COMP 182 Algorithmic Thinking

Markov Chains and Hidden Markov Models

Luay Nakhleh
Computer Science
Rice University

- What is $p(01110000)$?
- Assume:
  - 8 independent Bernoulli trials with success probability of α?
- Answer:
  - $(1-\alpha)^5\alpha^3$

- However, what if the assumption of independence doesn't hold?
- That is, what if the outcome in a Bernoulli trial depends on the outcomes of the trials the preceded it?

Markov Chains

- Given a sequence of observations $X_1, X_2, \ldots, X_T$
- The basic idea behind a Markov chain (or, Markov model) is to assume that X_t captures all the relevant information for predicting the future.
- In this case:

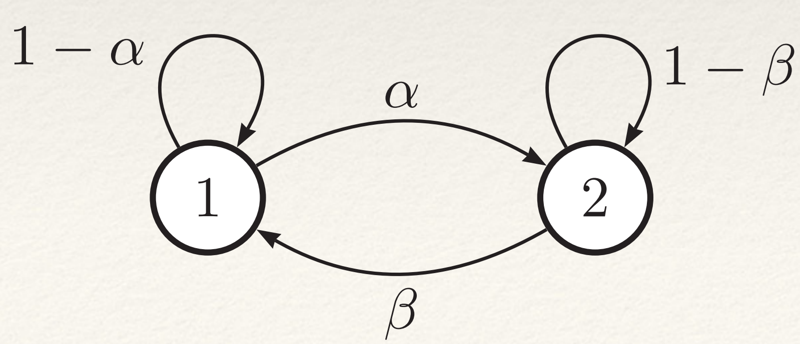
$$p(X_1 X_2 \ldots X_T) = p(X_1)p(X_2|X_1)p(X_3|X_2)\cdots p(X_T|X_{T-1}) = p(X_1)\prod_{t=2}^{T} p(X_t|X_{t-1})$$

Markov Chains

- When X_t is discrete, so $X_t \in \{1,\ldots,K\}$, the conditional distribution $p(X_t \mid X_{t-1})$ can be written as a $K \times K$ matrix, known as the transition matrix A, where $A_{ij} = p(X_t = j \mid X_{t-1} = i)$ is the probability of going from state i to state j.
- Each row of the matrix sums to one, so this is called a stochastic matrix.

Markov Chains

- A finite-state Markov chain is equivalent to a stochastic automaton.
- One way to represent a Markov chain is through a state transition diagram



$$\mathbf{A} = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix}$$

Markov Chains

- The A_{ij} element of the transition matrix specifies the probability of getting from i to j in one step.
- The n-step transition matrix $A(n)$ is defined as

$$A_{ij}(n) = p(X_{t+n} = j | X_t = i)$$

Markov Chains

- Obviously, $A(1)=A$.
- The Chapman-Kolmogorov equations state that

$$A_{ij}(m+n) = \sum_{k=1}^{K} A_{ik}(m)A_{kj}(n)$$

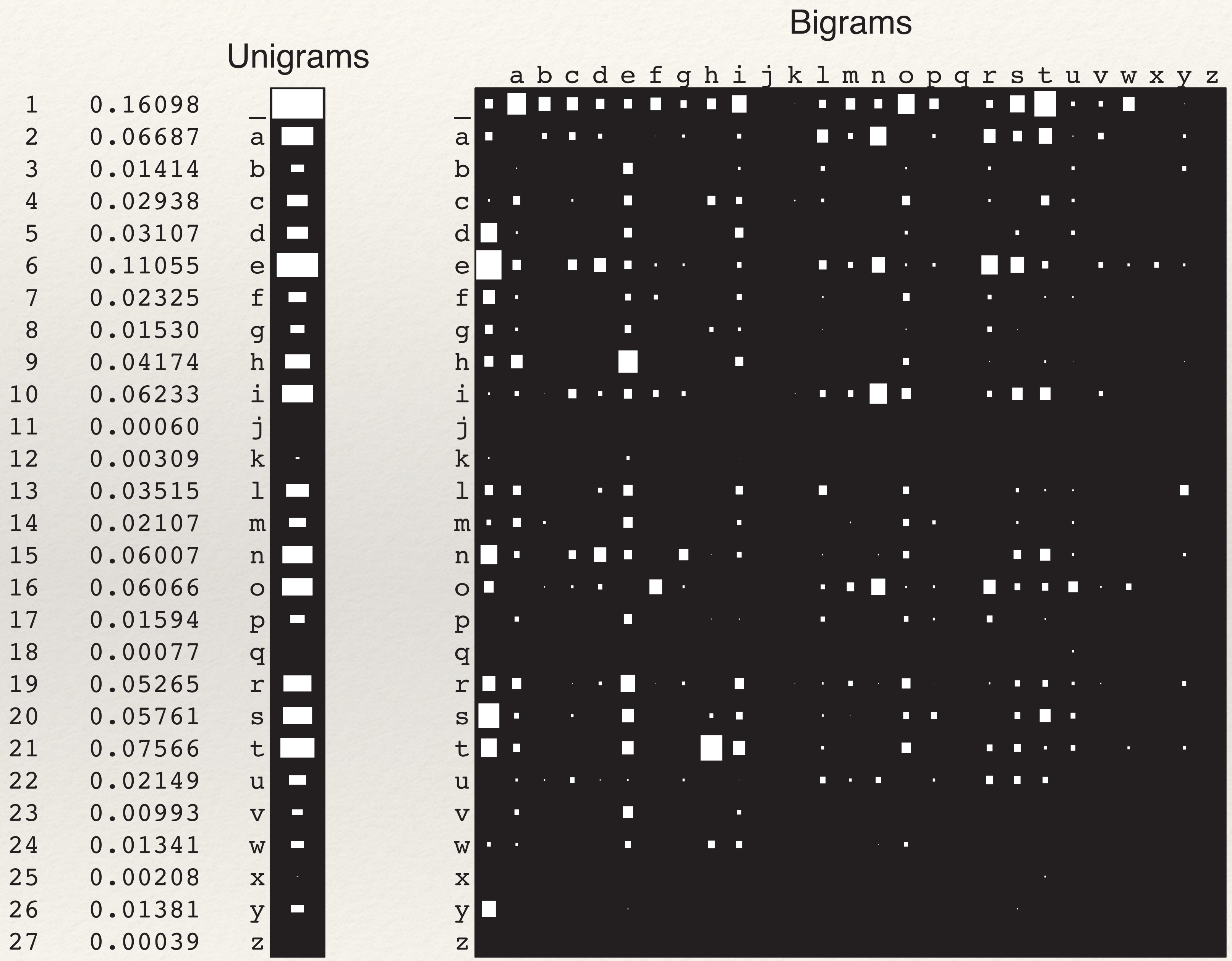
Markov Chains

- Therefore, we have $A(m+n)=A(m)A(n)$.
- Hence, $A(n)=AA(n\text{-}1)=AAA(n\text{-}2)=\ldots=A^n$.
- Thus, we can simulate multiple steps of a Markov chain by "powering up" the transition matrix.

Language Modeling

- One important application of Markov models is to make statistical language models, which are probability distributions over sequences of words.
- We define the state space to be all the words in English (or, the language of interest).
- The probabilities $p(X_t=k)$ are called unigram statistics.
- If we use a first-order Markov model, then $p(X_t=k \mid X_{t-1}=j)$ is called a bigram model.

Language Modeling



Unigram and bigram counts for Darwin's

On the Origin of Species

Parameter Estimation for Markov Chains

- The parameters of a Markov chain, denoted by θ, consist of the transition matrix (A) and the distribution on the initial states (π).
- We want to estimate these parameters from a training data set.
- Such a data set consists of sequences $X^1, X^2, \ldots, X^m$.
- Sequence X^k has length L_k.

Parameter Estimation for Markov Chains

- The maximum likelihood estimate (MLE) of the parameters is easy to obtain from the data:

$$N_j^1 = \sum_{i=1}^{m} \mathbb{I}(X_1^i = j) \quad N_{jk} = \sum_{i=1}^{m} \sum_{t=1}^{L_i - 1} \mathbb{I}(X_t^i = j, X_{t+1}^i = k)$$

$$\hat{\pi}_j = \frac{N_j^1}{\sum_i N_i^1} \qquad \hat{A}_{jk} = \frac{N_{jk}}{\sum_{k'} N_{jk'}}$$

Parameter Estimation for Markov Chains

- The maximum likelihood estimate (MLE) of the parameters is easy to obtain from the data:

$$N_j^1 = \sum_{i=1}^{m} \mathbb{I}(X_1^i = j) \qquad N_{jk} = \sum_{i=1}^{m} \sum_{t=1}^{L_i - 1} \mathbb{I}(X_t^i = j, X_{t+1}^i = k)$$

$$\hat{\pi}_j = \frac{N_j^1}{\sum_i N_i^1} \qquad \hat{A}_{jk} = \frac{N_{jk}}{\sum_{k'} N_{jk'}}$$

Indicator function: $\mathbb{I}(e) = \begin{cases} 1 & \text{if } e \text{ is true} \\ 0 & \text{if } e \text{ is false} \end{cases}$

Parameter Estimation for Markov Chains

- It is very important to handle zero counts properly (this is called <u>smoothing</u>).

Hidden Markov Models

- A hidden Markov model, or HMM, consists of a discrete-time, discrete state Markov chain, with hidden states $Z_t \in \{1,\ldots,K\}$, plus an observation model $p(X_t \mid Z_t)$ (emission probabilities).
- The corresponding joint distribution has the form

$$p(Z_{1:T}, X_{1:T}) = \left[p(Z_1) \prod_{t=2}^{T} p(Z_t | Z_{t-1}) \right] \left[\prod_{t=1}^{T} p(X_t | Z_t) \right]$$

Hidden Markov Models

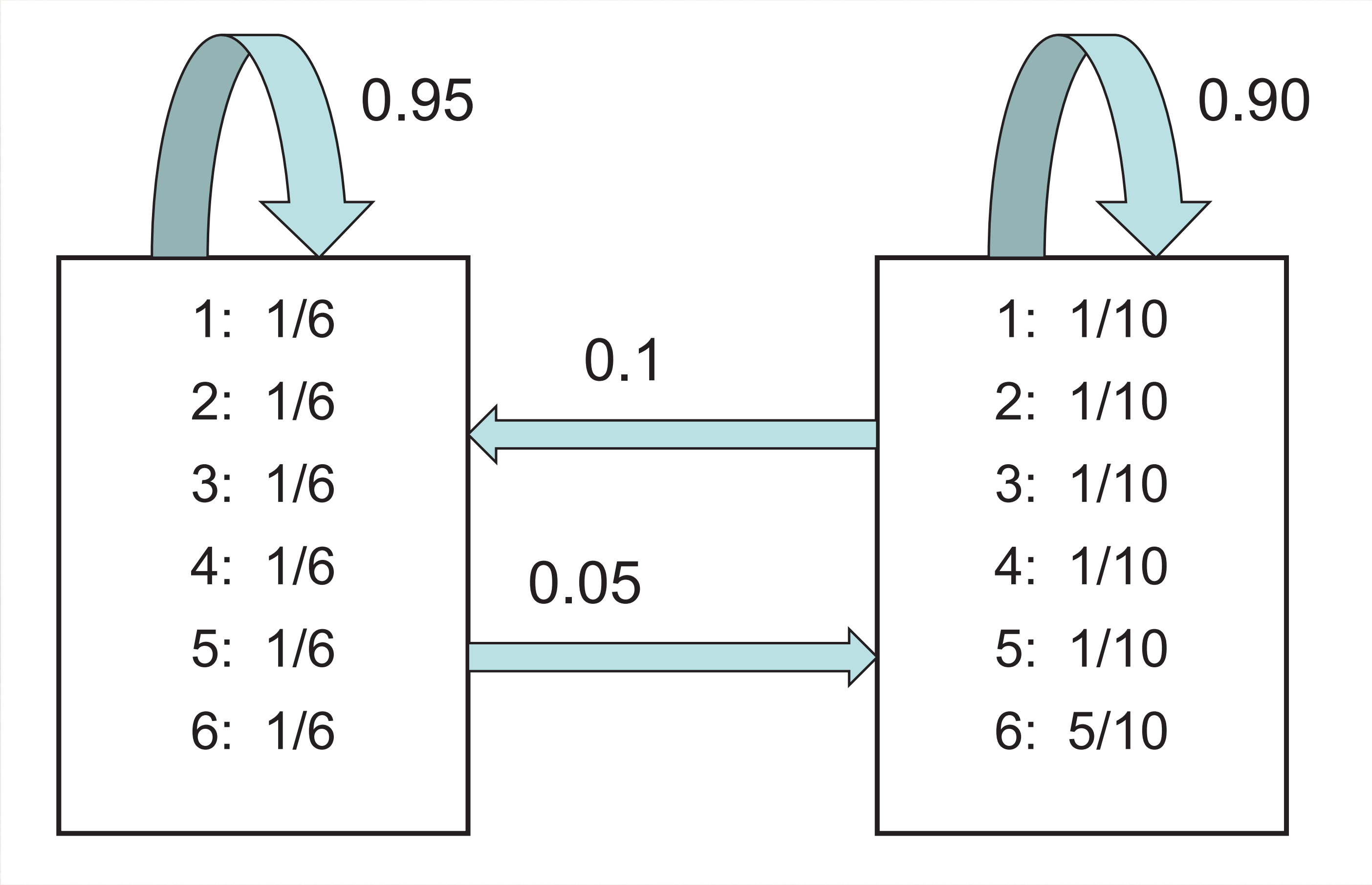
- Example: The "occasionally dishonest casino"
  - Most of the time, the casino uses a fair die ($Z=1$), but occasionally it switches for a short period to a loaded die ($Z=2$) that is skewed towards face *6*.

Hidden Markov Models

- Example: The "occasionally dishonest casino"
  - Most of the time, the casino uses a fair die (*Z*=*1*), but occasionally it switches for a short period to a loaded die (Z=2) that is skewed towards face *6*.

```
Rolls:    664153216162115234653214356634261655234232315142464156663246
Die:      LLLLLLLLLLLLLLFFFFFFLLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFLLLLLLLL
```

Hidden Markov Models



Hidden Markov Models

- Two important questions:
  - How do we obtain the HMM?
  - Given the observations, how do we find the hidden states that generated them?

Learning for HMMs

- The task of estimating the parameters (initial state distribution, transition, and emission probabilities)
- The more general case: Determining the set of states as well
  - In practice, the states are often known

Learning for HMMs

- Training from fully observed data:
  - If we observe hidden state sequences, we can compute MLEs for the parameters, similar to the case of Markov chains
- Training when hidden state sequences are not observed is much harder:
  - The Baum-Welch algorithm, which is an expectation-maximization (EM) algorithm, is used in this case (well beyond the scope of this course, though)

Computing the State Sequence

- The most probable sequence of states can be computed as

$$Z^* \leftarrow \text{argmax}_Z p(Z, X)$$

The same as $Z^* \leftarrow \text{argmax}_Z p(Z|X)$

Computing the State Sequence

- Z* can be computed efficiently using a dynamic programming algorithm, called the Viterbi algorithm.
- Define

$$v[\ell, i] = \max_{Z_{1:i-1}} p(Z_{1:i-1}, Z_i = \ell | X_{1:i})$$

Computing the State Sequence

Algorithm 1: Viterbi.

Input: A first-order HMM M with states $\mathcal{S} = \{1, 2, \ldots, K\}$ given by its transition matrix A, emission probability matrix E (alphabet Σ), and probability distribution π on the (initial) states; a sequence X of length L (indexed from 0 to $L-1$).

Output: A sequence Z, with $|Z| = |X|$, that maximizes $p(Z, X)$.

$v[\ell, 0] \leftarrow (\pi_\ell \cdot E_\ell(X_0))$ for every $\ell \in \mathcal{S}$;
for $i \leftarrow 1$ **to** $L-1$ **do**
    foreach $\ell \in \mathcal{S}$ **do**
        $v[\ell, i] \leftarrow E_\ell(X_i) \cdot \max_{\ell' \in \mathcal{S}}(v[\ell', i-1] \cdot A_{\ell', \ell})$;
        $bp[\ell, i] \leftarrow \mathrm{argmax}_{\ell' \in \mathcal{S}}(v[\ell', i-1] \cdot A_{\ell', \ell})$;
$Z_{L-1} \leftarrow \mathrm{argmax}_{\ell' \in \mathcal{S}} v[\ell', L-1]$;
for $i \leftarrow L-2$ **downto** 0 **do**
    $Z_i \leftarrow bp[Z_{i+1}, i+1]$;
return Z

Computing the State Sequence

Algorithm 1: Viterbi.

Input: A first-order HMM M with states $\mathcal{S} = \{1, 2, \ldots, K\}$ given by its transition matrix A, emission probability matrix E (alphabet Σ), and probability distribution π on the (initial) states; a sequence X of length L (indexed from 0 to $L-1$).
Output: A sequence Z, with $|Z| = |X|$, that maximizes $p(Z, X)$.

$v[\ell, 0] \leftarrow (\pi_\ell \cdot E_\ell(X_0))$ for every $\ell \in \mathcal{S}$;
for $i \leftarrow 1$ **to** $L - 1$ **do**
  foreach $\ell \in \mathcal{S}$ **do**
    $v[\ell, i] \leftarrow E_\ell(X_i) \cdot \max_{\ell' \in \mathcal{S}}(v[\ell', i-1] \cdot A_{\ell',\ell})$;
    $bp[\ell, i] \leftarrow \mathrm{argmax}_{\ell' \in \mathcal{S}}(v[\ell', i-1] \cdot A_{\ell',\ell})$;
$Z_{L-1} \leftarrow \mathrm{argmax}_{\ell' \in \mathcal{S}} v[\ell', L-1]$;
for $i \leftarrow L - 2$ **downto** 0 **do**
  $Z_i \leftarrow bp[Z_{i+1}, i+1]$;
return Z

If there's an 'end' state, replace by:

$$Z_{L-1} \leftarrow \mathrm{argmax}_{\ell' \in S}(v[\ell', L-1] \times A_{\ell', end})$$

Questions?