = \sum_{1 \leq j \leq N} \xi_t(i, j)$$

- So:

- $\sum_{1 \leq t \leq T-1} \gamma_t(i)$ = expected number of transitions from S_i
- $\sum_{1 \leq t \leq T-1} \xi_t(i, j)$ = expected number of transitions from S_i to S_j

Now We Can Reestimate Parameters

a_{ij}' = expected no. of transitions from S_i to S_j / expected no. of transitions from S_i

$$= \sum_{1 \leq t \leq T-1} \xi_t(i,j) / \sum_{1 \leq t \leq T-1} \gamma_t(i)$$

π_i' = probability of being in S_1 = $\gamma_1(i)$

b_{jk}' = expected no. of times in S_j , observing v_k / expected no. of times in S_j

$$= \sum_{1 \leq t \leq T, \text{ s.t. } O_t = v_k} \gamma_t(i) / \sum_{1 \leq t \leq T} \gamma_t(i)$$

Iterative Reestimation Formulae

\bar{a}_{ij} = expected ratio of # transitions to state j from/given state i

$$\begin{aligned} & \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} = \frac{\frac{1}{P} \sum_t \alpha_t(i) a_{ij} b_j(x_{t+1}) \beta_{t+1}(j)}{\frac{1}{P} \sum_j \sum_t \alpha_t(i) a_{ij} b_j(x_{t+1}) \beta_{t+1}(j)} \end{aligned}$$

$$\bar{b}_{jk} = \frac{\sum_t \gamma_t(j) \delta(x_t, k)}{\sum_t \gamma_t(j)} = \frac{\frac{1}{P} \sum_t \alpha_t(j) \beta_t(j) \delta(x_t, k)}{\frac{1}{P} \sum_t \alpha_t(j) \beta_t(j)}$$

$$\bar{\pi}_i = \gamma_1(i) = \frac{\alpha_1(i) \beta_1(i)}{P}$$

repeat

This Algorithm is Known As Baum-Welch Algorithm

- or *Forward-backward* algorithm
- It was proven (by Baum et al.) that this reestimation procedure leads to increased likelihood.
- But remember, it only guarantees climbing to a local maximum!
- It is a special case of a very general algorithm for incremental improvement by iteratively
Baum-Welch Algorithm
 - computing expected values some unobservables (e.g., transitions and state emissions),
 - using these to compute new MLEs of parameters (A, B, π)
 - The general procedure is called *Expectation-Maximization*, or the *EM algorithm*.

Implementation Considerations

- This doesn't quite work as advertised in practice.
- One problem: $\alpha_t(i)$, $\beta_t(j)$ get smaller with length of observation, and eventually underflow.
- This is readily fixed by “normalizing” these values.
 - I.e., instead of computing $\alpha_t(i)$, at each time step, compute $\alpha_t(i)/\sum_i \alpha_t(i)$.
 - Turns out that if you also used the $\sum_i \alpha_t(i)$ s to scale the $\beta_t(j)$ s, you get a numerically nice value, although it doesn't have a nice probabilities interpretation;
 - Yet, when you use both scaled values, the rest of the $\xi_t(i,j)$ computation is exactly the same.
- Another: Sometimes estimated probabilities will still get very small, and it seems better to not let these fall all the way to 0.

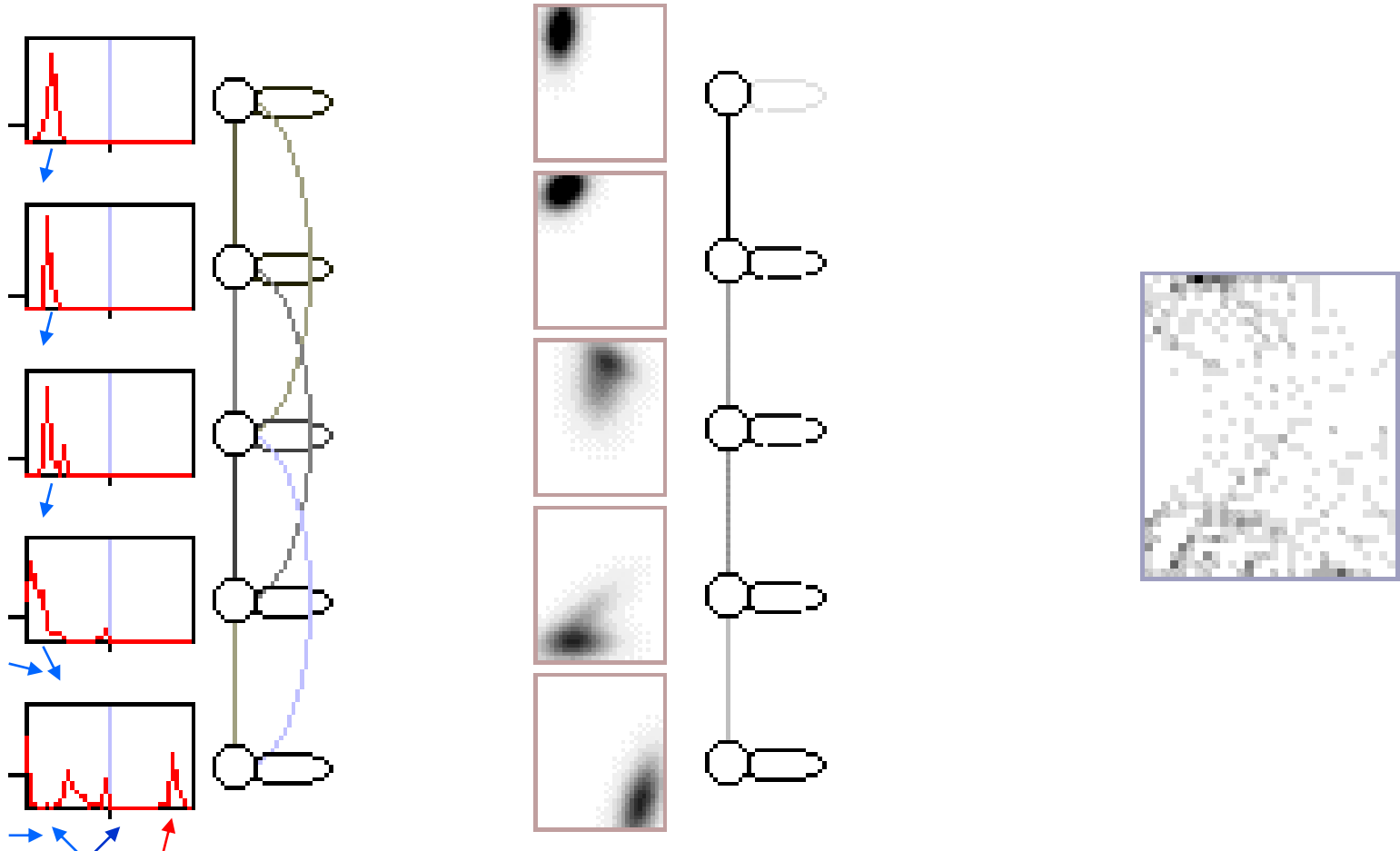
Other issues

- Other method of training
 - MAP estimation – for adaptation
 - MMI estimation
 - MDI estimation
 - Viterbi training
 - Discriminant/reinforcement training

- Other types of parametric structure
 - Continuous density HMM (CHMM)
 - More accurate, but much more parameters to train
 - Semi-continuous HMM
 - Mix of CHMM and DHMM, using parameter sharing
 - State-duration HMM
 - More accurate temporal behavior
- Other extensions
 - HMM+NN, Autoregressive HMM
 - 2D models: MRF, Hidden Mesh model, pseudo-2D HMM

Graphical DHMM and CHMM

- Models for '5' and '2'



Statistical Decision Making System

- Statistical K -class pattern recognition
 - Prepare K HMMs trained with samples
 - $\lambda_1, \dots, \lambda_K$
 - Integrate measurement and a priori knowledge
 - $P(X|\lambda_k)$
 - $P(\lambda_k)$
 - Decision

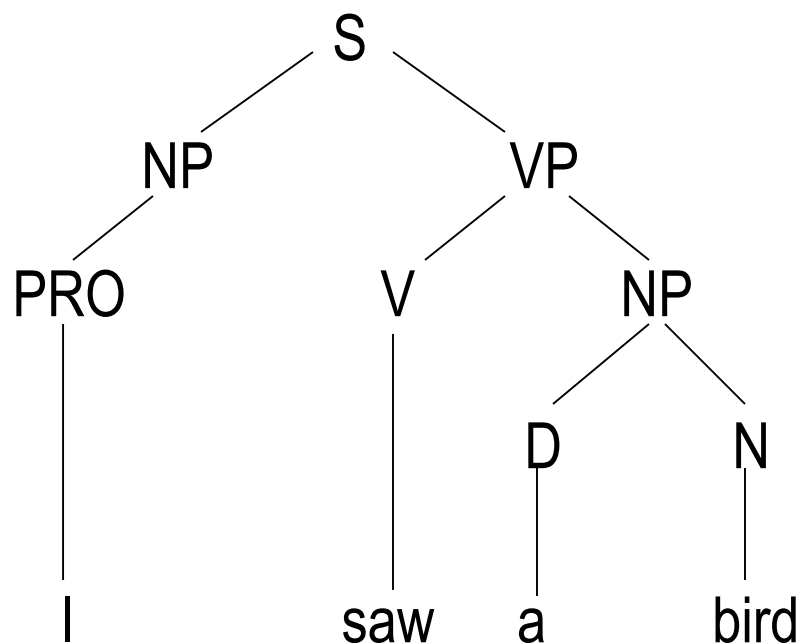
Choose ω_k if $P(\omega_k | \mathbf{X}) > P(\omega_i | \mathbf{X}) \quad \forall i \neq k$

An Application of HMMs: “Part of Speech” Tagging

- A problem: Word sense disambiguation.
 - I.e., words typically have multiple senses; need to determine which one a given word is being used as.
- For now, we just want to guess the “part of speech” of each word in a sentence.
 - Linguists propose that words have properties that are a function of their “grammatical class”.
 - Grammatical classes are things like verb, noun, preposition, determiner, etc.
 - Words often have multiple grammatical classes.
 - » For example, the word “rock” can be a noun or a verb.
 - » Each of which can have a number of different meanings.
- We want an algorithm that will “tag” each word with its most likely part of speech.
 - Why? Helping with parsing, pronunciation.

Parsing, Briefly

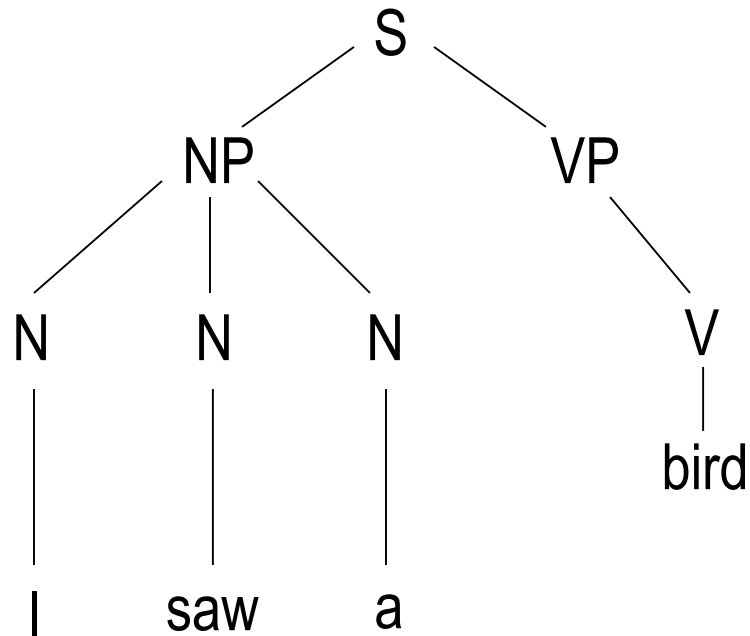
- Consider a simple sentence:
“I saw a bird.”
- Let's “diagram” this sentences, making a parse tree:



However

- Each of these words is listed in the dictionary as having multiple POS entries:
 - “saw”: noun, verb
 - “bird”: noun, intransitive verb (“to catch or hunt birds, birdwatch”)
 - “I”: pronoun, noun (the letter “I”, the square root of -1 , something shaped like an I (I-beam), symbol (I-80, Roman numeral, iodine)
 - “a”: article, noun (the letter “a”, something shaped like an “a”, the grade “A”), preposition (“three times a day”), French pronoun.

Moreover, there is a parse!



What's It Mean?

- It's nonsense, of course.
- The question is, can we avoid such nonsense cheaply.
- Note that this parse corresponds to a very unlikely set of POS tags.
- So, just restricting our tags to reasonably probably ones might eliminate such silly options.

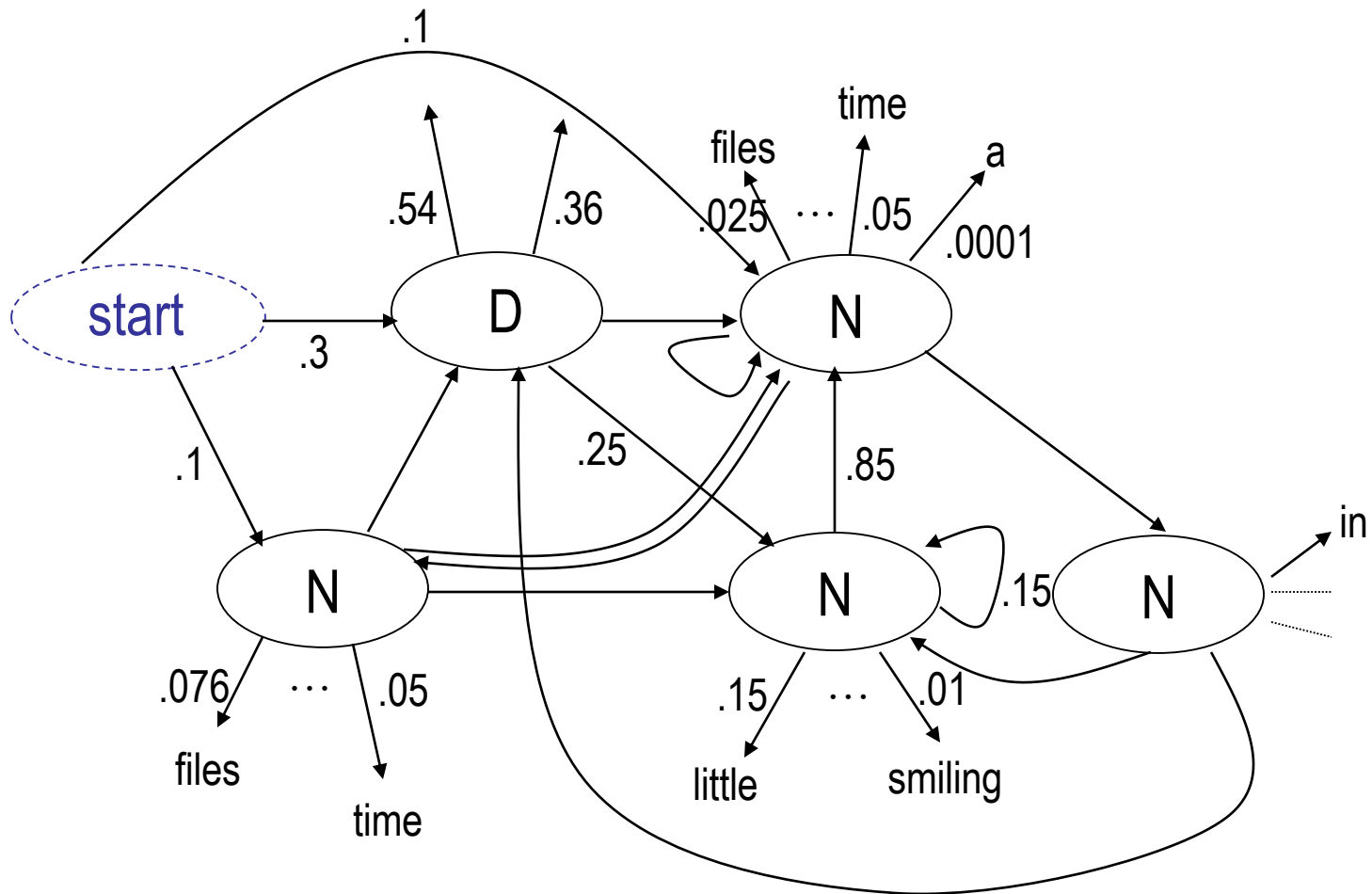
Solving the Problem

- A “baseline” solution: Just count the frequencies in which each word occurs as a particular part of speech.
 - Using “tagged corpora”.
- Pick most frequent POS for each word.
- How well does it work?
- Turns out it will be correct about 91% of the time.
- Good?
- Humans will agree with each other about 97-98% of the time.
- So, room for improvement.

A Better Algorithm

- Make POS guess depend on context.
- For example, if the previous word were “the”, then the word “rock” is much more likely to be occurring as a noun than as a verb.
- We can incorporate context by setting up an HMM in which the hidden states are POSs, and the words the emissions in those states.

HMM For POS Tagging: Example



HMM For POS Tagging

- First-order HMM equivalent to POS bigrams
- Second-order equivalent to POS trigrams.
- Generally works well if there is a hand-tagged corpus from which to read off the probabilities.
 - Lots of detailed issues: smoothing, etc.
- If none available, train using Baum-Welch.
- Usually start with some constraints:
 - E.g., start with 0 emissions for words not listed in dictionary as having a given POS; estimate transitions.
- Best variations get about 96-97% accuracy, which is approaching human performance.